XMC4700/XMC4800 errata sheet

Marking/Step: ES-AA, AA

Package: PG-LQFP-100/144, PG-LFBGA-196

About this document

Scope and purpose

This document describes the errata for XMC4700/4800 product family.

Intended audience

This document is intended for users who plan to use XMC4700/4800 devices in an application.

Table of contents

About this document **1**
Table of contents **2**
1 Overview **4**
2 Functional deviations **10**
2.1 ADC_AI.008 10
2.2 ADC_AI.016 10
2.3 ADC_CM.001 11
2.4 ADC_CM.002 12
2.5 ADC_TC.064 13
2.6 CCU8_AI.003 13
2.7 CCU8_AI.006 15
2.8 CCU_AI.005 16
2.9 CCU_AI.006 17
2.10 CCU_AI.008 17
2.11 CPU_CM.001 18
2.12 CPU_CM.004 19
2.13 CPU_CM.005 20
2.14 DAC_CM.003 21
2.15 DEBUG_CM.002 21
2.16 DSD_AI.001 21
2.17 DTS_CM.001 22
2.18 EBU_CM.001 22
2.19 ETH_AI.001 23
2.20 ETH_AI.002 23
2.21 ETH_AI.003 24
2.22 FCE_CM.001 25
2.23 LEDTS_AI.001 25
2.24 PARITY_CM.001 29
2.25 PARITY_CM.002 31
2.26 PORTS_CM.001 31
2.27 PORTS_CM.005 31
2.28 PORTS_CM.006 32
2.29 PORTS_CM.007 33
2.30 POSIF_AI.001 33
2.31 SCU_CM.003 34
2.32 SCU_CM.006 35
2.33 SCU_CM.021 35
2.34 SDMMC_CM.001 36
2.35 SDMMC_CM.002 36
2.36 SDMMC_CM.004 37
2.37 SDMMC_CM.005 37
2.38 SDMMC_CM.006 38
2.39 SDMMC_CM.007 38
2.40 SDMMC_CM.008 39
2.41 SDMMC_CM.009 39
2.42 STARTUP_CM.002 39
2.43 STARTUP_CM.003 40

Table of contents

2.44 USB_CM.002 40
2.45 USB_CM.003 40
2.46 USB_CM.005 41
2.47 USIC_AI.008 41
2.48 USIC_AI.014 42
2.49 USIC_AI.017 42
2.50 USIC_AI.018 42
2.51 USIC_AI.020 43
2.52 USIC_AI.021 43
2.53 WDT_CM.001 44
3 Application hints 45
3.1 ADC_AI.H003 45
3.2 ADC_AI.H004 46
3.3 ADC_AI.H008 46
3.4 ADC_TC.H011 47
3.5 ETH_AI.H001 47
3.6 ECAT_CM.H001 47
3.7 ECAT_CM.H002 47
3.8 MultiCAN_AI.H005 48
3.9 MultiCAN_AI.H006 48
3.10 MultiCAN_AI.H007 49
3.11 MultiCAN_AI.H008 49
3.12 MultiCAN_AI.H009 49
3.13 MultiCAN_TC.H003 50
3.14 MultiCAN_TC.H004 50
3.15 PORTS_CM.H001 51
3.16 PORTS_CM.H002 51
3.17 RESET_CM.H001 52
3.18 USIC_AI.H004 52
4 Documentation updates 53
4.1 WDT_CM.D001 53
4.2 MultiCAN_CM.D001 53
4.3 MPU_CM.D001 54
4.4 STARTUP_CM.D003 54
Revision history 56
Disclaimer 57

1 Overview

Document ID is **056/18**.

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	V1.3	July 2016
XMC4700/XMC4800 Data Sheet	V1.3	December 2024

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

Note:

1. *The errata described in this sheet applies 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. 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**.**Type Number**:

- **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

Table 2 Errata fixed in this step

Errata	Short Description	Change
- no previous step -		

Table 3 Errata fixed compared to XMC4500 AC step (contd)

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
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
USIC_AI.005	Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time	Fixed
USIC_AI.006	Dual SPI format not supported	Fixed
USIC_AI.007	Protocol-related argument and error bits in register RBUFSR contain incorrect values following a received data word	Fixed

Errata	Short description	Change
USIC_AI.009	Baud rate generator interrupt cannot be used	Fixed
USIC_AI.010	Minimum and maximum supported word and frame length in multi-IO SSC modes	Fixed
USIC_AI.011	Write to TBUF01 has no effect	Fixed
USIC_AI.013	SCTR register bit fields DSM and HPCDIR are not shadowed with start of data word transfer	Fixed
USIC_AI.015	Wrong generation of FIFO standard transmit/receive buffer events when TBCTR.STBTEN/RBCTR.SRBTEN = 1	Fixed
USIC_AI.016	Transmit parameters are updated during FIFO buffer bypass	Fixed

Table 4 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	–	10
ADC_AI.016	No Channel Interrupt in Fast Compare Mode with GLOBRES	–	10
ADC_CM.001	Conversion results can be wrong if groups are not synchronized	–	11
ADC_CM.002	Converter diagnostics not functional	–	12
ADC_TC.064	Effect of conversions in 10-bit fast compare mode on post-calibration	–	13
CCU8_AI.003	CCU8 Parity Checker Interrupt Status is cleared automatically by hardware	–	13
CCU8_AI.006	Timer concatenation does not work when using external count signal	–	15
CCU_AI.005	CCU4 External IP clock Usage	–	16
CCU_AI.006	Value update not usable in period dither mode	–	17
CCU_AI.008	Clock ratio limitation when using MCSS inputs	–	17
CPU_CM.001	Interrupted loads to SP can cause erroneous behavior	–	18
CPU_CM.004	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used	–	19
CPU_CM.005	Store immediate overlapping exception return operation might vector to incorrect interrupt	–	20
DAC_CM.003	FIFO usage limitation in “Data Processing Mode”	–	21

1 Overview

Functional deviation	Short description	Chg	Pg
DEBUG_CM.002	CoreSight logic only reset after power-on reset	–	21
DSD_AI.001	Possible Result Overflow with Certain Decimation Factors	–	21
DTS_CM.001	DTS offset calibration value limitations	–	22
EBU_CM.001	32-bit memory with byte write capability needs address ranges 0 and 1 enabled	–	22
ETH_AI.001	Incorrect IP Payload Checksum at incorrect location for IPv6 packets with Authentication extension header	–	23
ETH_AI.002	Incorrect IP Payload Checksum Error status when IPv6 packet with Authentication extension header is received	–	23
ETH_AI.003	Overflow Status bits of Missed Frame and Buffer Overflow counters get cleared without a Read operation	–	24
FCE_CM.001	Result value is wrong if read directly after last write	–	25
LEDTS_AI.001	Delay in the update of FNCTL.PADT bit field	–	25
PARITY_CM.001	Parity error signaling deviation in immediate write/read sequences	–	29
PARITY_CM.002	Clock limitations for ETH and SDMMC modules when using parity check of module SRAMs	–	31
PORTS_CM.001	P15_PDISC.[4,5] register bits cannot be written	–	31
PORTS_CM.005	Different PORT register reset values after module reset	–	31
PORTS_CM.006	PORT driver strength register Pn_PDR not writable for Ports 7,8,9	–	32
PORTS_CM.007	P14 and P15 cannot be used in boundary scan test	–	33
POSIF_AI.001	Input Index signal from Rotary Encoder is not decoded when the length is 1/4 of the tick period	–	33
SCU_CM.003	The state of HDCR.HIB bit of HCU gets updated only once in the register mirror after reset release	–	34
SCU_CM.006	Deep Sleep entry with PLL power-down option generates SOSCWDGT and SVCOLCKT trap	–	35
SCU_CM.021	Registering of service requests in SRRAW register can fail	–	35

1 Overview

Functional deviation	Short description	Chg	Pg
SDMMC_CM.001	Unexpected interrupts after execution of CMD13 during bus test	–	36
SDMMC_CM.002	Unexpected Tx complete interrupt during R1b response	–	36
SDMMC_CM.004	Busy response from card in write resume operation not detected	–	37
SDMMC_CM.005	Controller sends other command when Auto CMD12 enabled	–	37
SDMMC_CM.006	Stream write issue due to wrong FIFO handling causes data corruption in eMMC mode	–	38
SDMMC_CM.007	Consecutive write to the same register in SD clock domain	–	38
SDMMC_CM.008	Receive state machine hangs if driver programs stop at block gap request using CMD18 when receive buffers are full	–	39
SDMMC_CM.009	Latching current value in the response register	–	39
STARTUP_CM.002	Bootstrap loader may fail for devices with reduced PSRAM size	–	39
STARTUP_CM.003	Unique CHIP ID copied to DSRAM1 by SSW might be incorrect	New	40
USB_CM.002	GAHBCFG.GlblIntrMsk not cleared with a software reset	–	40
USB_CM.003	Endpoint NAK not sent in Device Class applications with multiple endpoints enabled	–	40
USB_CM.005	DMA support for USB host mode operation	–	41
USIC_AI.008	SSC delay compensation feature cannot be used	–	41
USIC_AI.014	No serial transfer possible while running capture mode timer	–	42
USIC_AI.017	Clock phase of data shift in SSC slave cannot be changed	–	42
USIC_AI.018	Clearing PSR.MSLS bit immediately deasserts the SELOx output signal	–	42
USIC_AI.020	Handling unused DOUT lines in multi-IO SSC mode	–	43
USIC_AI.021	In I2C master mode a falling edge on SCL is wrongly regarded as a valid START condition	New	43
WDT_CM.001	No overflow is generated for WUB default value	–	44

Table 5 Application hints

Hint	Short description	Chg	Pg
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion	–	45
ADC_AI.H004	Completion of Startup Calibration	–	46
ADC_AI.H008	Injected conversion with broken wire detection	–	46
ADC_TC.H011	Bit DCMSB in register GLOBCFG	–	47
ETH_AI.H001	Sequence for Switching between MII and RMII Modes	–	47
ECAT_CM.H001	LINK_MII can only externally be tied to the logic value high if ECAT_MII interface shall not be used	–	47
ECAT_CM.H002	Sync0/Sync1 events must not be enabled via the ESC AL Event Mask register in SSC V5.13 or later	New	47
MultiCAN_AI.H005	TxD Pulse upon short disable request	–	48
MultiCAN_AI.H006	Time stamp influenced by resynchronization	–	48
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off	–	49
MultiCAN_AI.H008	Effect of CANDIS on SUSACK	–	49
MultiCAN_AI.H009	Behavior of MSGVAL for Remote Frames in Single Data Transfer Mode - Documentation Update	New	49
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode	–	50
MultiCAN_TC.H004	Double remote request	–	50
PORTS_CM.H001	RTC_XTAL pins are swapped	–	51
PORTS_CM.H002	Class A2 pins GPIO driver strength configuration	–	51
RESET_CM.H001	Power-on reset release	–	52
USIC_AI.H004	I2C slave transmitter recovery from deadlock situation	–	52

Table 6 Documentation updates

Hint	Short description	Chg	Pg
WDT_CM.D001	Correction to section "Pre-warning mode"	–	53
MultiCAN_CM.D001	MultiCAN chapter Table “Minimum Operating Frequencies”	–	53
MPU_CM.D001	No restrictions on using Bit5 to Bit8 of register MPU_RBAR	–	54
STARTUP_CM.D003	Alignment of ABM/PSRAM Header	–	54

2 Functional deviations

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

2.1 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=1B)
- The Wait-for-Read mode for GLOBRES is enabled (GLOBRCR.WFR=1B), each conversion of the auto-scan sequence must 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.

2.2 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.

2.3 ADC_CM.001

Conversion results can be wrong if groups are not synchronized

The VADC module clock f_{ADC} is internally divided by each of the converter groups Gx separately to $f_{ADCI(Gx)}$. The division factor is programmable by bit field DIVA of register GLOBCFG: $f_{ADCI} = f_{ADC}$ / (DIVA+1) valid for DIVA >= 1.

Due to this architecture, the individual prescalers may be displaced by one module clock as shown in Figure 1 for $G0_2$ and $G1_1$. In this example the division factor is 5 (DIVA=4).

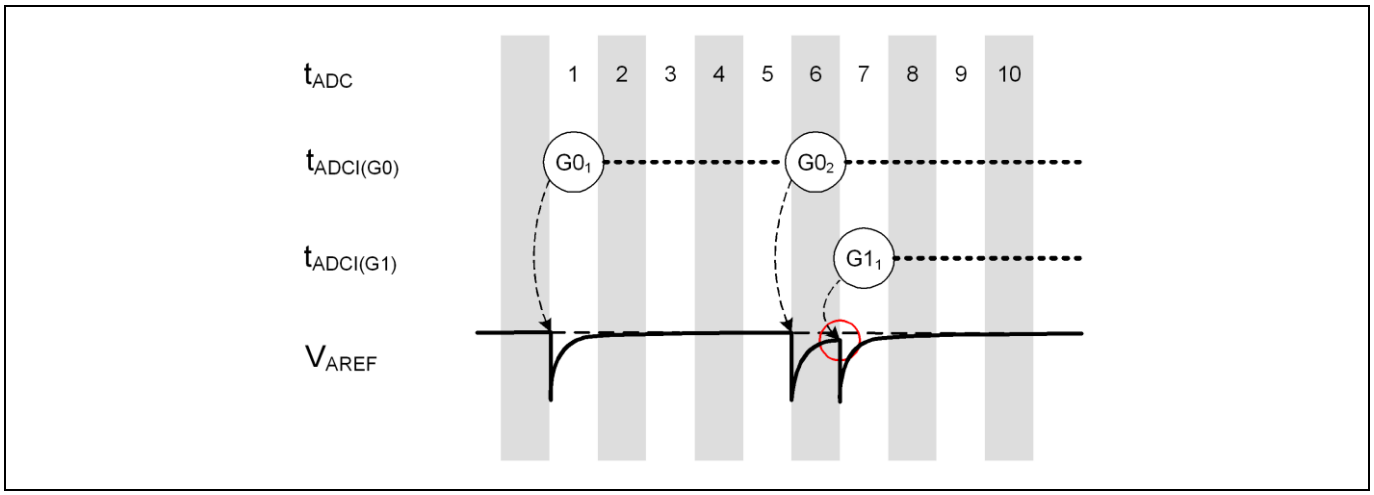

Figure 1 Influence of concurrent conversions on reference voltage

In cycle 1 only the G0 conversion step is started. At start time the reference voltage is strobed and a certain amount of electric charge (see data sheet parameter Q_{CONV}) is consumed. This causes a drop of the reference voltage level V_{AREF}. Quantity of voltage drop depends on analog voltage level to convert.

In cycle 6 another G0 conversion step is starting and in cycle 7 a G1 conversion step. In this case the reference voltage may not yet be fully recovered when the G1 conversion step strobes V_{REF}. Consequently, the G1 conversion step yields an incorrect value.

In repeated measurements using the example setup errors up to 25 LSB have been observed in 10-16 of 4096 conversions. Magnitude of the error however depends on the system configuration and load. The workaround therefore should be applied for all applications.

Implications

If two or more converter groups (G0-G3) are started (initialized) asynchronously and multiple conversion steps or sample phases occur in the same time frame then the reference voltage settling time can be violated, which leads to incorrect conversion results.

Workaround

The workaround makes sure that all ADC group clocks $f_{ADCI(Gx)}$ are started concurrently. Even if your application does not require a master/slave configuration this sequence ensures a synchronized start behavior.

This example is assuming that 4 ADC groups are available and used. G0 will be used as master G1-G3 as slaves. If less groups are used or available, it can simply be reduced.

1. Disable all ADC groups (x=0-3) by clearing ASENy (y=0-2) bits:

```
GxARBPR.ASENy = 0
```

2. Configure the queue, scan and background sources as required
3. Configure one group as master, others as slaves and set EVALRy (y=1-3):

```
GxSYNCTR.EVALRy = 1 G0SYNCTR.STSEL = 00 -- G0 is master 0 G1SYNCTR.STSEL
= 01 -- G1 is slave of master 0 G2SYNCTR.STSEL = 01 -- G2 is slave of
master 0 G3SYNCTR.STSEL = 01 -- G3 is slave of master 0
```

4. Configure the slave groups G1, G2, G3

```
GxARBCFG.ANONC = 00 -- Converter off
-- Remaining GxARBCFG bits are set as required
```

5. Configure the master G0

```
G0ARBCFG.ANONC = 11 -- Converter on
-- Remaining GxARBCFG bits are set as required
```

6. If dual masters are required by the application, then the sequence must be extended. This example changes G2 to become the second master and G3 its slave.

```
-- Change configuration of new master G3 G3ARBCFG.ANONC = 11 -- Converter
on G2SYNCTR.STSEL = 00 -- make G2 master 1 G3SYNCTR.STSEL = 10 -- make G3
slave of master 1
```

2.4 ADC_CM.002

Converter diagnostics not functional

The analog converter diagnostics feature of the VADC to test the proper operation is not functional.

Implications

All diagnostic pull devices remain disconnected, also if the converter diagnostics feature is enabled.

No portions of VAref can be selected for diagnostic purpose.

Workaround

None

2.5 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 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 post calibration on the calibrated converters Gx (x = 0..3). Instead, use a dedicated group for fast-compare operations.

2.6 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 2.

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.

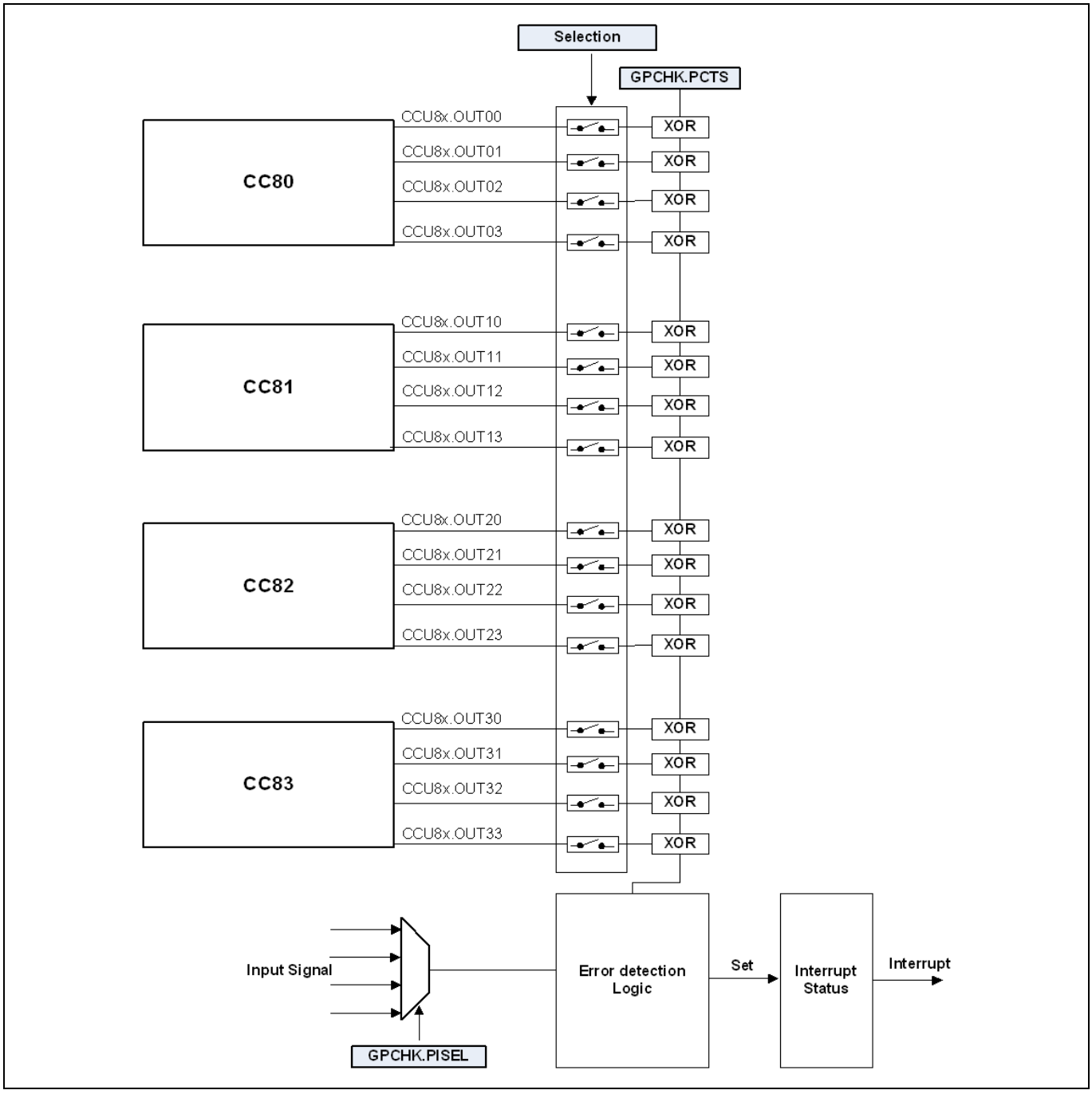

Figure 2 Parity Checker diagram

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 the origin of the interrupt was - because there is only one source.

2.7 CCU8_AI.006

Timer concatenation does not work when using external count signal

Each CCU8 peripheral contains four 16-bit timers. It is possible nevertheless to concatenate multiple timers to achieve a timer/counter with 32, 48 or 64 bits. To enable the concatenation feature, the CC8yCMC.TCE bitfield needs to be set to 1_B – Figure 3 a), where CCU8x represents a CCU8 peripheral instance x, and CC80 and CC81, represents timer 0 and timer 1 respectively (please notice that CC80 and CC81 are just used for simplicity, meaning that this function can be used also with the other timers inside CCU8x).

It is also possible to use an external signal as a count trigger. This means that when using an external count signal, the LSB timer is incremented each time that a transition on this external signal occurs - Figure 3 b).

When an external count signal is enabled - by programming the CC8yCMC.CNTS with 01_B, 10_B or 11_B - the concatenation function does not work. One cannot use in parallel the timer concatenation and external count signal features.

Note: On Figure 3, the count signal is used in CCU80 because this timer represents the LSBs. While the count signal could be enabled in the MSB timer (CC81), this does not make sense when the timers are concatenated - because the count should be used to increment the LSB timer. The LSB timer will then be in each wrap around, increment the MSB timer.

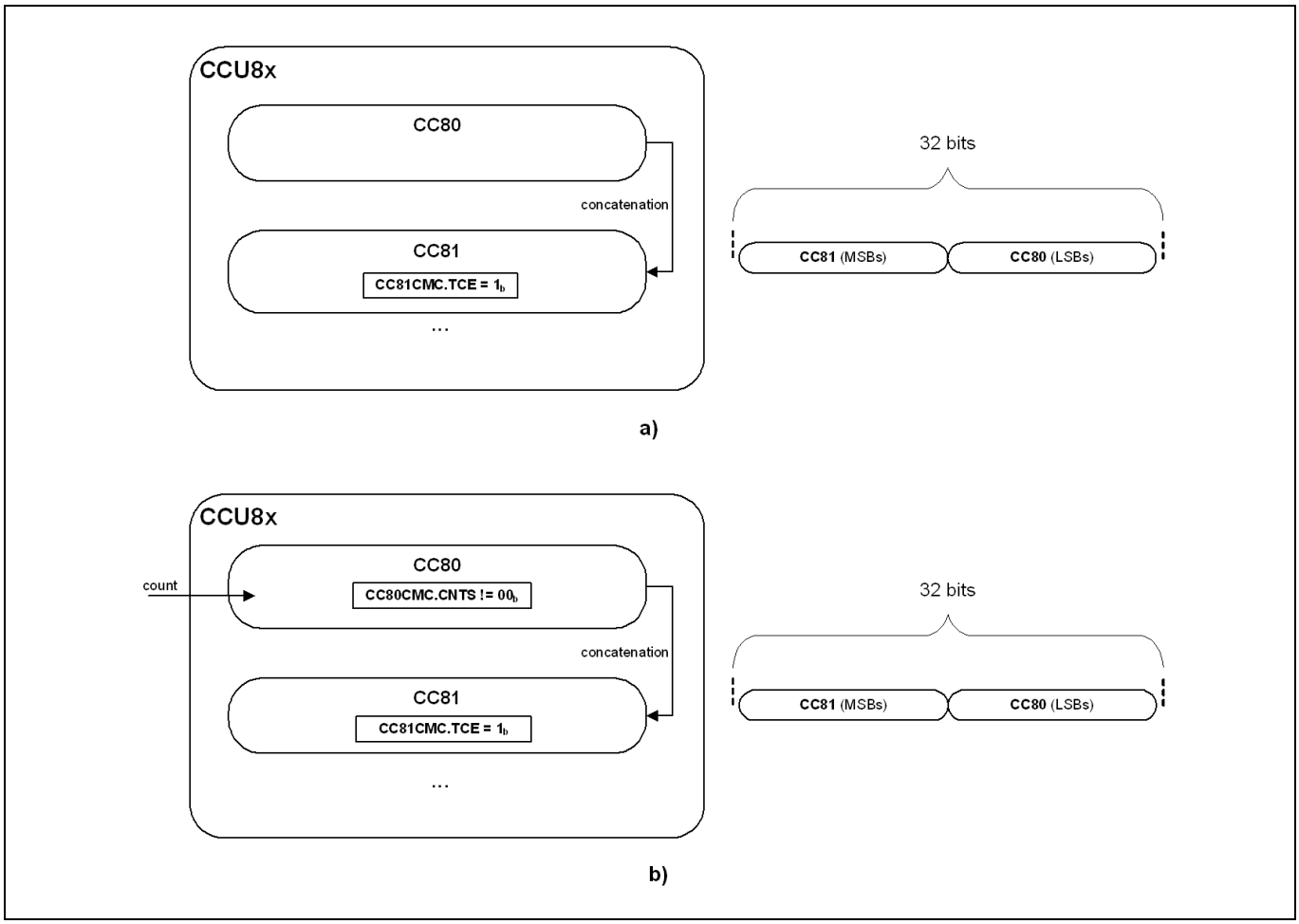

Figure 3 CCU8x concatenation feature resource configuration - a) without external count function; b) with external count function

Workaround

None

2.8 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 of Figure 4. 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 the clock is 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 it 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 is 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.

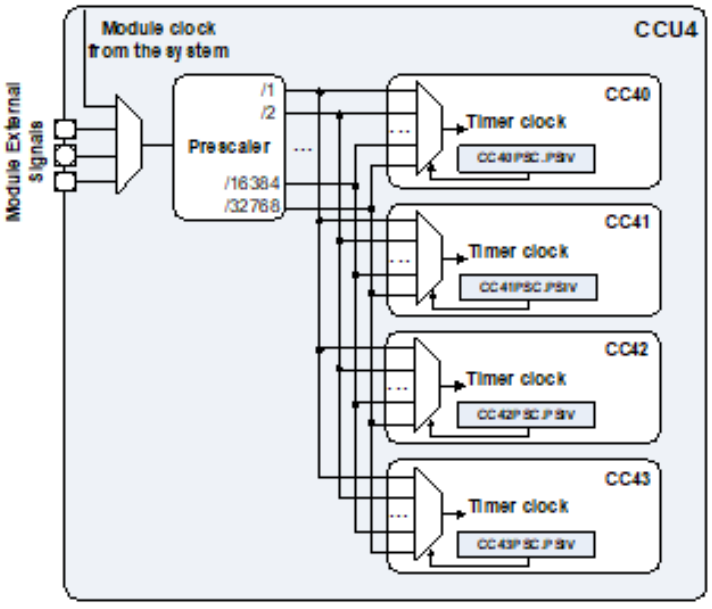

Figure 4 Clock selection diagram for CCU4

Workaround

None

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

2.10 CCU_AI.008

Clock ratio limitation when using MCSS inputs

The MCSS input signals of CCU8 and CCU4 units are erroneously sampled with the AHB bus clock f_{PERIPH} instead of the module clock f_{CCU}.

Implication

If the f_{PERIPH} and f_{CCU} frequencies are programmed to a ratio different from 1:1 then the MCSS signals running from POSIF to CCU4/CCU8 are not correctly sampled by CCU8/CCU4.

This can for example affect brushless DC motor drive applications when a clock ratio different from 1:1 is required.

Workaround

None

2.11 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

An LDR is executed, with SP/R13 as the destination

1. The address for the LDR is successfully issued to the memory system
2. 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 having 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.

2.12 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 contexts 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 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

2.13 CPU_CM.005

Store immediate overlapping exception return operation might vector to incorrect interrupt

The Arm® Cortex®-M4 includes a write buffer that permits execution to continue while a store is waiting on the bus. Under specific timing conditions, during an exception return while this buffer is still in use by a store instruction, a late change in selection of the next interrupt to be taken might result in there being a mismatch between the interrupt acknowledged by the interrupt controller and the vector fetched by the processor.

This erratum only affects systems where writeable memory locations can exhibit more than one wait state. For the XMC4000 Family only devices with external memory controller (EBC) used in Application are affected. All internal memory use zero wait state access.

Implications

The processor should execute interrupt handler C, and on completion of handler C should execute the handler for B. If the conditions above are met, then this erratum results in the processor erroneously clearing the pending state of interrupt C and then executing the handler for B twice. The first time the handler for B is executed it will be at interrupt C's priority level. If interrupt C is pended by a level-based interrupt which is cleared by C's handler, then interrupt C will be pended again once the handler for B has completed and the handler for C will be executed. If interrupt C is level based, then this interrupt will eventually become re-pending and subsequently be handled. If interrupt C is a single pulse interrupt, then there is a possibility that this interrupt will be lost.

Workaround

For software not using the memory protection unit, this erratum can be worked around by setting DISDEFWBUF in the Auxiliary Control Register.

In all other cases, the erratum can be avoided by ensuring a DSB occurs between the store and the BX instruction. For exception handlers written in C, this can be achieved by inserting the appropriate set of intrinsics or inline assembly just before the end of the interrupt function, for example:

```
ARMCC:
...
__schedule_barrier();
__asm{DSB};
__schedule_barrier();
}
GCC:
...
__asm volatile ("dsb 0xf":::"memory");
}
```

2.14 DAC_CM.003

FIFO usage limitation in “Data Processing Mode”

The reference manual describes in section X.2.1.2 of the DAC chapter that the FIFO “...is introduced to allow a longer request latency...”.

“Data Processing mode” is the only operation mode based on FIFO usage. For this mode it was intended that a service request is raised only if the FIFO runs into the empty state after a DAC trigger occured.

In fact, service request(s) occur after each DAC trigger. Additionally, some service requests can be delayed. Due to this misbehavior a reliable refill mechanism cannot be implemented.

Implications

Unexpectedly delayed and superfluous service requests from the DAC FIFO inhibit the implementation of useful refill mechanisms based on interrupt service routines or GPDMA service.

Workaround

None

2.15 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 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 must 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

2.16 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

2.17 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 of $7F_H$ (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.

2.18 EBU_CM.001

32-bit memory with byte write capability needs address ranges 0 and 1 enabled

The external bus unit (EBU) should enable byte control signals BC2 and BC3 if 32-bit memory with byte write capability is programmed.

In the affected device BC2 and BC3 are enabled only if address range CS1 respectively CS0 are activated additionally.

Implications

Both address ranges CS0 and CS1 must be enabled to operate 32-bit memory with byte write capability.

Workaround

Activate both CS0 and CS1 if 32-bit memory with byte write capability needs to be connected:

- `ADDRSEL0.REGENAB = 1`
- `ADDRSEL1.REGENAB = 1`

32-bit multiplexed memory with byte write capability is programmed if:

- `BUSWCONx.AGEN = 0`
- `BUSWCONx.PORTW = 11`

Note:

1. *In de-multiplexed mode 32-bit access is not supported*
2. *If CS0 and/or CS1 address range are not used for the 32-bit memory with byte write capability, then it can be used for another EBU application*
3. *If CS0 and/or CS1 address range is not used for the 32-bit memory with byte write capability (or other EBU application) then the pin related to the unused range/s can be used for other purposes. Please program required value using P3_HWSEL.HW2 for CS0 or P0_HWSEL.HW9 for CS1*

2.19 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 header (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 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 in 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.

2.20 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.

2.21 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.

2.22 FCE_CM.001

Result value is wrong if read directly after last write

If a result register RESm is read directly after the last write of input data to the corresponding IRm register, then the calculated result is wrong.

Workaround

Insert a wait cycle between last write and result read.

This can be accomplished by:

- reading the result twice or
- inserting a NOP instruction between last write and result read

2.23 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 5).

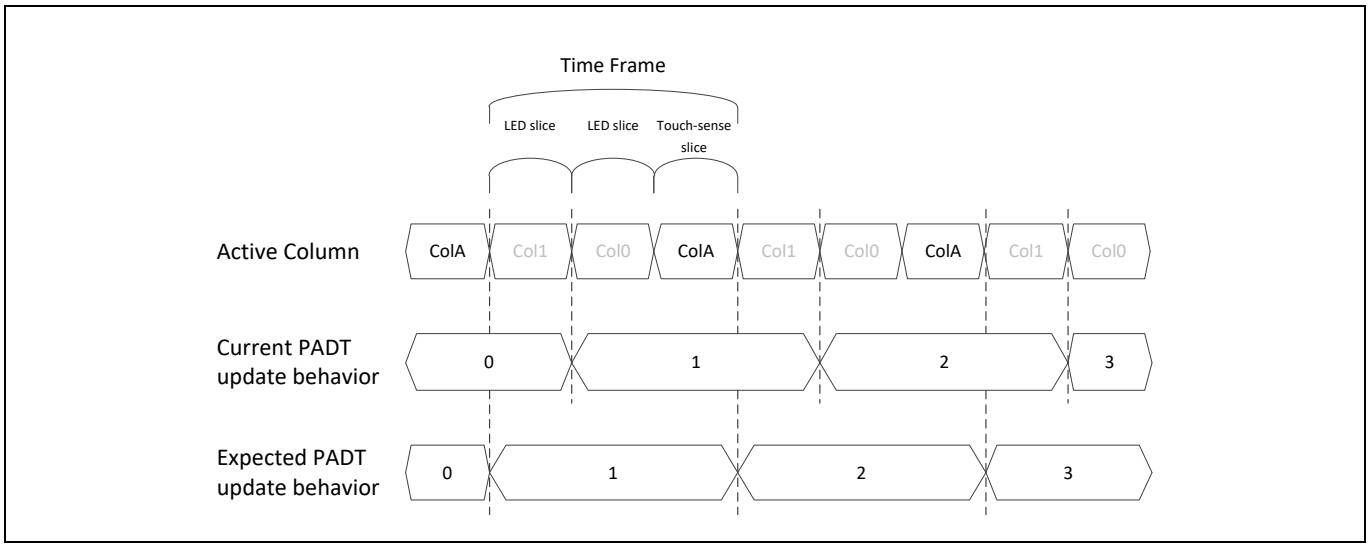

Figure 5 PADT update behavior

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 6)In addition to this, when no LEDs are enabled, pad turn 0 will prevail for one time slice longer before it gets updated (Figure 7).

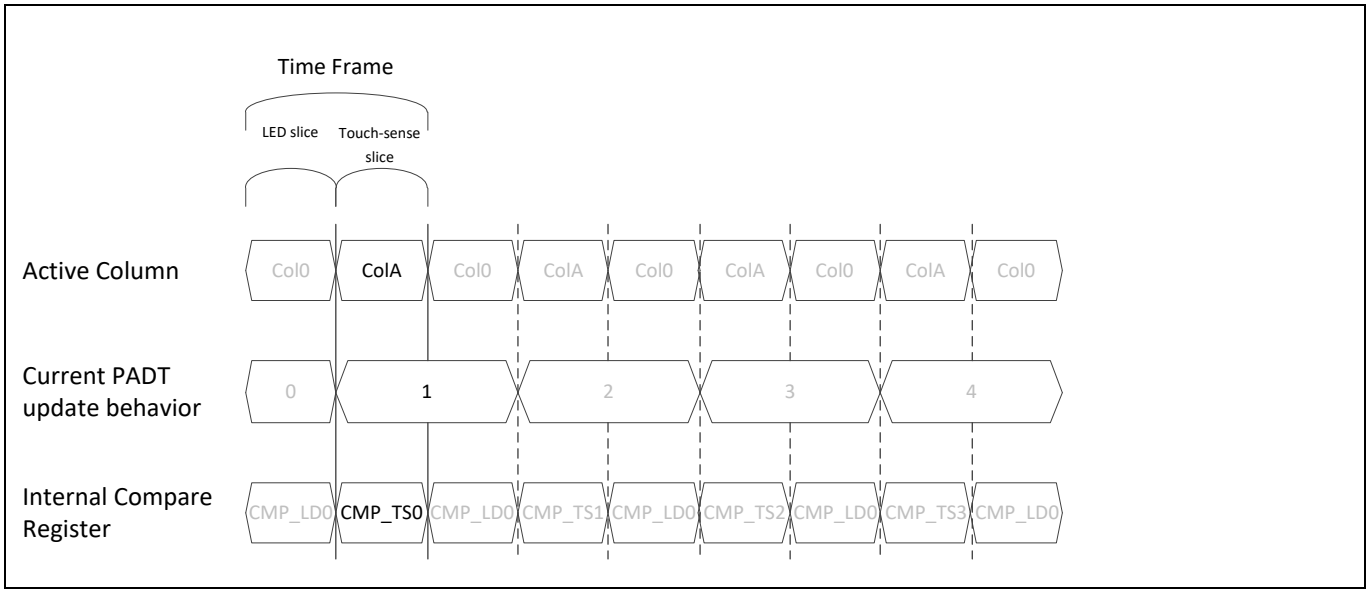

Figure 6 Effect of delay on the update of Internal Compare register with 1 LED column enabled

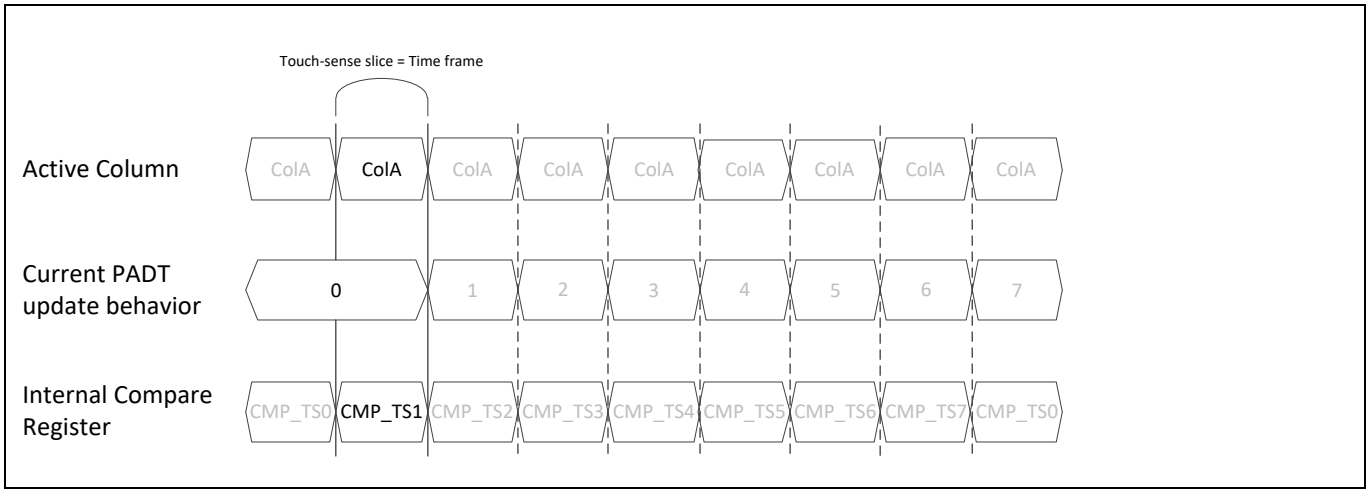

Figure 7 Pad turn 0 prevails for one time slice longer when no LEDs are enabled

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 8).

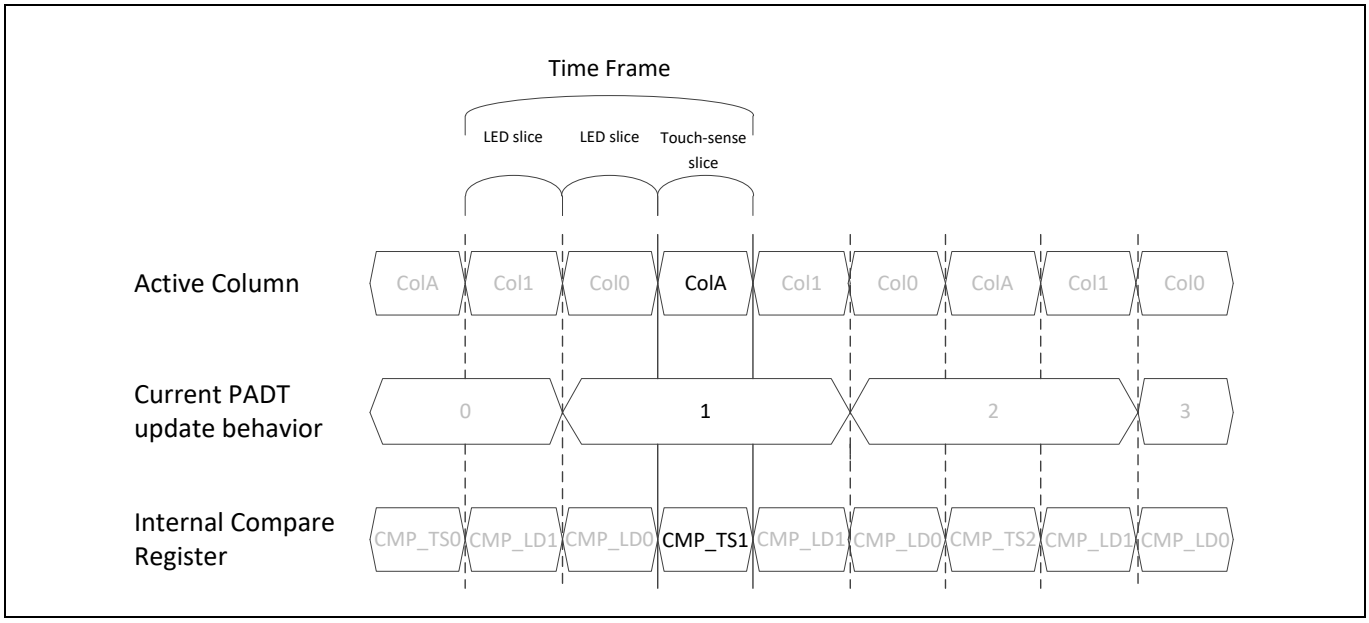

Figure 8 Internal Compare register updated with correct compare register value with 2 LED columns enabled

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 into 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 (Figure 8). 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	101B to 000B	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 9)

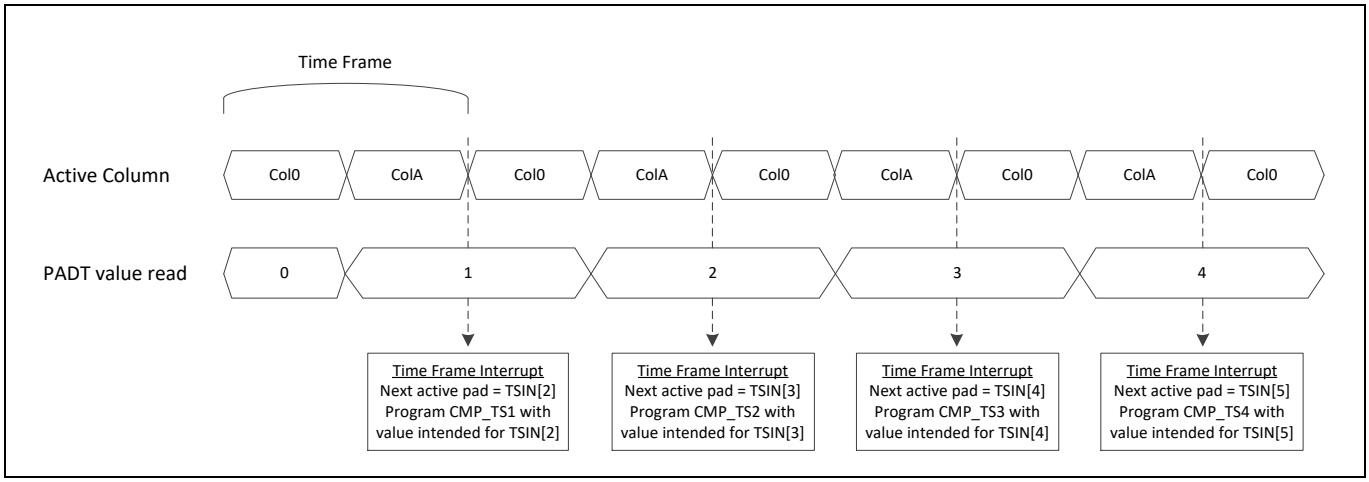

Figure 9 Software workaround demonstration

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].

2.24 PARITY_CM.001

Parity error signaling deviation in immediate write/read sequences

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

Due to buffering of AHB write requests, immediate write/read sequences on the same 32-bit aligned memory cell can produce two kinds of parity error signaling deviation:

- **Case A:** Parity error signaling by AHB bus error may be suppressed
- **Case B:** A parity error may be signaled by AHB bus error although the just-written data would have stored correct parity

Both cases share the same root cause and the same trigger conditions (immediate write followed by read to the same 32-bit aligned word); they differ only in the size relation between the write and the read and in the resulting symptom.

2.24.1 Case A: Suppressed signaling

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
- if another byte or half-word (D2) is written to the same word
- 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. 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 @2000_2303H and if the byte @2000_2302H (same 32-bit word) is written and immediately after this the word @2000_2300H (or half-word @2000_2302H) is read, then no parity error by AHB bus error is signaled.

2.24.2 Case B: Spurious signaling

If a memory cell holds invalid parity for any reason (for example the cell was never written since power-on reset, a prior single-event upset, or deliberate injection via PMTPR/MCHKCON), an immediate write followed by a read of the same address and the same size can raise a parity BusFault even though the write should have stored correct parity.

The read races the still buffered write and observes the old cell contents, including old data and the old (invalid) parity, before the write commits to the SRAM array. Consequently, the parity checker evaluates the stale contents and reports a parity mismatch.

Following types of access sequences are affected.

Table 9 Affected write/read sequences (Case B)

write	read	condition
byte	byte	same address, same 32-bit word
half-word	half-word	same address, same 32-bit word
word	word	same address

Example:

If a parity error is present in the byte @2000_0000H (for example because the cell has not been written since reset) and a byte is written to @2000_0000H, and immediately after this the same byte @2000_0000H is read, a parity BusFault can be signaled although the just-written data would have stored correct parity.

2.24.3 Scope

Case B can only expose parity errors that already exist in memory. It cannot generate parity errors on memory cells that hold valid data and valid parity. On correctly initialized parity-protected memory the effect does not occur.

The functional problem in both cases 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.

2.24.4 Workaround

2.24.4.1 For case A

If the described usage scenario occurs in the application, then it is recommended to enable the parity error trap for the used PSRAM and/or DSRAM units. By this, an 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.

2.24.4.2 For case B

The following measures individually or combined to prevent the effect:

- Initialize all parity-protected SRAM (including stack and unused areas) to a defined value before enabling parity checking (PEEN). This ensures that no cell holds invalid parity, eliminating the error source entirely
- If an immediate write/read to the same cell is unavoidable on possibly uninitialized memory, insert a DSB instruction between the write and the read. The DSB drains the AHB write buffer so that the new data and its correct parity are visible before the read executes
- Enabling the parity error trap (PETE) as recommended for Case A provides an additional, independent detection path

2.25 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: $f_{CPU} \geq f_{SDMMC}$+25%. For example, if SDMMC shall operate with f_{SDMMC} at 48 MHz then f_{CPU} must be set for 60 MHz or higher

2.26 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 bits and later reads P15_PDISC will not see the expected read value but always reads 0_B for P15_PDISC.[4,5].

Workaround

None

2.27 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 10 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	$2222\ 2222_H$	$0000\ 0000_H$

[1]Only in XMC4500 devices.
[2]Device and package dependent

Register	Sytem reset value	Module reset value
Pn_PDR1	2222 2222_H	0000 0000_H

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 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

2.28 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 cannot be used. To use the ETM Trace please select a other set of available pins for that interface.

Port 7 provides additional redundant connections 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

2.29 PORTS_CM.007

P14 and P15 cannot be used in boundary scan test

P14 and P15 are analog ports with selectable digital input functionality. After reset the digital input functionality is disabled. Due to this the input value present at related pins is not visible inside the device.

Implications

The digital logic values present at package pins related to P14 and P15 cannot be captured in IEEE 1149.1 boundary scan test.

Workaround

None

2.30 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 10.

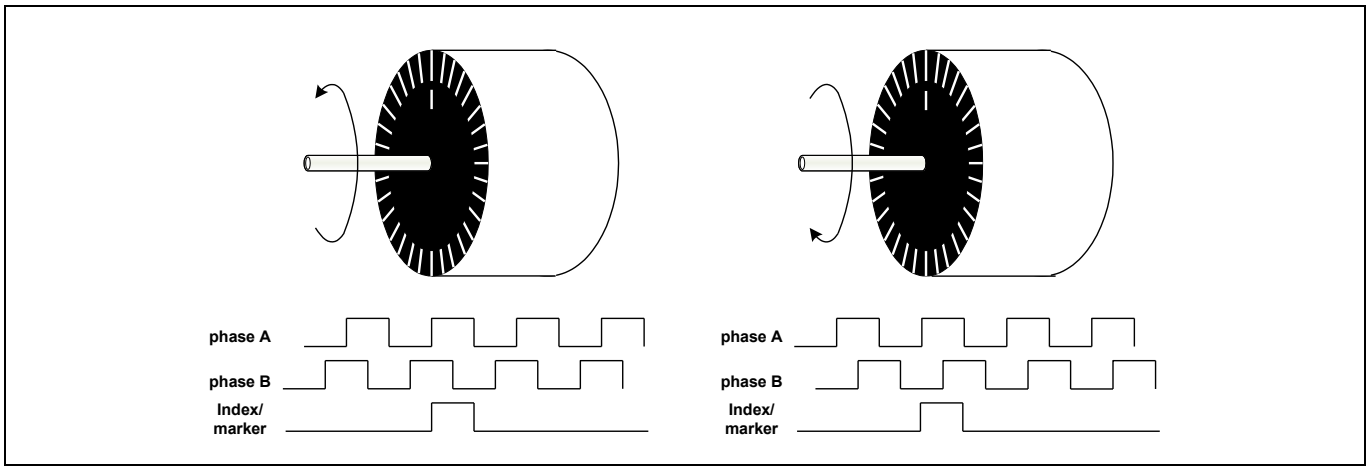

Figure 10 Rotary encoder outputs - Phase A, Phase B and Index

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 11 notices 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 types of signals (index with duration below 1/2 of the tick period).

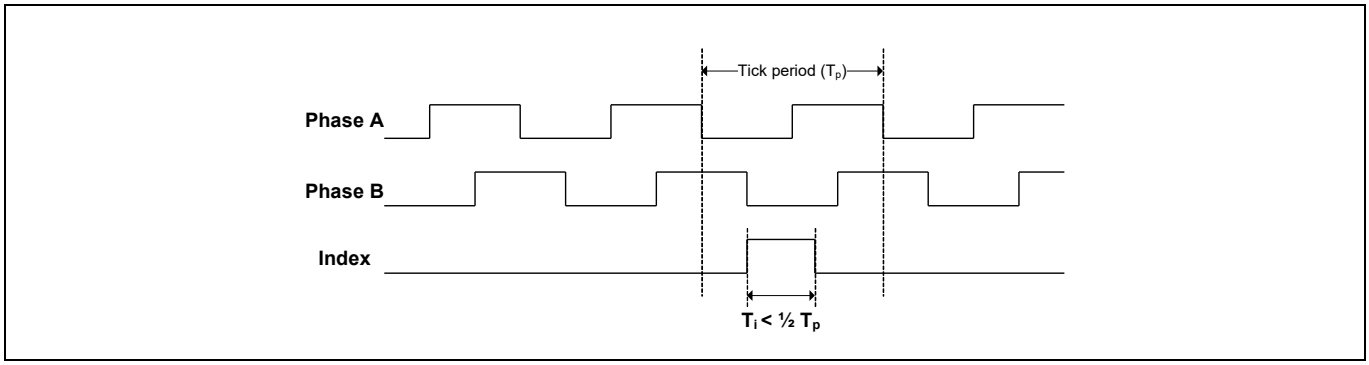

Figure 11 Different index signal types

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 12.

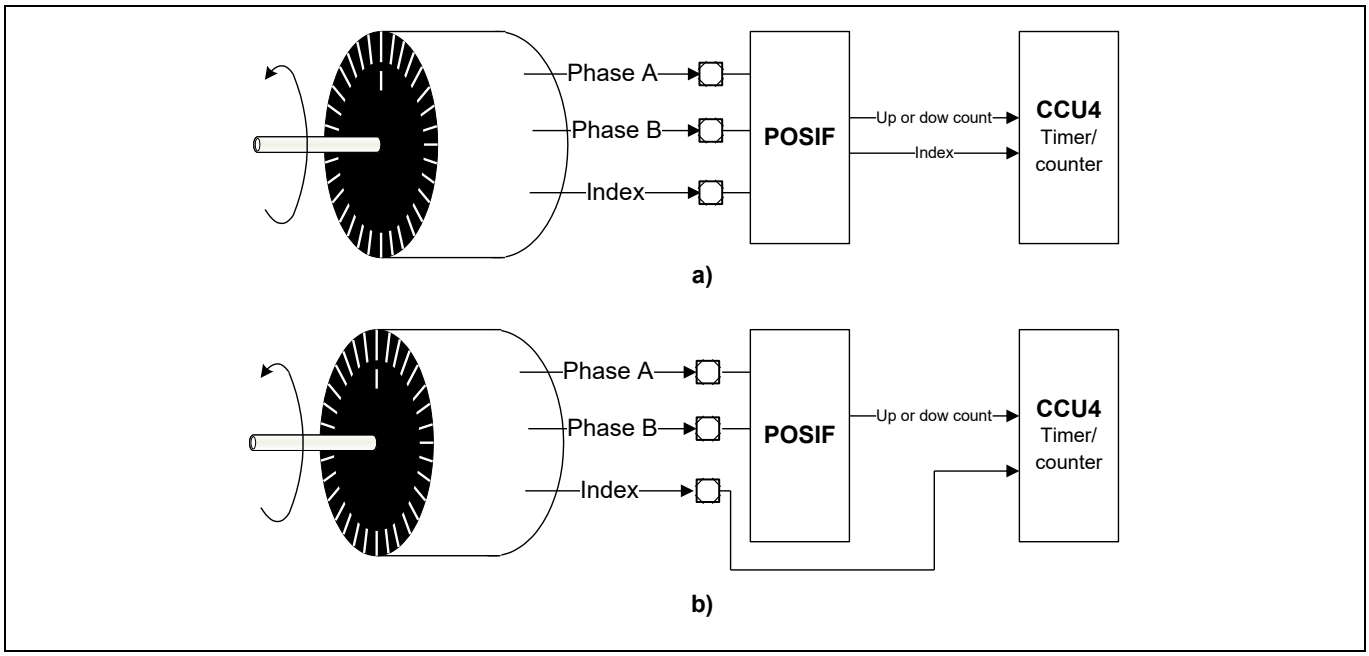

Figure 12 Index usage workaround - a) Non working solution; b) Working solution

2.31 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 of debugging of hibernate mode control software.

Workaround

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

2.32 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.

2.33 SCU_CM.021

Registering 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 11) 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 11. 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 11 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
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

2.34 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. Consequently, unexpected interrupts may be generated.

Workaround

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

2.35 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 12 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 13 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.

2.36 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) to free the bus for a higher priority transfer to a different function or memory. The host controller 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

2.37 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.

2.38 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

2.39 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}).

1. BLOCK_SIZE
2. BLOCK_COUNT
3. ARGUMENT1
4. TRANSFER_MODE
5. COMMAND
6. HOST_CTRL
7. 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.

2.40 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

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

2.41 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 into 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 $f_{SDMMC}/2$.

2.42 STARTUP_CM.002

Bootstrap loader may fail for devices with reduced PSRAM size

The internal UART and CAN bootstrap loader always copies the code to the beginning of the physical PSRAM. The Data Sheet defines derivatives with reduced PSRAM size. For such devices the physical start address is in an address range that is defined as “reserved”.

Devices affected

All XMC4800 derivatives with 200 Kbytes SRAM (72 Kbytes PSRAM).

Implications

In some cases, the “bootstrap loaded” code, usually a flash loader used to load/update flash images, compiled and linked for such a target device may fail, especially if it references to code or data inside this code with absolute addresses. This is because the uploaded code is copied to a different address range than intended/defined by the compiler/locator.

Workaround

Compile and link the “bootstrap loaded” code/data for a target device with the full physical PSRAM size, by using the linker script or project settings of the used compiler tool.

For XMC4800 derivatives compile for a XMC4800 with 96 Kbytes PSRAM.

2.43 STARTUP_CM.003

Unique CHIP ID copied to DSRAM1 by SSW might be incorrect

Inside the section “Initial Boot Sequence” in chapter “Startup Modes” it is described, how startup software reads staging information from flash configuration sector and writes this information to the beginning of DSRAM1. This information also includes a “Unique Chip ID”, a 16-byte value, which is defined to be absolute unique for every XMC4000.

After first power on, the first four bytes of the 16 byte “Unique Chip ID” might contain incorrect non-deterministic data.

Workaround

If your application is using the “Unique Chip ID” from DSRAM1, apply a software system reset for every startup before proceeding with your application. After software system reset also the first four bytes of the 16 byte “Unique Chip ID” contain the correct data.

2.44 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.

2.45 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.

2.46 USB_CM.005

DMA support for USB host mode operation

USB host core can be operated in two data exchange modes:

- Direct Memory Access based “DMA mode”
- CPU supported “Slave mode”

In DMA mode the USB core is supposed to move data between USB bus and internal memory without CPU support. Only after a completed data transfer the CPU is notified.

In the erroneous device DMA transfers between USB module and DSRAM1 or DSRAM2 are frequently disturbed by AHB errors.

Implications

Due to frequent AHB errors DMA based data transfer mode is inefficient and cannot be used reliably.

Workaround

Operate the USB host core in “Slave mode”.

Please refer to the programming sequences described in section "Host Programming in Slave mode" of the reference manual for details.

2.47 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

2.48 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

2.49 USIC_AI.017

Clock phase of data shift in SSC slave cannot be changed

Setting PCR.SLPHSEL bit to 1 in SSC slave mode is intended to change the clock phase of the data shift such that reception of data bits is done on the leading SCLKIN clock edge and transmission on the other (trailing) edge.

However, in the current implementation, the feature is not working.

Workaround

None

2.50 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

2.51 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.

2.52 USIC_AI.021

In I2C master mode a falling edge on SCL is wrongly regarded as a valid START condition

The I2C-bus specification Rev.6 – 4 April 2014 (and earlier versions) specifies on page 9, section “3.1.4 START and STOP conditions”:

All transactions begins with a START (S) and are terminated by a STOP (P).

A HIGH to LOW transition on the SDA line while SCL is HIGH defines a START condition.

A LOW to HIGH transition on the SDA line while SCL is LOW defines a STOP condition.

In contrast to this, in I2C mode of the USIC peripheral, a falling edge on SCL is regarded as a valid START condition. This wrongly detected START-condition causes the USIC channel entering busy state. Consequently, further transmission attempts by the I2C master will not be forwarded onto the bus, until the busy state is exit. Busy state can only be exit by detecting a STOP condition on the respective USIC channel IOs.

Implications

If any device, connected to the I2C-bus, pulls the SCL to low, the USIC channel will detect a START condition and enter busy state. In busy state no further transmissions as a master are possible.

Workaround

Generate manually STOP condition, if USIC channel is in busy state:

Flush FIFOs if used:

1. Abort current transmission (see chapter “Non-Acknowledge and Error Conditions” of reference manual)
2. Check bus lines are released
3. Generate stop condition by toggling SCL and SDA IOs. (See to XMC4700/4800 reference manual - Chapter 18.5.1)
4. Clear all flags

2.53 WDT_CM.001

No overflow is generated for WUB default value

The Window Watchdog Timer (WDT) does not generate an overflow event if the default counter value $FFFFFFFF_H$ is used in register WUB.

Implications

Without a timer overflow no reset or pre-warning is requested. For other WUB values the WDT operates correctly and a reset or pre-warning is requested upon WDT overflow.

Workaround

Do not use $FFFFFFFF_H$ as counter value.

3 Application hints

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

3.1 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:

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

   Where,

   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 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:

$$NA \leq ND*(NAR+3)$$

3.2 ADC_AI.H004

Completion of Startup Calibration

Before using the VADC the startup calibration must be completed.

The calibration starts 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 while waiting for GxARBCFG.CAL = 1.

3.3 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 pre-charging 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).

3.4 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.

3.5 ETH_AI.H001

Sequence for Switching between MII and RMII Modes

When switching between MII and RMII modes is required, the ETH module must be in a defined state to avoid unpredictable behavior.

Therefore, it is recommended to use the defined sequence listed below:

1. Finish running transfers and make sure that transmitters and receivers are set to stopped state:
   a) Check the RS and TS status bits in ETH0_STATUS register
   b) Check that ETH0_DEBUG register content is equal to the reset value
2. Wait until a currently running interrupt is finished and globally disable interrupts
3. Apply and release reset to ETH0 module by writing to corresponding bit fields of PRSET2 and PRCLR2 registers
4. Initialize the new mode (MII or RMII)
5. Apply software reset by writing to ETH0_BUS_MODE.SWR bit

3.6 ECAT_CM.H001

LINK_MII can only externally be tied to the logic value high if ECAT_MII interface shall not be used

The reference manual states inside chapter 16.3.1 MII Interface:

“If an ESC MII interface is not used, input pins like LINK_MII has to be tied to the logic value high which indicates no link. RX_CLK, RXD, RX_ER, and especially RX_DV must be tied to GND. For this purpose, user can select a input position in register ECAT0_CONPx of these signals which are not connected to a pin on the XMC4[3/8]00.”

This statement can be misleadingly interpreted, that LINK_MII can internally be tied to the logic value HIGH.

The LINK_MII signal can only be tied to the logic value high, by connecting it to an external pin which is externally set to logic value HIGH.

In case only one of the two ports is used inside an application, LINK_MII of the unused interface must be tied externally to the logic value HIGH.

3.7 ECAT_CM.H002

Sync0/Sync1 events must not be enabled via the ESC AL Event Mask register in SSC V5.13 or later

The Beckhoff EtherCAT Slave stack code (SSC) v5.13 introduced inside *ecatslv.c*, function `StartInputHandler()`, a run-time block that reads the ESC PDI Configuration register (offset 0x0140) and, based on the SYNC0_MAPPED_TO_ALEVENT / SYNC1_MAPPED_TO_ALEVENT indication bits, sets the corresponding SYNC0_EVENT / SYNC1_EVENT bits in the local variable u16ALEventMask. This mask is subsequently written to the ESC AL Event Mask register (offset 0x0204) via `SetALEventMask()` during the PREOP → SAFEOP transition, thereby enabling Sync0 and Sync1 events on the AL Event path.

On XMC4800 the integrated EtherCAT Slave controller delivers the Distributed Clocks (DC) Sync0 and Sync1 events to the Arm® Cortex®-M4 core through dedicated internal signal lines connected to the interrupt controller (see the EtherCAT chapter of the XMC4700/XMC4800 reference manual).

Enabling Sync0/Sync1 additionally on the AL Event register while the same events are already delivered through the dedicated interrupt path can lead to duplicated ISR entries, unexpected AL Event flag states, and instability of the DC synchronization while the slave is in SAFEOP or OP.

Therefore the "Get Sync mapped to AL Event indication" block introduced in SSC V5.13 must be removed from `StartInputHandler()` in *ecatslv.c*. The block is delimited by ECATCHANGE_START (v5.13) ECAT1 / ECATCHANGE_END (v5.13) ECAT1 and is located immediately after the assignment i16WaitForPllRunningTimeout = 0; (line ~1152 in SSC v5.13). After removing the block, u16ALEventMask remains as configured by the surrounding SSC logic and only PROCESS_OUTPUT_EVENT / PROCESS_INPUT_EVENT (and other non-Sync events) are enabled on the AL Event Mask register. Sync0 and Sync1 events continue to be handled through the dedicated XMC ECAT Sync0/Sync1 interrupt path. The corresponding unified diff is delivered together with the Infineon SSC EtherCAT application as file *SSC_Infineon.patch*.

3.8 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 an 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.

3.9 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

3.10 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:

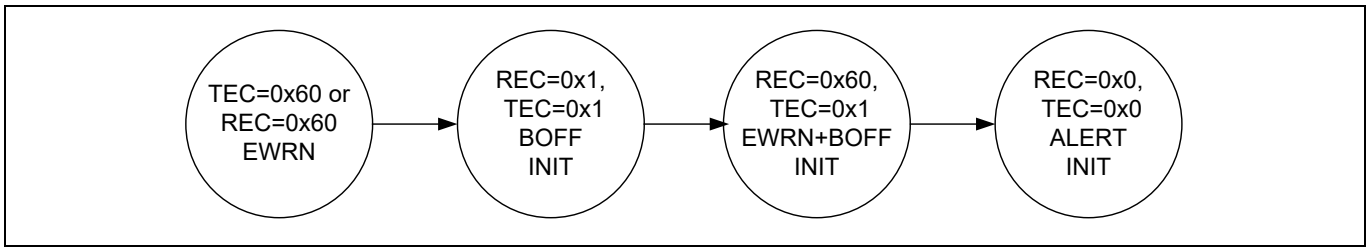

Figure 13 Alert Interrupt Behavior in case of Bus-Off

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.

3.11 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.

3.12 MultiCAN_AI.H009

Behavior of MSGVAL for Remote Frames in Single Data Transfer Mode - Documentation Update

In Single Data Transfer Mode (SDT = 1_B), bit MSGVAL is automatically cleared after transmission/reception of a Remote Frame.

The corresponding sections of MultiCAN sub-chapter “Single Data Transfer Mode” of the User’s Manual are copied below, with text updates marked in **bold**:

Message Reception

When a received message stored in a message object is overwritten by a new received message, the contents of the first message are lost and replaced with the contents of the new received message (indicated by MSGLST = 1_B).

If SDT is set (Single Data Transfer Mode activated), bit MSGVAL of the message object is automatically cleared by hardware after the storage of a received Data or **Remote Frame**. This prevents the reception of further messages.

Message Transmission

When a message object receives a series of multiple remote requests, it transmits several Data Frames in response to the remote requests. If the data within the message object has not been updated in the time between the transmissions, the same data can be sent more than once on the CAN bus.

In Single Data Transfer Mode (SDT = 1_B), this is avoided because MSGVAL is automatically cleared after the successful transmission of a Data or **Remote Frame**.

3.13 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 place.

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.

3.14 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.

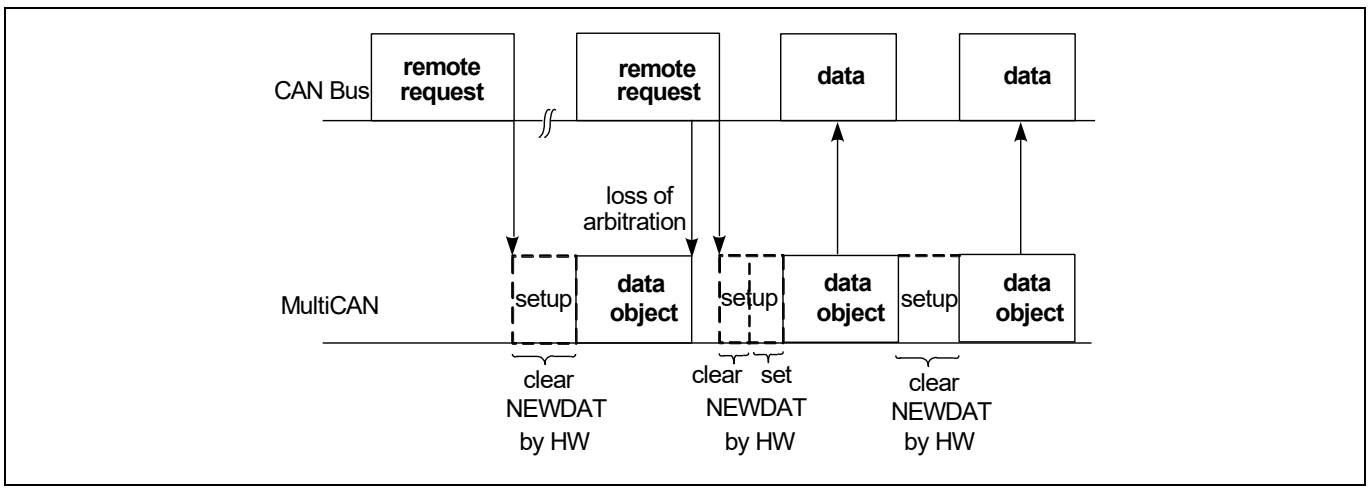

Figure 14 Loss of arbitration

3.15 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.

3.16 PORTS_CM.H002

Class A2 pins GPIO driver strength configuration

Before activating the push-pull driver, it is recommended to configure its driver strength and slew rate according to its pad class and the application needs using the Pad Driver mode register (Pn_PDR).

Selecting the appropriate driver strength allows optimizing the outputs for the needed interface performance, can help to reduce power consumption, and limits noise, crosstalk and electromagnetic emissions (EMI).

There are three classes of GPIO output drivers:

- Class A1 pads (low speed 3.3 V LVTTL outputs)
- Class A1+ pads (medium speed 3.3 V LVTTL outputs)
- Class A2 pads (high speed 3.3 V LVTTL outputs for example, for EBU or fast serial interfaces)

Class A1 pins provide the choice between medium and weak output drivers. Speed grade 6 MHz.

Class A1+ pins provide the choice between strong/medium/weak output drivers. For the strong driver, the signal transition edge can be additionally selected as soft or slow. Speed grade 25 MHz.

Class A2 pins provide the choice between strong/medium/weak output drivers. For the strong driver, the signal transition edge can be additionally selected as sharp/medium/soft. Speed grade 80 MHz.

If the output driver strength of Class A2 pins is configured as strong/sharp care need to be taken to avoid overshoots, undershoot and ringing that may lead to high radiated emissions and crosstalk.

The high radiated emissions may lead to Bus Errors exceptions (or Hard Fault exception in case the Bus Error exception is not enabled) caused by a double bit error fail in a flash read access. Flash double bits errors are identified in the `FLASH0.FSR` register.

Recommendation

In general, to avoid the high radiated emissions it is recommended the usage of damping resistors (10 Ω) between the high-speed drivers and the transmission lines.

It is also recommended to adapt the driver strength to the needs of the application, i.e. to drive a 25 MHz signal strong/medium or strong/soft would be suitable lowering the potential overshoots and undershoots.

3.17 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 PORST pin. If the PORST release requirement cannot be met due to external circuitry then spikes or toggling on the 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 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 PORST signal as specified in section “Power-Up and Supply Monitoring” of the datasheet. The recommended approach is to apply a pull-up resistor on the 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 PORST pin. The required pull-up resistor value may vary depending on the electrical parameters of the system influencing the signal edges of the PORST signal; for example, resistance and capacitance of the PCB and other components connected to the PORST pin.

3.18 USIC_AI.H004

I2C slave transmitter recovery from deadlock situation

While operating the USIC channel as an IIC slave transmitter, if the slave runs out of data to transmit before the master receiver issues clock pulses, for example due to an error in the application flow, it ties the SCL infinitely low.

Recommendation

To recover and reinitialize the USIC IIC slave from such a deadlock situation, the following software sequence can be used:

1. Switch the SCL and SDA port functions to be general port inputs for the slave to release the SCL and SDA lines:
   a) Write 0 to the two affected Pn_IOCRx.PCy bit fields
2. Flush the FIFO buffer:
   a) Write 1_B to both USICx_CHy_TRBSCR.FLUSHTB and FLUSHRB bits
3. Invalidate the internal transmit buffer TBUF:
   a) Write 10_B to USICx_CHy_FMR.MTDV
4. Clear all status bits and reinitialize the IIC USIC channel if necessary
5. Reprogram the Pn_IOCRx.PCy bit fields to select the SCL and SDA port functions again

At the end of this sequence, the IIC slave is ready to communicate with the IIC master again.

4 Documentation updates

The errata in this section contains updates to or completions of the user documentation. These updates are subject to be taken over into upcoming user documentation releases.

4.1 WDT_CM.D001

Correction to section "Pre-warning mode"

Section “Pre-warning mode” of WDT chapter in the reference manual states the following:

"… The alarm status is shown via register WDTSTS and can be cleared via register WDTCLR. A clear of the alarm status will bring the WDT back to normal state. The alarm signal is routed as request to the SCU, where it can be promoted to NMI. …"

Correction

The statement "A clear of the alarm status will bring the WDT back to normal state" is wrong.

A clear of the alarm status bit via write to WDTCLR.ALMC will clear only the bit WDSTSTS.ALMS.

To transfer the WDT back to the normal state a WDT service request is required.

4.2 MultiCAN_CM.D001

MultiCAN chapter Table “Minimum Operating Frequencies”

The table “Minimum Operating Frequencies” of the MultiCAN chapter shows the required minimum frequency of maximum five active CAN nodes at the same time. The XMC4700/XMC4800 provide six CAN nodes which can be used.

Correction

The Table 14 shows the minimum operating frequency extended to 6 CAN nodes.

Table 14 Minimum operating frequencies [1][MHz]

Number of allocated message objects MO[1,2,3,4,5,6]	Number of Active CAN nodes					
	1	2	3	4	5	6
16 MO	12	19	26	33	40	48
32 MO	15	23	30	37	44	51
64 MO	21	28	37	46	53	59
128 MO	40	45	50	55	61	69
256 MO	72	77	82	88	93	100

Notes:

1. In the case of 15-time quanta, the minimum operating frequency required is 15 MHz
2. In case of using CAN bootstrap loader, with one active node and two active message objects, the Mod_Name module needs minimum frequency of 20 MHz
3. In case of using CAN bootstrap loader, with one active node and two active message objects, the Mod_Name module needs minimum frequency of 20 MHz

4. In case of using CAN bootstrap loader, with one active node and two active message objects, the Mod_Name module needs minimum frequency of 20 MHz
5. Only those message objects must be taken into account that are allocated to a CAN node. The unallocated message objects have no influence on the minimum operating frequency
6. In case of using CAN bootstrap loader, with one active node and two active message objects, the Mod_Name module needs minimum frequency of 20 MHz

4.3 MPU_CM.D001

No restrictions on using Bit5 to Bit8 of register MPU_RBAR

The XMC4000 reference manuals describe, Bit5 to Bit8 of register MPU_RBAR are read-only and fixed to 0.

The Arm® documentation for the Arm® Cortex®-M4 processors specifies these bits are used to extend the LSBs to set the region base address and size.

Implications

The reference manual limits the range of possible base addresses and region sizes for memory protection.

Workaround

Use the Arm® documentation set for Arm® Cortex®-M4 processor for reference.

4.4 STARTUP_CM.D003

Alignment of ABM/PSRAM Header

The reference manual specifies PSRAM/ABM Header of 32-byte size. Only for 20byte of these 32 bytes the functionality is defined. The remaining 12 bytes are reserved. Inside the chapter for Startup modes of the reference manual only the functional bytes are specified but not the location of the reserved bytes.

The Figure 15 provides a detailed view on the location of the reserved bytes:

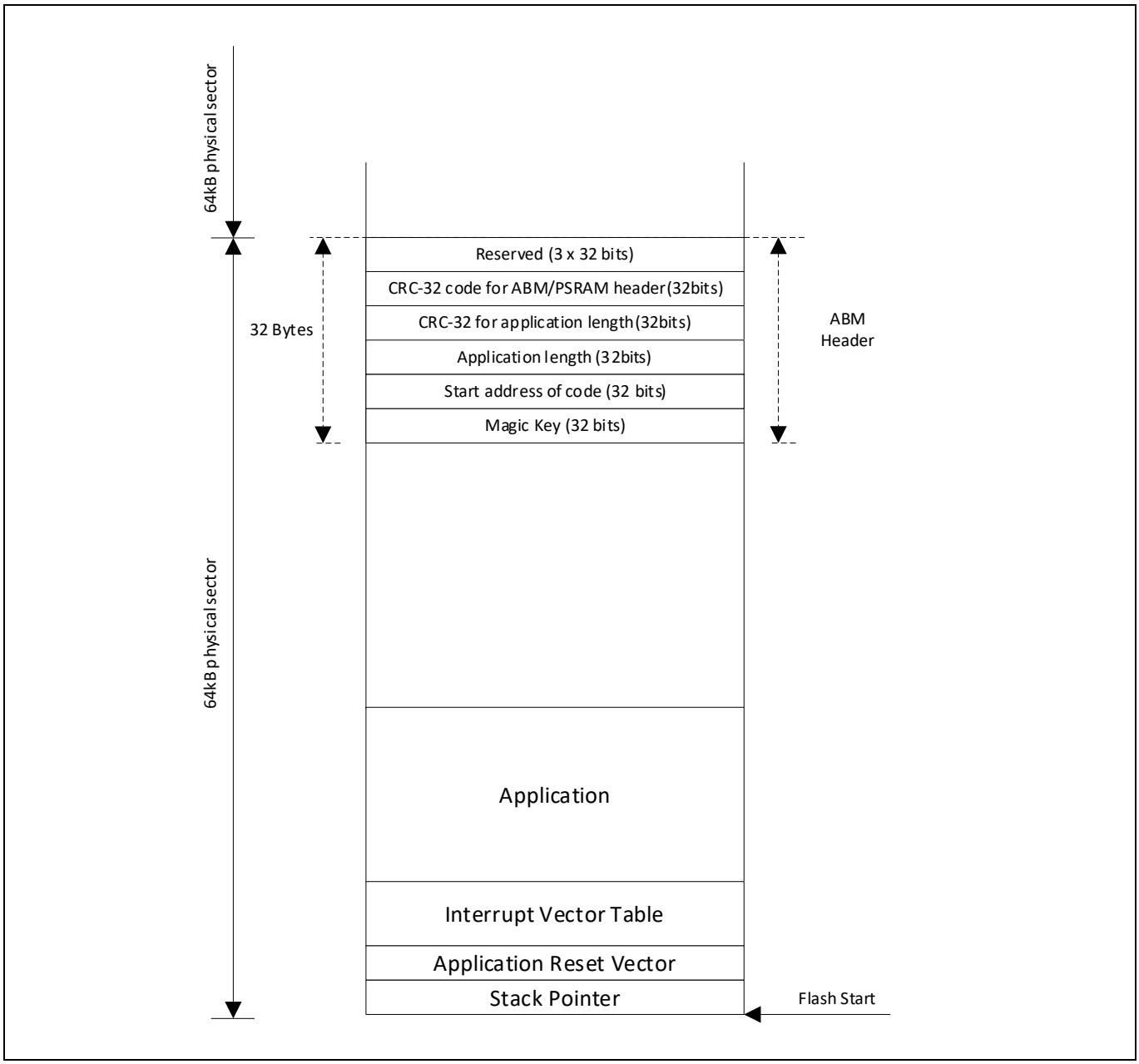

Figure 15 ABM/PSRAM Header - Location of reserved bytes

Revision history

Document revision	Date	Description of changes
1.0	2016-01	Initial AA step version.
1.1	2016-03	Wrong USIC version was used in generating V1.0 therefore the following issue is added: USIC_AI.017 and the following issues are removed: USIC_AI.005, USIC_AI.006, USIC_AI.007, USIC_AI.009, USIC_AI.010, USIC_AI.011, USIC_AI.013, USIC_AI.015, USIC_AI.016.
1.2	2016-10	Added Functional Deviations: ADC_CM.001, CCU8_AI.006, CCU_AI.008, DAC_CM.003, EBU_CM.001, FCE_CM.001, PORTS_CM.007, STARTUP_CM.00, USB_CM.005 Added Application Hints: ETH_AI.H001, USIC_AI.H004 Added Application Documentation Updates: WDT_CM.D001. USB_CM.004 was included eroneously in previous version and has been removed.
1.3	2017-08	Added Functional Deviations: ADC_CM.002. Added Application Documentation Updates: MultiCan_CM.D001 For updates and new issues see column "Chg" in the tables below.
1.4	2018-05	Added Functional Deviations: CPU_CM.005 Updated Functional Deviations: PORTS_CM.001 Added Application Hints: ECAT_CM.H001, PORTS_CM.H002 Added Documentation Updates: MPU_CM.D001, STARTUP_CM.D003 For updates and new issues see column "Chg" in the tables below.
1.5	2021-01	This document. Added Functional Deviations: USIC_AI.021, STARTUP_CM.003 Added Application Hints: MultiCAN_AI.H009 For updates and new issues see column "Chg" in the tables below.
1.6	2026-11-02	Migrated to latest template. This document. Updated Functional Deviations: PARITY_CM.001 (retitled to "Parity error signaling deviation in immediate write/read sequences"; extended with Case B describing spurious BusFault on immediate same-size / same-address write/read to a memory cell that holds pre-existing invalid parity; workaround section extended with SRAM initialization and DSB recommendations). Added Application Hints: ECAT_CM.H002 (SSC V5.13 dynamic enabling of Sync0/Sync1 in the ESC AL Event Mask register must not be used on XMC4800, because Sync0/Sync1 are already delivered to the CPU through dedicated internal interrupt lines; the "Get Sync mapped to AL Event indication" block in StartInputHandler() must be removed as delivered by SSC_Infineon.patch). For updates and new issues see column "Chg" in the tables below.

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition: 2026-11-02

Published by
Infineon Technologies AG
Am Campeon 1-15
85579 Neubiberg, Germany

© 2026 Infineon Technologies AG.
All rights reserved

Do you have a question about this document?
Email: erratum@infineon.com

Document reference:

V1.6

Important Notice
Products which may also include samples and may be comprised of hardware or software or both ("Product(s)") are sold or provided and delivered by Infineon Technologies AG and its affiliates ("Infineon") subject to the terms and conditions of the frame supply contract or other written agreement(s) executed by a customer and Infineon or, in the absence of the foregoing, the applicable Sales Conditions of Infineon. General terms and conditions of a customer or deviations from applicable Sales Conditions of Infineon shall only be binding for Infineon if and to the extent Infineon has given its express written consent.
For the avoidance of doubt, Infineon disclaims all warranties of non-infringement of third-party rights and implied warranties such as warranties of fitness for a specific use/purpose or merchantability.
Infineon shall not be responsible for any information with respect to samples, the application or customer's specific use of any Product or for any examples or typical values given in this document.
If this document is marked as 'Preliminary', 'Target', 'Draft' or 'Proposal', Infineon reserves the right to change all information given in this document at any time without notice. Subject to the development and release of a Product for series supply by Infineon, the technical specifications of the Product are set forth in the relevant datasheet provided by Infineon without such marking.
The data contained in this document is exclusively intended for technically qualified and skilled customer representatives. It is the responsibility of the customer to evaluate the suitability of the Product for the intended application and the customer's specific use and to verify all relevant technical data contained in this document in the intended application and the customer's specific use. The customer is responsible for properly designing, programming, and testing the functionality and safety of the intended application, as well as complying with any legal requirements related to its use.
Unless otherwise explicitly approved by Infineon, Products may not be used in any application where a failure of the Products or any consequences of the use thereof can reasonably be expected to result in personal injury. However, the foregoing shall not prevent the customer from using any Product in such fields of use that Infineon has explicitly designed and sold it for, provided that the overall responsibility for the application lies with the customer.
Infineon expressly reserves the right to use its content for commercial text and data mining (TDM) according to applicable laws, e.g. Section 44b of the German Copyright Act (UrhG).
If the Product includes security features:
Because no computing device can be absolutely secure, and despite security measures implemented in the Product, Infineon does not guarantee that the Product will be free from intrusion, data theft or loss, or other breaches ("Security Breaches"), and Infineon shall have no liability arising out of any Security Breaches.
If this document includes or references software:
The software is owned by Infineon under the intellectual property laws and treaties of the United States, Germany, and other countries worldwide. All rights reserved.
Therefore, you may use the software only as provided in the software license agreement accompanying the software.
If no software license agreement applies, Infineon hereby grants you a personal, non-exclusive, non-transferable license (without the right to sublicense) under its intellectual property rights in the software (a) for software provided in source code form, to modify and reproduce the software solely for use with Infineon hardware products, only internally within your organization, and (b) to distribute the software in binary code form externally to end users, solely for use on Infineon hardware products. Any other use, reproduction, modification, translation, or compilation of the software is prohibited.
For further information on the Product, technology, delivery terms and conditions, and prices, please contact your nearest Infineon office or visit https://www.infineon.com.