Errata Sheet

Rel. 1.6, 2018-07

Device	**XMC4400**
Marking/Step	**ES-AB, AB**
Package	**PG-LQFP-64/100**

Overview

Document ID is **084/18**.

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

Table 1 Current User Documentation

Document	Version	Date
XMC4400 Reference Manual	V1.6	July 2016
XMC4400 Data Sheet	V1.1	March 2014

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

Notes

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

Conventions used in this Document

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

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

1 History List / Change Summary

Table 2 History List

Version	Date	Remark
1.0	2013-05	Initial AB step version. Previous step is AA. Changes wrt. XMC4400 AA Errata Sheet v1.1: Added DEBUG_CM.003.
1.1	2013-09	ADC_AI.002 was listed as Erratum in v1.0. Item is validated in v1.1 and Errata removed. Please note ADC_AI.H003 has been added in v1.1 regarding this Topic. Updated RESET_CM.H001
1.2	2014-04	Added: ADC_TC.064, DSD_AI.001, HRPWM_AI.002, HRPWM_AI.004, SCU_CM.015, USIC_AI.008, USIC_AI.020, ADC_AI.H008, ADC_TC.H011
1.3	2016-05	Added: ADC_AI.016, CACHE_CM.001, CCU8_AI.006, CCU_AI.006, DSD_AI.002, DTS_CM.001, FCE_CM.001, PARITY_CM.002, PORTS_CM.007, SCU_CM.021, USB_CM.004, WDT_CM.001, POWER_CM.P001, POWER_CM.P003, POWER_CM.P004, ETH_AI.H001, USIC_AI.H004 Changed: ADC_TC.064, SCU_CM.015, ADC_TC.H011, MultiCAN_AI.H005, RESET_CM.H001
1.4	2016-10	Added: ADC_CM.001, CCU_AI.008, DAC_CM.003, USB_CM.005, MultiCAN_AI.H009, WDT_CM.D001

Subject to Agreement on the Use of Product Information

Table 2 History List (cont'd)

Version	Date	Remark
1.5	2017-08	This Document. Added Functional Deviation: ACD_CM.002 For changes see column "Chg" in the tables below.
1.6	2018-07	This document. Added Functional Deviations: CPU_CM.005 Added Application Hints: PORTS_CM.H002 Added Documentation Updates: MPU_CM.D001, STARTUP_CM.D003 For updates and new issues see column "Chg" in the tables below.

Table 3 Errata fixed in this step

Errata	Short Description	Change
ADC_AI.002	Result of Injected Conversion may be wrong	Fixed
CCU8_AI.002	CC82 Timer of the CCU8x module cannot use the external shadow transfer trigger connected to the POSIFx module	Fixed
ETH_CM.001	Ethernet module synchronization limits CPU frequency	Fixed
HRPWM_AI.003	HRPWM Usage Limitations	Fixed
PMU_CM.001	Branch from non-cacheable to cacheable address space instruction may corrupt the program execution	Fixed
PORTS_CM.002	P0.9 Pull-up permanently active	Fixed
STARTUP_CM.001	CAN Bootstrap Loader	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		11
ADC_AI.016	No Channel Interrupt in Fast Compare Mode with GLOBRES		12
ADC_CM.001	Conversion results can be wrong if groups are not synchronized		12
ADC_CM.002	Converter diagnostics not functional		14
ADC_TC.064	Effect of conversions in 10-bit fast compare mode on post-calibration		15
CACHE_CM.001	Instruction buffer invalidation control bit needs to be cleared after an invalidation was triggered		16
CCU8_AI.003	CCU8 Parity Checker Interrupt Status is cleared automatically by hardware		17
CCU8_AI.004	CCU8 output PWM glitch when using low side modulation via the Multi Channel Mode		19
CCU8_AI.006	Timer concatenation does not work when using external count signal		22
CCU_AI.002	CCU4 and CCU8 Prescaler synchronization clear does not work when Module Clock is faster than Peripheral Bus Clock		23
CCU_AI.004	CCU4 and CCU8 Extended Read Back loss of data		24
CCU_AI.005	CCU4 and CCU8 External IP clock Usage		26
CCU_AI.006	Value update not usable in period dither mode		28

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
CCU_AI.008	Clock ratio limitation when using MCSS inputs		28
CPU_CM.001	Interrupted loads to SP can cause erroneous behavior		29
CPU_CM.004	VDIV or VSQRT instructions might not complete correctly when very short ISRs are used		30
CPU_CM.005	Store immediate overlapping exception return operation might vector to incorrect interrupt	New	32
DAC_CM.003	FIFO usage limitation in "Data Processing Mode"		33
DEBUG_CM.003	AB step chip revision number mismatch		33
DSD_AI.001	Possible Result Overflow with Certain Decimation Factors		34
DSD_AI.002	Timestamp can be calculated wrong		34
DTS_CM.001	DTS offset calibration value limitations		37
ETH_AI.001	Incorrect IP Payload Checksum at incorrect location for IPv6 packets with Authentication extension header		37
ETH_AI.002	Incorrect IP Payload Checksum Error status when IPv6 packet with Authentication extension header is received		38
ETH_AI.003	Overflow Status bits of Missed Frame and Buffer Overflow counters get cleared without a Read operation		39
FCE_CM.001	Result value is wrong if read directly after last write		40

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
HRPWM_AI.001	HRPWM output signal interference while using two control sources		40
HRPWM_AI.002	HRPWM CSG missing DAC conversion trigger in static mode		43
HRPWM_AI.004	HRPWM Peripheral Bus Clock Limitation		45
LEDTS_AI.001	Delay in the update of FNCTL.PADT bit field		46
PARITY_CM.002	Clock limitations for ETH and SDMMC modules when using parity check of module SRAMs		51
PORTS_CM.005	Different PORT register reset values after module reset		52
PORTS_CM.007	P14 and P15 cannot be used in boundary scan test		53
POSIF_AI.001	Input Index signal from Rotary Encoder is not decoded when the length is 1/4 of the tick period		53
SCU_CM.006	Deep sleep entry with PLL power-down option generates SOSCWDGT and SVCOLCKT trap		55
SCU_CM.015	Functionality of parity memory test function limited		56
SCU_CM.021	Registering of service requests in SRRAW register can fail		57
USB_CM.004	USB core is not able to detect resume or new session request after PHY clock is stopped		58
USB_CM.005	DMA support for USB host mode operation		59
USIC_AI.008	SSC delay compensation feature cannot be used		59

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
USIC_AI.010	Minimum and maximum supported word and frame length in multi-IO SSC modes		60
USIC_AI.013	SCTR register bit fields DSM and HPCDIR are not shadowed with start of data word transfer		60
USIC_AI.014	No serial transfer possible while running capture mode timer		61
USIC_AI.015	Wrong generation of FIFO standard transmit/receive buffer events when TBCTR.STBTEN/RBCTR.SRBTEN = 1		61
USIC_AI.016	Transmit parameters are updated during FIFO buffer bypass		61
USIC_AI.017	Clock phase of data shift in SSC slave cannot be changed		62
USIC_AI.018	Clearing PSR.MSLS bit immediately deasserts the SELOx output signal		62
USIC_AI.019	First data word received by IIC receiver triggers RIF instead of AIF		63
USIC_AI.020	Handling unused DOUT lines in multi-IO SSC mode		64
WDT_CM.001	No overflow is generated for WUB default value		64

Table 5 Deviations from Electrical- and Timing Specification

AC/DC Deviation	Short Description	Chg	Pg
POWER_CM.P001	Risk of increased current consumption in internally controlled hibernate mode		65

Table 5 Deviations from Electrical- and Timing Specification (cont'd)

AC/DC Deviation	Short Description	Chg	Pg
POWER_CM.P003	Current consumption when executing from PSRAM		65
POWER_CM.P004	Current consumption while PORST low can exceed specified value		66

Table 6 Application Hints

Hint	Short Description	Chg	Pg
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion		67
ADC_AI.H004	Completion of Startup Calibration		68
ADC_AI.H008	Injected conversion with broken wire detection		68
ADC_TC.H011	Bit DCMSB in register GLOBCFG		69
ETH_AI.H001	Sequence for Switching between MII and RMII Modes		70
MultiCAN_AI.H005	TxD Pulse upon short disable request		70
MultiCAN_AI.H006	Time stamp influenced by resynchronization		70
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off		71
MultiCAN_AI.H008	Effect of CANDIS on SUSACK		71
MultiCAN_AI.H009	Behavior of MSGVAL for Remote Frames in Single Data Transfer Mode - Documentation Update		72
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		73
MultiCAN_TC.H004	Double remote request		73
PORTS_CM.H002	Class A2 pins GPIO driver strength configuration	New	74

Table 6 Application Hints (cont'd)

Hint	Short Description	Chg	Pg
RESET_CM.H001	Power-on reset release		75
USIC_AI.H004	I2C slave transmitter recovery from deadlock situation		76

Table 7 Documentation Updates

Hint	Short Description	Chg	Pg
MPU_CM.D001	No restrictions on using Bit5 to Bit8 of register MPU_RBAR	New	77
STARTUP_CM.D003	Alignment of ABM/PSRAM Header	New	77
WDT_CM.D001	Correction to section "Pre-warning Mode"		78

2 Functional Deviations

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

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

In the following scenario:

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

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

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

Workaround

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

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

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

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

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

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

Workaround

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

ADC_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 the figure below for $G0_2$ and $G1_1$. In this example the division factor is 5 (DIVA=4).

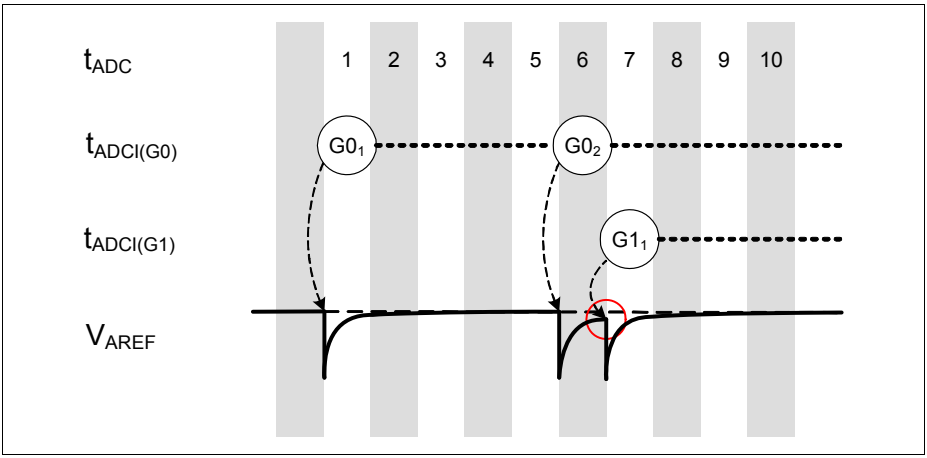

Figure 1 Influence of concurrent conversions on reference voltage

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 mesurements 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 behaviour.

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
```

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.

- Step 6:
```
-- 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
```

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.

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

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

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

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

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

Workaround

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

CACHE_CM.001 Instruction buffer invalidation control bit needs to be cleared after an invalidation was triggered

The device reference manual describes the PCON.IINV bit of PREF unit as write only. Writing 1_B is initiating the invalidation of the instruction buffer, and writing 0_B has no effect.

However, writing 1_B to PCON.IINV will force sustaining instruction buffer invalidation until the bit is cleared again by software writing 0_B.

Implications

As long as PCON.IINV remains set to 1_B, the system will not benefit from the performance improvement by the instruction buffer. In fact, as the prefetch unit attempts to perform instruction buffer refills in the background, the system may show slower code execution performance as with execution from uncached Flash address range or with bypassed instruction buffer.

Workaround

Execute the following sequence if the PREF instruction cache needs to be invalidated:

1. Branch to RAM or uncached Flash address range.
2. Enable the instruction buffer bypass.
   PCON.IBYP = 1_B
3. Invalidate the instruction buffer.
   PCON.IINV = 1_B
4. Clear the invalidate bit.
   PCON.IINV = 0_B
5. Disable the instruction buffer bypass.
   PCON.IBYP = 0_B
6. Read back PCON to resolve pipelining effects.
7. Return to cacheable Flash address range.

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

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

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

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

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

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

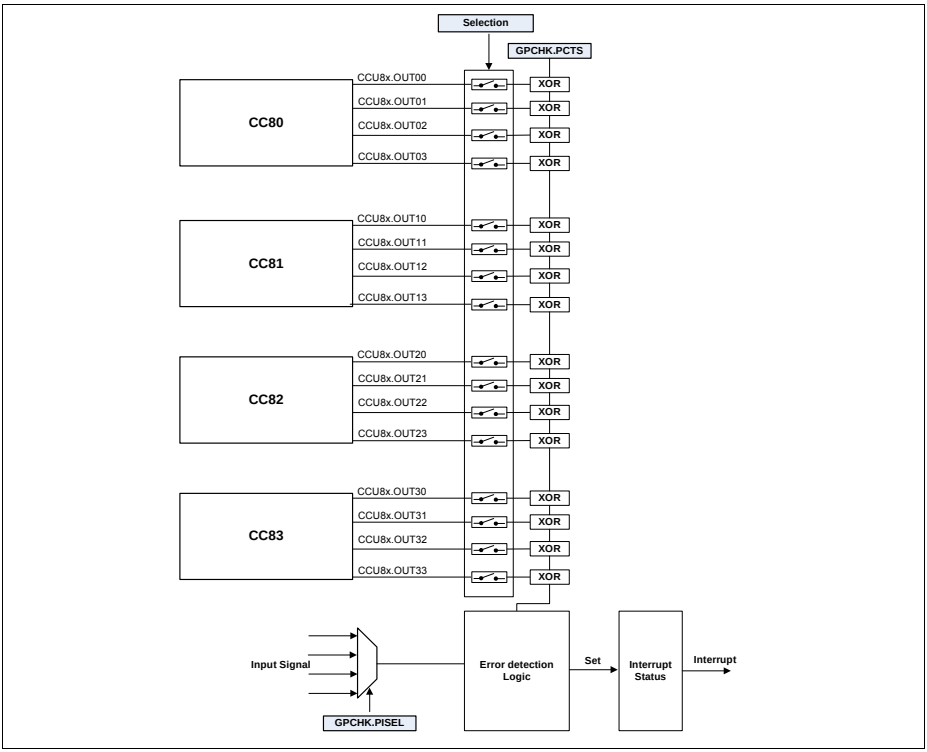

Figure 2 Parity Checker diagram

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

CCU8_AI.004 CCU8 output PWM glitch when using low side modulation via the Multi Channel Mode

Each CCU8 Timer Slice can be configured to use the Multi Channel Mode - this is done by setting the CC8yTC.MCME1 and/or CC8yTC.MCME2 bit fields to 1_B. Each bit field enables the multi channel mode for the associated compare channel of the CCU8 Timer Slice (each CCU8 Timer Slice has two compare channels that are able to generate each a complementary pair of PWM outputs).

After enabled, the Multi Channel mode is then controlled by several input signals, one signal per output. Whenever an input is active, the specific PWM output is set to passive level - Figure 1.

The Multi Channel mode is normally used to modulate in parallel several PWM outputs (a complete CCU8 - up to 16 PWM signals can be modulated in parallel).

A normal use case is the parallel control of the PWM output for BLDC motor control. In Figure 2, we can see the Multi Channel Pattern being updated synchronously to the PWM signals. Whenever a multi channel input is active (in this case 0), the specific output is set into passive level (the level in which the external switch is OFF).

Subject to Agreement on the Use of Product Information

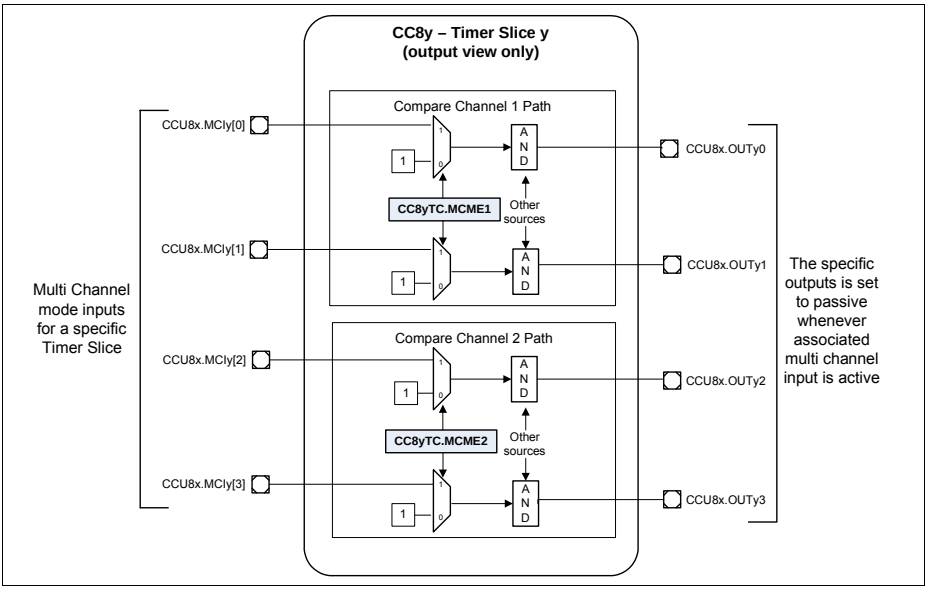

Figure 3 Multi Channel Mode diagram

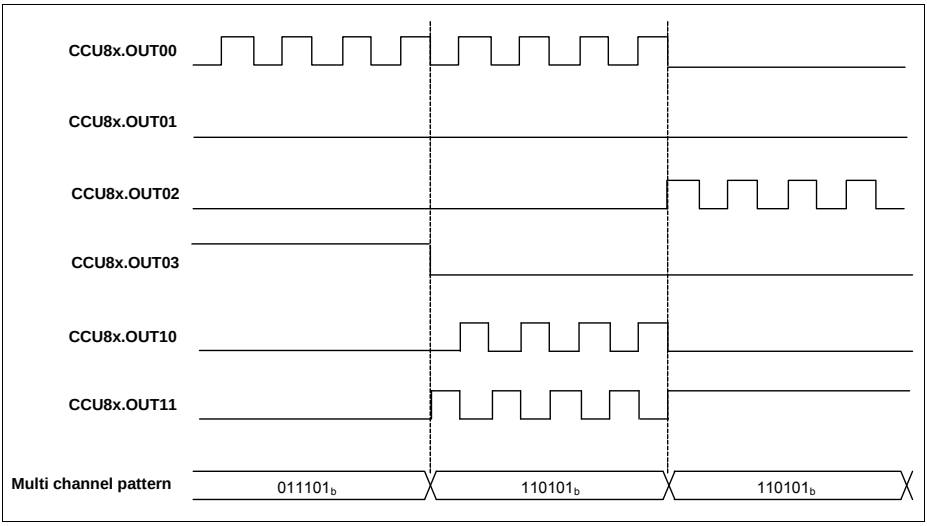

Figure 4 Multi Channel Mode applied to several CCU8 outputs

A glitch is present at the PWM outputs whenever the dead time of the specific compare channel is enabled - CC8yDTC.DTE1 and/or CC8yDTC.DTE2 set to 1_B (each compare channel has a separate dead time function) - and the specific multi channel pattern for the channel is 01_B or 10_B.

This glitch is not present if the specific timer slice is configure in symmetric edge aligned mode - CC8yTC.TCM = 0_B and CC8yCHC.ASE = 0_B.

This glitch only affects the PWM output that is linked to the inverting ST path of each compare channel (non inverting outputs are not affected).

The effect of this glitch can be seen in Figure 3. The duration of the PWM glitch has the same length has the dead time value programmed into the CC8yDC1R.DT1F field (for compare channel 1) or into the CC8yDC1R.DT2F.

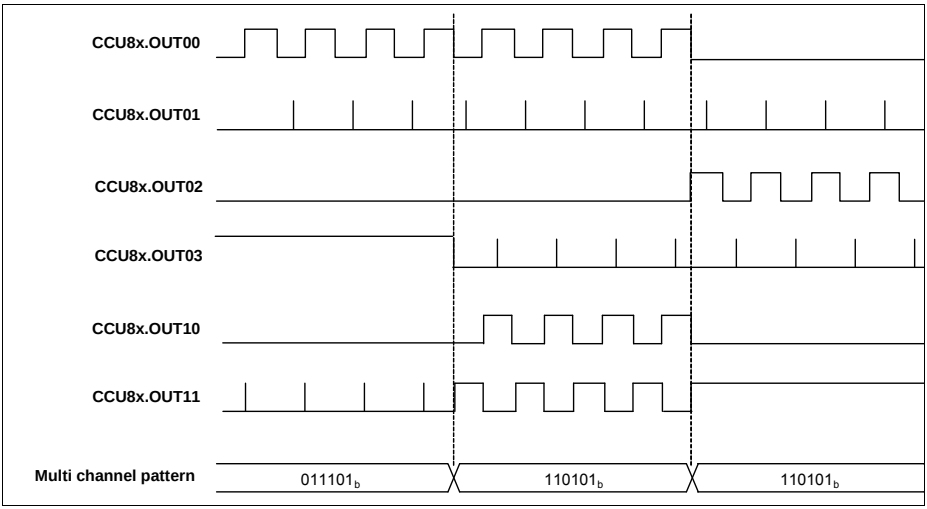

Figure 5 PWM output glitch

Workaround

To avoid the glitch on the inverting path of the PWM output, one can disable the dead time function before the Multi Channel Pattern is set to 01_B or 10_B. Disabling the dead time of the inverting PWM output can be done by setting:

```
CC8yDTC.DCEN2 = 0 //if compare channel 1 is being used
```

```
CC8yDTC.DCEN4 = 0 //if compare channel 2 is being used
```

The dead time needs to be re enabled, before the complementary outputs become modulated at the same time:

```
CC8yDTC.DCEN2 = 1 //if compare channel 1 is being used
CC8yDTC.DCEN4 = 1 //if compare channel 2 is being used
```

CCU8_AI.006 Timer concatenation does not work when using external count signal

Each CCU8 peripheral contains four sixteen 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 1 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 1 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 1, 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 concatenate - because the count should be used to increment the LSB timer. The LSB timer will then in each wrap around, increment the MSB timer.

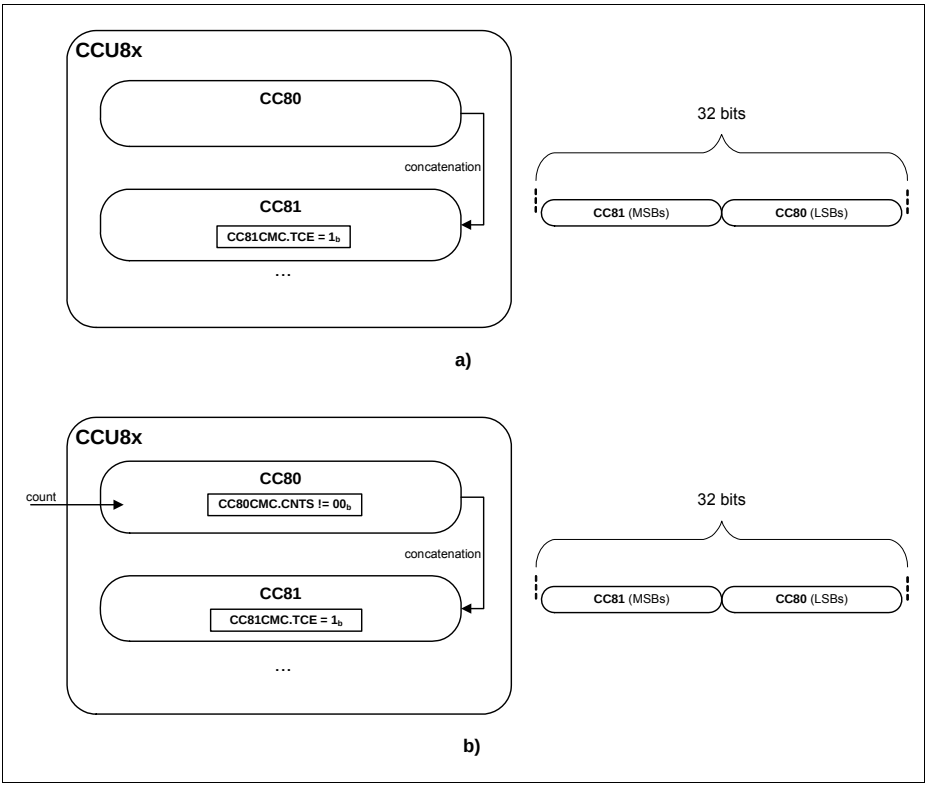

Figure 6 CCU8x concatenation feature resource configuration - a) without external count function; b) with external count function

Workaround

None

CCU_AI.002 CCU4 and CCU8 Prescaler synchronization clear does not work when Module Clock is faster than Peripheral Bus Clock

Each CCU4/CCU8 module contains a feature that allows to clear the prescaler division counter synchronized with the clear of a run bit of a Timer Slice. This is

configure via the GCTRL.PRBC field. The default value of 000_B dictates that only the software can clear the prescaler internal division counter. Programming a value different from 000_B into the PRBC will impose that the prescaler division counter is cleared to 0_D whenever the selected Timer Slice (selected via the PRBC field) run bit is cleared (TRB bit field).

In normal operating conditions, clearing the internal prescaler division counter is not needed. The only situation were a clear of the division may be needed is when several Timer Slices inside one unit (CCU4/CCU8) are using different prescaling factors and a realignment of all the timer clocks is needed. This normally only has a benefit if there is a big difference between the prescaling values, e.g. Timer Slice 0 using a module clock divided by 2_D and Timer Slice 1 using a module clock divided by 1024_D.

When the peripheral bus clock frequency is smaller than the CCU4/CCU8 module clock frequency, $f_{periph} < f_{ccu}$, it is not possible to clear the prescaler division counter, synchronized with the clear of the run bit of one specific Timer Slice.

Workaround 1

The clearing of the prescaler internal division counter needs to be done via software: GCTRL.PRBC programmed with 000_B and whenever a clear is needed, writing 1_B into the GIDLS.CPRB bit field.

Workaround 2

When the usage of the Prescaler internal division clear needs to be synchronized with a timer run bit clear, the module clock of the CCU4/CCU8 should be equal to the peripheral bus clock frequency: $f_{periph} = f_{ccu}$.

To do this, the following SCU (System Control Unit) registers should be set with values that force this condition: CCUCLKCR.CCUDIV, CPUCLKCR.CPUDIV and PBCLKCR.PBDIV.

CCU_AI.004 CCU4 and CCU8 Extended Read Back loss of data

Each CCU4/CCU8 Timer Slice contains a bit field that allows the enabling of the Extended Read Back feature. This is done by setting the

CC8yTC.ECM/CC4yTC.ECM = 1_B. Setting this bit field to 1_B only has an impact if the specific Timer Slice is working in Capture Mode (CC8yCMC.CAP1S or CC8yCMC.CAP0S different from 00_B - same fields for CCU4).

By setting the bit field to ECM = 1_B, is then possible to read back the capture data of the specific Timer Slice (or multiple Timer Slices, if this bit field is set in more than one Timer Slice) trough a single address. This address is linked to the ECRD register.

Referring to **Figure 7**, the hardware every time that the software reads back from the ECRD address, will return the immediately next capture register that contains new data. This is done in a circular access, that contains all the capture registers from the Timer Slices that are working in capture mode.

When using this feature, there is the possibility of losing captured data within a Timer Slice. The data that is lost is always the last captured data within a timer slice, e.g (with CCU4 nomenclature - same applies to CCU8):

- Timer X has 4 capture registers and is the only Timer set with ECM = 1_B. At the moment that the software starts reading the capture registers via the ECRD address, we have already capture four values. The ECRD read back will output CC4xC0V -> CC4xC1V -> CC4xC2V -> CC4xC2V (CC4xC3V value is lost)
- Timer X has 4 capture registers and is the only Timer set with ECM = 1_B. At the moment that the software starts reading the capture registers via the ECRD address, we have already capture two values. The ECRD read back will output CC4xC2V -> CC4xC2V (CC4xC3V value is lost)
- Timer X and Timer Y have 4 capture registers each and they are both configured with ECM = 1_B. At the moment that the software starts reading the capture registers via the ECRD address, we have already capture two values on Timer X and 4 on Timer Y. The ECRD read back will output CC4xC0V -> CC4xC1V -> CC4xC2V -> CC4xC3V -> CC4yC2V -> CC4yC2V (CC4yC3V value is lost)

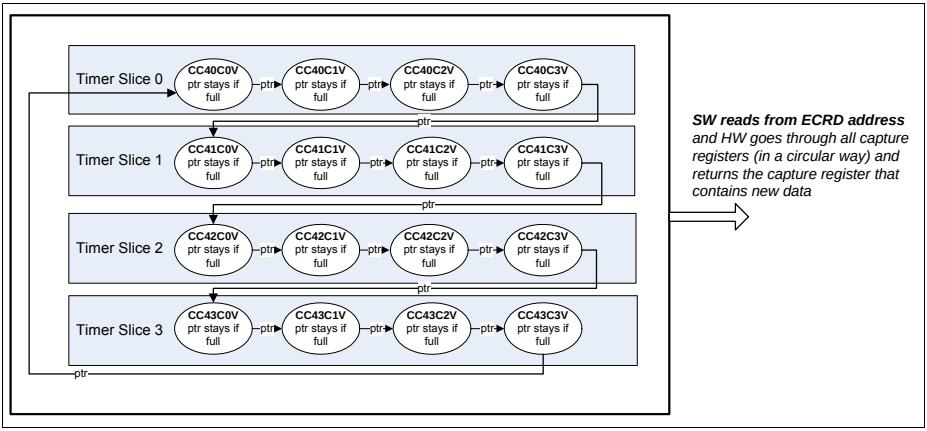

Figure 7 Extended Read Back access - example for CCU4 (CCU8 structure is the same)

Workaround

None.

CCU_AI.005 CCU4 and CCU8 External IP clock Usage

Each CCU4/CCU8 module offers the possibility of selecting an external signal to be used as the master clock for every timer inside the module Figure 1. External signal in this context is understood as a signal connected to other module/IP or connected to the device ports.

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

This division is selected by the PSIV field inside of the CC4yPSC/CC8yPSC register. Notice that each Timer Slice (CC4y/CC8y) have a specific PSIV field, which means that each timer can operate in a different frequency.

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

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

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

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

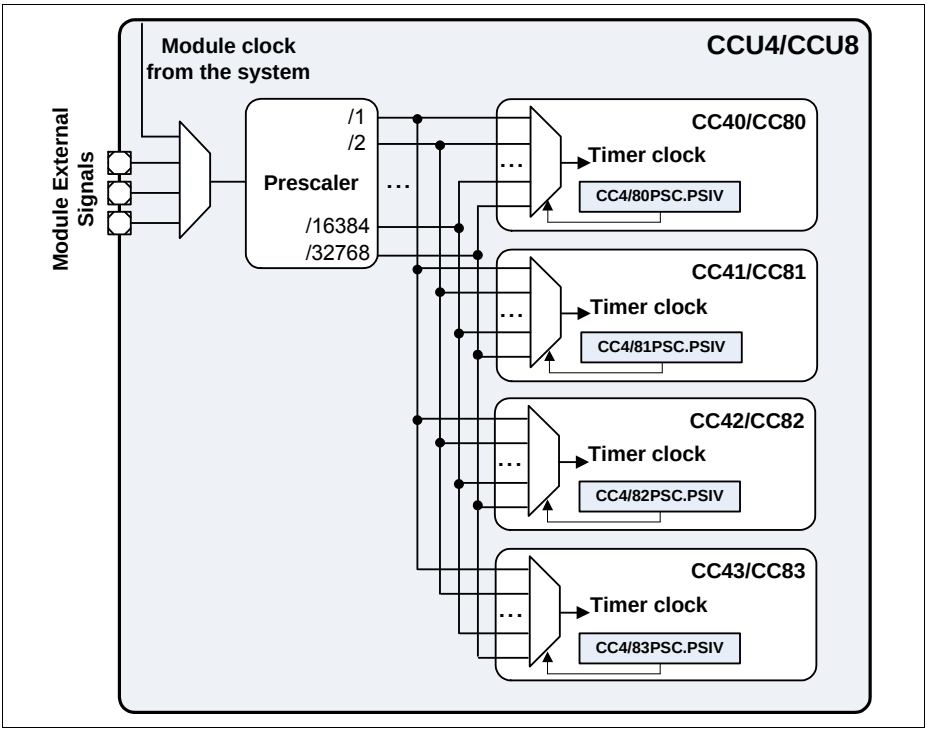

Figure 8 Clock Selection Diagram for CCU4/CCU8

Workaround

None.

CCU_AI.006 Value update not usable in period dither mode

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

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

- DITHE = 00_B - dither disabled
- DITHE = 01_B - dither applied to the 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 an period value)

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

Implication

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

Workaround

None.

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

CPU_CM.001 Interrupted loads to SP can cause erroneous behavior

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

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

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

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

Conditions

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

Implications

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

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

Workaround

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

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

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

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

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

Conditions

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

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

Implications

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

Workaround

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

There are two workarounds:

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

CPU_CM.005 Store immediate overlapping exception return operation might vector to incorrect interrupt

The 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");
}
```

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 misbehaviour 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.

DEBUG_CM.003 AB step chip revision number mismatch

The chip revision number of the AB step devices is 0x2.

This number is correctly returned in bits [31:28] when reading the JTAG IDCODE thru the debug interface.

Same number should be returned in bits [7:4] when reading the ROM table PID2 cell by software. The erronous returned value however is 0x0.

Workaround

Both values 0x0 and 0x2 are unique identifiers for the AB step and can be used. Software must be made aware of the mismatch.

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.

DSD_AI.002 Timestamp can be calculated wrong

Some applications need to determine a result value at points of time in between two regular output values. An interpolation algorithm is then used to determine the point of time in relation to the last regular result.

The cycles consumed since the last regular result value can be calculated from the decimation factor FCFGCx.CFMDF and bit fields NVALCNT and CFMDCNT of TSTMPx register.

In the affected device the value of NVALCNT is calculated wrong upon underflow of CFMDCNT.

Implications

Calculation for cycles consumed (TICKS) since the last regular result value is done according to the following formula.

TICKS = NVALCNT * (CFMDF + 1) + (CFMDF – CFMDCNT) (1)

Upon underflow of CFMDCNT the following actions appear:

- NVALCNT is increased
- CFMDCNT is reloaded from CFMDF

Examples for expected (**Table 8**) and wrong (**Table 9**) behavior are shown below.

Table 8 Expected values using CFMDF = 15

NVALCNT	CFMDCNT	TICKS
2	3	44
2	2	45
2	1	46
2	0	47
3 - correct value	15	48 - correct value
3	14	49
...	...	...

The wrong behavior is that NVALCNT increases only one TICK after the underflow:

Table 9 Measured values using CFMDF = 15

NVALCNT	CFMDCNT	TICKS
2	3	44
2	2	45
2	1	46
2	0	47

Table 9 Measured values using CFMDF = 15 (cont'd)

NVALCNT	CFMDCNT	TICKS
2 - wrong value!	15	**32 - wrong value!**
3	14	49
...	...	...

The cycles consumed are therefore calculated wrong for the later case. Consequently the interpolation is wrong.

For long DSD periods a wrong interpolation can lead to substantial error.

Workaround 1

- Read TSTMPx register containing CFMDCNT and NVALCNT values
- While CFMDCNT is equal to FCFGCx.CFMDF repeat the read of TSTMPx
- Calculate cycles consumed

Workaround 2

- Read TSTMPx register containing CFMDCNT and NVALCNT values
- If CFMDCNT is equal to FCFGCx.CFMDF then increment NVALCNT
- Calculate cycles consumed

Note that this workaround delivers a wrong result for the case that TSTMPx is read at the same time as the last regular output occurred (see **Table 10** below).

Therefore in case of NVALCNT=0 the read of TSTCMPx may be repeated as described in Workaround 1.

Alternatively the wrong result must be regarded by the application.

Table 10 Values calculated with Workaround 2 using CFMDF = 4

NVALCNT	CFMDCNT	TICKS
0 - correct value	4	**5 - wrong value!**
0	3	1
0	2	2
0	1	3
0	0	4

Table 10 Values calculated with Workaround 2 using CFMDF = 4

NVALCNT	CFMDCNT	TICKS
0 - wrong value!	4	5 - this and later values are correct
1	3	6
...	...	...

DTS_CM.001 DTS offset calibration value limitations

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

Implication

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

Workaround

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

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

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

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

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

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

Workaround

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

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

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

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

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

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

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

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

Workaround

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

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

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

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

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

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

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

Effects

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

Workaround

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

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.

HRPWM_AI.001 HRPWM output signal interference while using two control sources

The High Resolution PWM (HRPWM) unit has 4 High Resolution Channel (HRC) sub modules. These are the sub modules that are used to extend the normal PWM resolution up to 150 ps. The structure of each of these sub modules is depicted in **Figure 9**.

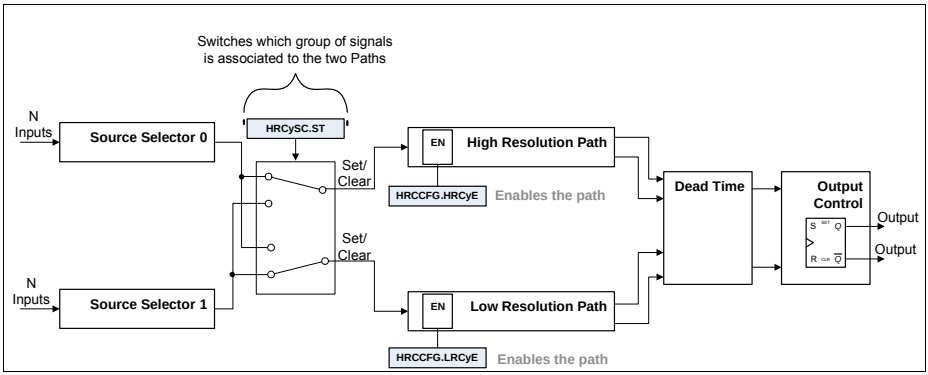

Figure 9 High Resolution Channel (HRC) simplified block diagram

From the scheme of each HRC, it is possible to control the output PWM signal via two Source Selectors (Source Selector 0 and Source Selector 1). Each of these Source Selectors can generate a pair of PWM set and clear signals (that are propagated to the output control stage).

When both paths are being used (High Resolution Path and Low Resolution Path), the Source Selector 0 output signals always have priority over the signals generated via Source Selector 1.

At any given instant a bitfield that controls which Source Selector is connected to which path can be updated by Software, via the HRCyST.ST bitfield.

When only one path is used (only High Resolution Path or only High Resolution Path) the software can still update at any given time which is the Source Selector controlling this specific path (via the HRCyST.ST bitfield).

This scenario where only one path is enabled but both Source Selectors are being used, and by being used is understood that signals are being actively decoded from the Source Selectors (input signals for the Source Selectors are being updated) is depicted on **Figure 10** - in this example the High Resolution Path is the one being used.

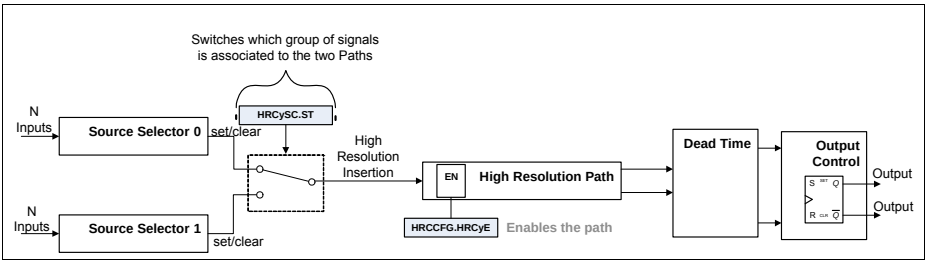

Figure 10 HRC with only one path enabled and two Source Selectors

In the above scenario, every time that the HRCyST.ST bitfield is 1_B, the Source Selector 1 is the one that has control over the PWM signal. Whenever the HRCyST.ST bitfield is 0_B is the Source Selector 0 that has control over the PWM signal.

An issue exists whenever the Source Selector 1 is selected (HRCyST.ST = 1_B) and there is a collision between signals from both Source Selectors. A collision is understood has:

- at the same module clock time frame Source Selector 0 and Source Selector 1 both decode a signal (a PWM set or PWM clear).

If the described condition occurs, then the signal generated by Source Selector 1 is lost and the PWM signal is not handled properly - **Figure 11**.

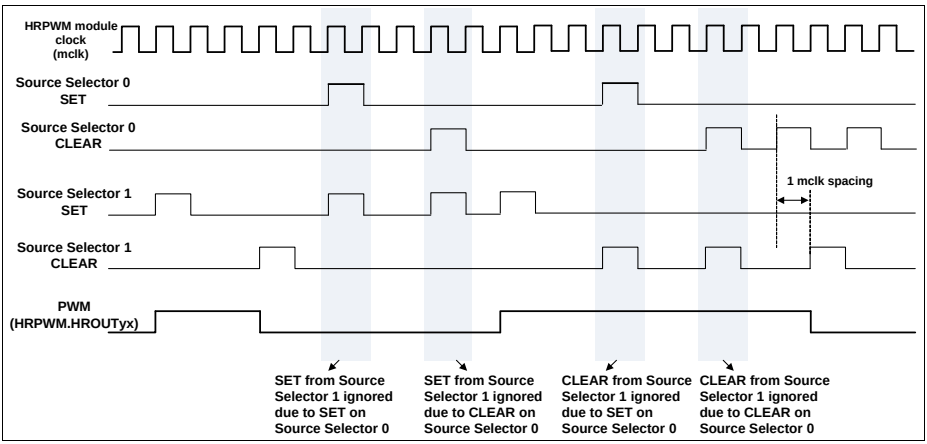

Figure 11 Collision between signals of the two Source Selectors - valid

only when HRCyST.ST = 1_B

Workaround 1

Ensuring that the signals controlling both Source Selectors are spaced by a minimum of 1 HRPWM module clock cycle. This is indeed only necessary if is currently Source Selector 1 the one selected to control the PWM path (HRCyST.ST = 1_B).

Workaround 2

Ensuring that the signals controlling Source Selector 0 are stopped whenever Source Selector 1 the one selected to control the PWM path (HRCyST.ST = 1_B).

HRPWM_AI.002 HRPWM CSG missing DAC conversion trigger in static mode

The High Resolution PWM (HRPWM) unit has 3 Comparator and Slope Generation (CSG) sub modules. These sub modules are comprised of a DAC an analog Comparator and additional control logic. The structure of each of these sub modules is depicted in **Figure 9**.

Each CSG has two major modes that can be used to control the DAC values:

- Static Mode
- Slope Generation Mode

The selection between these two modes can be done via the CSGySC.SCM field. When the SCM field is set to 0_H, the static mode is selected.

When the static mode is selected, the DAC is only converting a single value and is not generating any type of automatic pattern. Only the software can update the value that is converted by the DAC. This value needs to be written into the CSGyDSV1 or CSGyDSV2 registers.

When the static mode is selected (SCM = 0_H), the DAC conversion trigger is not generated whenever the CSGyDSV1 or CSGyDSV2 values are updated via software.

This issue imposes that during run time, it is not possible to update on-the-fly the DAC value when the static mode is selected (SCM = 0_H), without additional software routines or additional hardware resources.

When a pattern is being generated automatically by the hardware (SCM != 0_H), this issue does not exist.

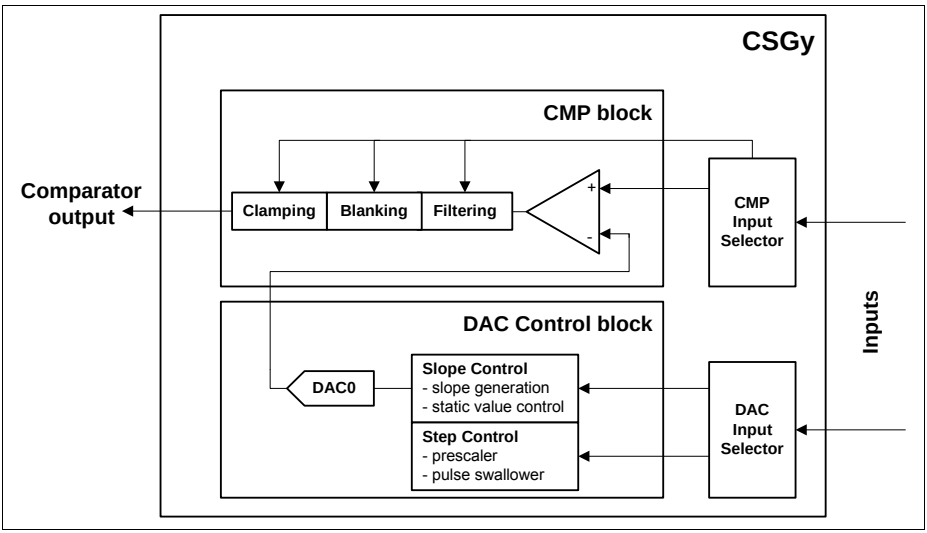

Figure 12 High Resolution Channel (HRC) simplified block diagram

Workaround

Every time that the DAC value needs to be updated (with SCM = 0_H), the DAC control logic run bit should be cleared and then re set.

When the CSGyDSV2 register is being used, then the following code should be implemented:

```
//configures the CSGyDSV2 value has the one used for the
//input register (y is the sub module instance)- this is
//one time configuration at startup
CSGySWSM = 0;

//During the run time the loop to update the DSV2 value is
//as follows (y is the sub module instance)
CSGyDSV2 = [NEW_VALUE];
CSGCLRG.CDyR = 1; //clears the DAC control logic run bit
CSGSETG.SDyR = 1; //sets the DAC control logic run bit
```

When the CSGyDSV1 register is being used, then the following code should be implemented:

```
//configures the CSGyDSV1 value has the one used for the
//input register (y is the sub module instance)- this is
//one time configuration at startup
CSGySWSM = 1;
CSGySC.IST = 1;

//During the run time the loop to update the DSV2 value is
//as follows (y is the sub module instance)
CSGySDSV1 = [NEW_VALUE];
CSGTRG = 0x1 //in case the CSG0 is being used. If other CSG
is being used then the specific bit position should be
written
CSGCLRG.CDyR = 1; //clears the DAC control logic run bit
CSGSETG.SDyR = 1; //sets the DAC control logic run bit
```

HRPWM_AI.004 HRPWM Peripheral Bus Clock Limitation

The High Resolution PWM (HRPWM) peripheral uses two clocks generated by the SCU, the module clock (defined as f_{ccu} in the System Control Unit) and the peripheral bus clock (defined as f_{periph} in the System Control Unit).

The module clock is the one being used by the HRPWM kernel. The peripheral bus clock is the one being used by the interface between the CPU and the HRPWM to write and read back the module registers.

Due to synchronization issues, the HRPWM module clock frequency needs to be the same as the peripheral bus clock frequency ($f_{periph} < f_{ccu}$).

The peripheral bus clock frequency is controlled via the PBCLKCR.PBDIV field. The HRPWM module clock frequency is controlled via the CCUCLKCR.CCUDIV field. Both register are located in the SCU (System Control Unit).

Workaround

The peripheral bus clock and the module clock of the HRPWM need to have the same frequency ($f_{periph} = f_{ccu}$).

This is achieved by setting accordingly the following three fields:

- the CCUCLKCR.CCUDIV (field controlling the HRPWM and CCU4, CCU8 and POSIF peripherals module clock ratio);
- the CPUCLKCR.CPUDIV (field controlling the CPU clock division ratio);
- the PBCLKCR.PBDIV (field controlling the peripheral bus clock division ratio)

Table 11 shows the valid combinations for each of the previously mentioned fields. Notice that the frequency values given in the table are just an example (the real frequency would depend on the specific device and the PLL configuration).

Table 11 Valid Clock Combinations

Valid	CCUCLKCR.CCUDIV (frequency example)	CPUCLKCR.CPUDIV (frequency example)	PBCLKCR.PBDIV (frequency example)
Yes	0 (120 MHz)	0 (120 MHz)	0 (120 MHz)
No	0 (120 MHz)	0 (120 MHz)	1 (60 MHz)
No	0 (120 MHz)	1 (60 MHz)	0 (60 MHz)
Yes	1 (60 MHz)	0 (120 MHz)	1 (60 MHz)
Yes	1 (60 MHz)	1 (60 MHz)	0 (60 MHz)

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 13**).

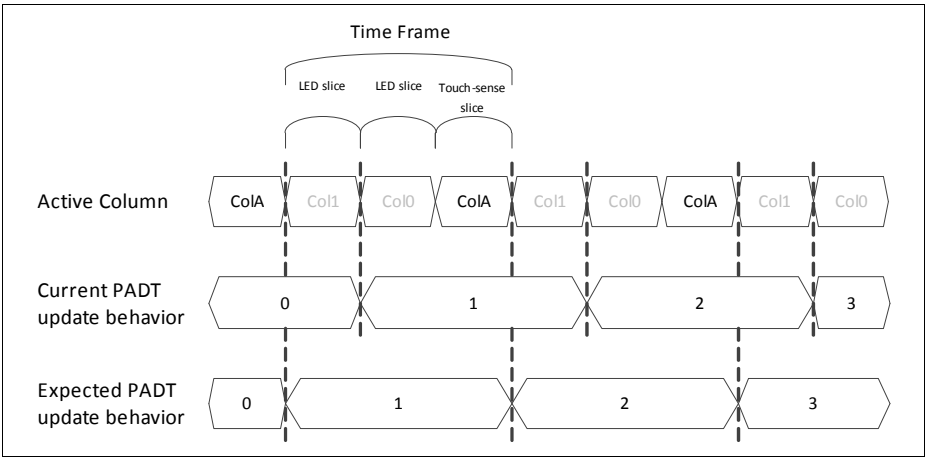

Figure 13 PADT update behavior

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 14**). In addition to this, when no LEDs are enabled, pad turn 0 will prevail for one time slice longer before it gets updated (**Figure 15**).

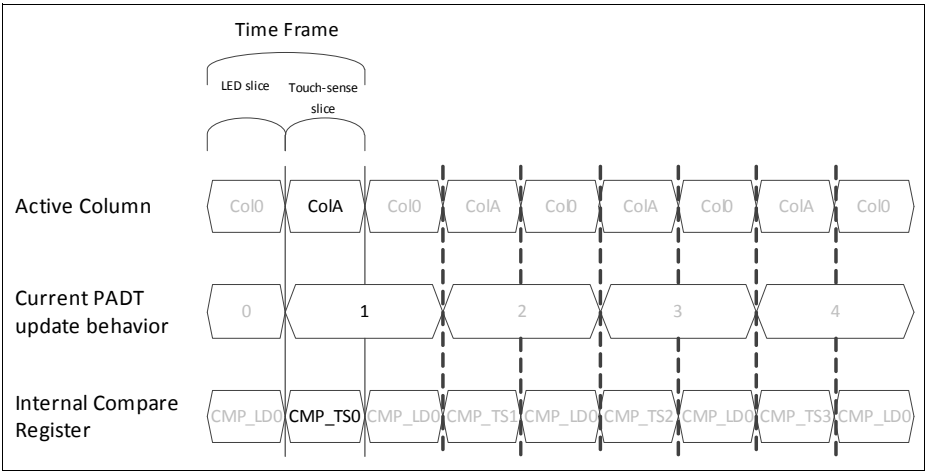

Figure 14 Effect of delay on the update of Internal Compare Register with 1 LED column enabled

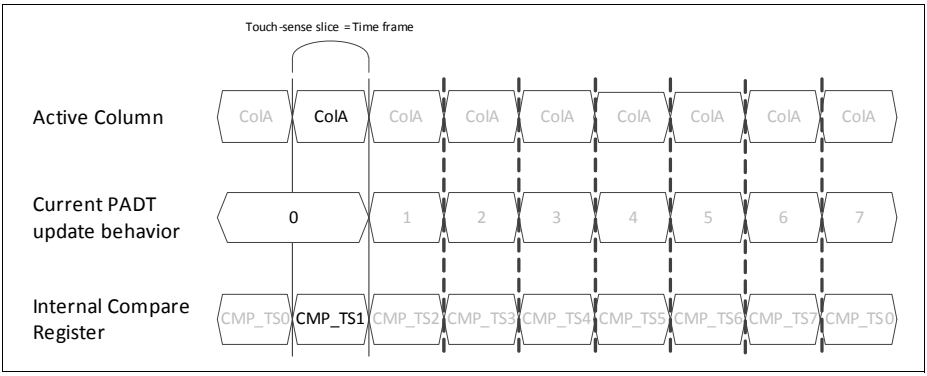

Figure 15 Pad turn 0 prevails for one time slice longer when no LEDs are enabled

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 16**).

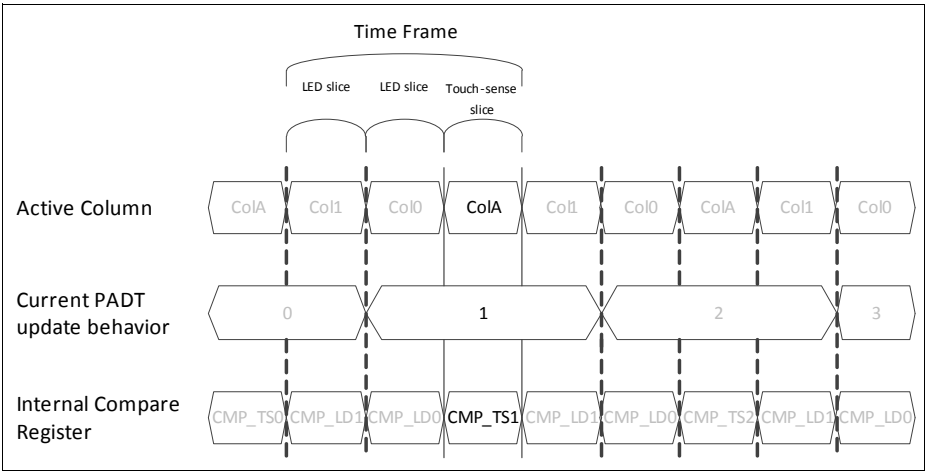

Figure 16 Internal Compare Register updated with correct compare register value with 2 LED columns enabled

Conditions

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

Workaround

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

Workaround for reading PADT

Due to the delay in the PADT update, the user would get the current active pad turn when PADT is read in the time frame interrupt. However, this PADT value read differs when read in a time slice interrupt. This depends on the number of LED columns enabled and the active function or LED column in the previous time slice (**Table 12**). The bit field FNCTL.FNCOL provides a way of interpreting the active function or LED column in the previous time slice.

Table 12 PADT value as read in the time slice interrupt

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

Workaround for aligning CMP_TSx

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

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

1. At the end of the time frame interrupt service routine, software can prepare for the next active pad turn by programming the CMP_TS[x-1] bit field with the intended compare value for TSIN[x]. As an example, if the next active pad is TSIN[2], program CMP_TS[1] with the compare value intended for TSIN[2] (**Figure 17**).

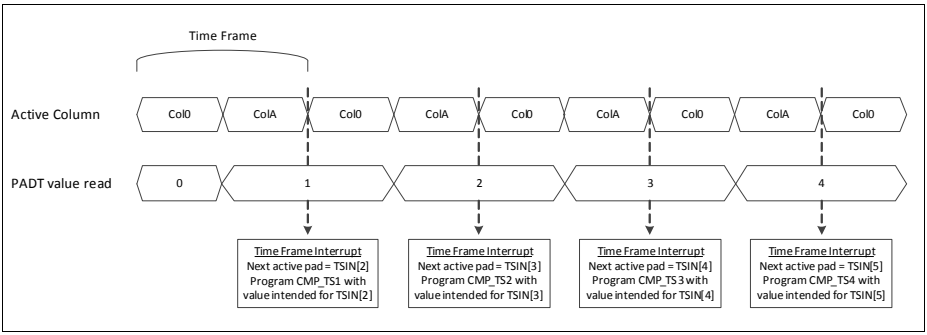

Figure 17 Software workaround demonstration

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

PARITY_CM.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.

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 13 PORTS registers reset values

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

1) Only in XMC4500 devices.

2) Device and package dependent

Implications

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

With the different value in Pn_PDISC of the digital ports the availability of digital pins in a device can no longer be verified via this register. Note that Pn_PDISC of pure digital ports is read-only; user software can't write to them.

The Pn_PDISC of the shared analog/digital port pins (P14 and P15) enables/disables the digital input path. After a system reset this path is disabled, after a module reset enabled. User software in Priviledged Mode can reconfigure them as needed by the application.

The different value in the Pn_PDR registers configures output port pins with a "Strong-Sharp" output driver mode, as opposed to "Strong-Soft" driver mode after a system reset. This may result in a higher current consumption and more noise induced to the external system. User software in Priviledged Mode can reconfigure them as needed by the application.

Subject to Agreement on the Use of Product Information

Workaround

None.

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.

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

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

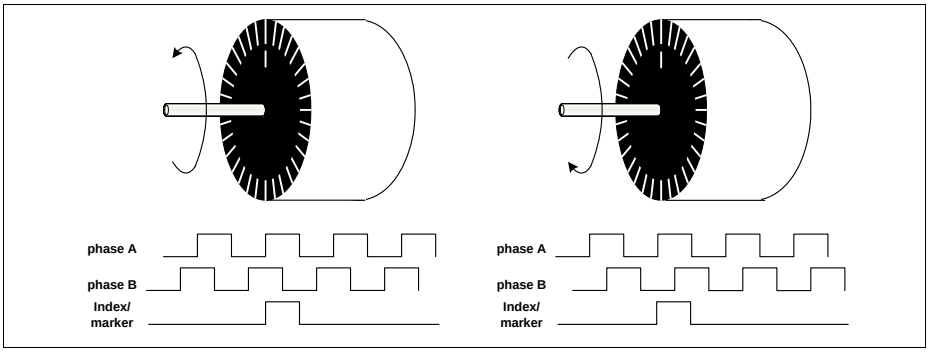

Figure 18 Rotary Encoder outputs - Phase A, Phase B and Index

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

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

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

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

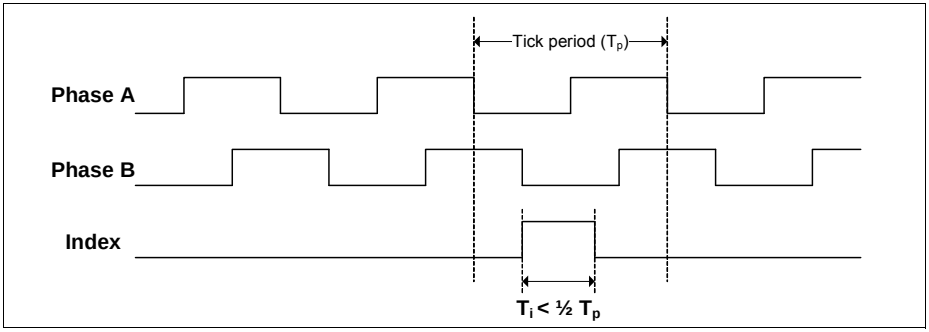

Figure 19 Different index signal types

Workaround

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

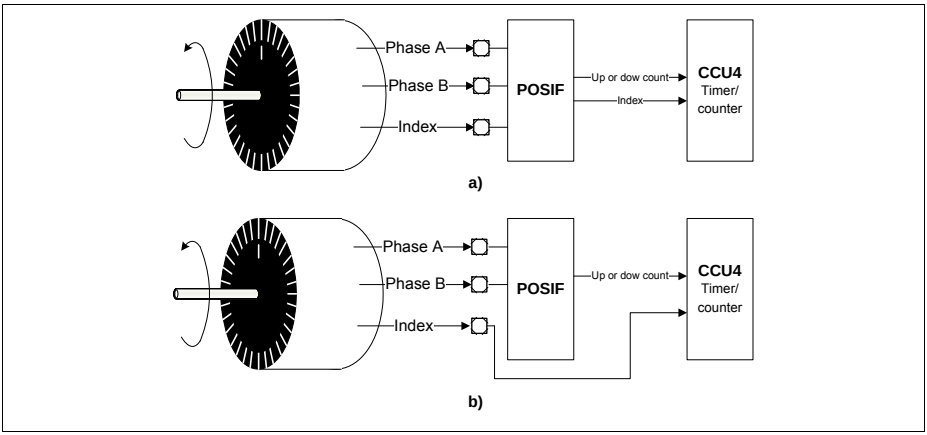

Figure 20 Index usage workaround - a) Non working solution; b) Working solution

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

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

Implications

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

Workaround

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

SCU_CM.015 Functionality of parity memory test function limited

The device provides an interface to access the parity bits of the contained SRAM memories for test purpose. This feature is typically used by safety applications which must ensure that the parity mechanism is operational.

The test interface is based on using SCU registers PMTPR, PMTSR and MCHKCON. By those registers it is possible to implant (write) user defined parity bits to selected memory cells. For checking of the parity value a read function is defined.

Due to synchronization issues wrong results can be produced for the PMTPR.PRD read value.

Implications

The values read back by PMTPR.PRD can be incorrect. Therefore it is not possible to directly check the parity information. Testing for the correct function of the parity logic is still possible by directed activation of parity errors.

Test of parity function

It is possible to test the correct function of the parity logic of PSRAM, DSRAM, USIC, CAN and USB memories using the following scheme:

1. Enable parity error generation and parity error trap generation using registers PEEN and PETE
2. Enable one target for parity test via registers PMTSR and MCHKCON
3. Write parity value for memory test to register bit field PMTPR.PWR
4. Write data value whose parity values conflicts with the parity value written in step 3 to memory location (@address0)
5. For PSRAM and DSRAM only: a 2nd write operation to another memory location (@address1) is required to flush the write buffer from step 4
6. Read back the content from the first memory location (@address0)
7. Parity error and NMI trap occurrence is expected

The NMI handler should check on the right content of the registers PEFLAG and TRAPSTAT and clear the related parity and NMI trap flags before returning.

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

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

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

Implications

If affected hardware requests (see SRRAW column in **Table 14**) 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 14**. 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 14 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

Table 14 Request source and related SRRAW register bit field (cont'd)

Request Source		SRRAW
Module	**Bit field**	**Bit field**
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

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

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

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

Implications

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

Workaround

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

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 not 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.

USIC_AI.008 SSC delay compensation feature cannot be used

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

Workaround

None.

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

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

Table 15

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

Workaround

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

To generate the master slave select signal:

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

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

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

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

Workaround

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

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

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

Workaround

None.

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

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

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

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

Workaround

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

USIC_AI.016 Transmit parameters are updated during FIFO buffer bypass

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

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

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

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

Workaround

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

USIC_AI.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.

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

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

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

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

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

Workaround

There are two possible workarounds:

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

USIC_AI.019 First data word received by IIC receiver triggers RIF instead of AIF

When operating in IIC mode as a master or slave receiver, the first data word received following a start condition and address match triggers a receive event (indicated by PSR.RIF flag) instead of an alternate receive event (indicated by PSR.AIF flag).

Workaround

To determine if a received data word is the first word of a new frame, bit 9 of RBUF needs to be read:

- When RBUF[9] is 1, the first data word of a new frame is indicated;
- When RBUF[9] is 0, subsequent data words of the frame are indicated.

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.

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 an 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 Deviations from Electrical- and Timing Specification

The errata in this section describe deviations from the documented electrical- and timing specifications.

POWER_CM.P001 Risk of increased current consumption in internally controlled hibernate mode

When

- the device is in internally controlled hibernate mode and
- if a voltage difference of (V_{DDP} - V_{BAT}) > 1 V is present

then I_{DDPH} may be up to 50 µA higher than specified.

Recommendation

When the device is in internally controlled hibernate mode it must be ensured that V_{BAT} level is maintained close to the level of V_{DDP} .

For details please refer to the “System Level Integration” examples of the “Hibernate Control” section in the Reference Manual.

POWER_CM.P003 Current consumption when executing from PSRAM

The Data Sheet defines a typical current consumption for execution from PSRAM and Flash in sleep mode.

Additional measurements have shown that the actual values can exceed the current values defined in the Data Sheet. The table below lists the updated values.

Subject to Agreement on the Use of Product Information

Table 16 Power Supply Parameters

Parameter	Symbol	Values			Unit	Note / Test Condition
		Min	Typ	Max		
Active supply current Code execution from RAM Flash in Sleep mode Frequency: f_{CPU} / f_{PERIPH} / f_{CCU} in MHz	I_{DDPA} CC	–	78	–	mA	120 / 120 / 120
		–	67	–	mA	120 / 60 / 60

POWER_CM.P004 Current consumption while $\overline{\text{PORST}}$ low can exceed specified value

The Data Sheet defines a maximum current consumption while the device is held in reset via $\overline{\text{PORST}}$, I_{DDP_PORST}.

Additional measurements have shown that the actual values can exceed the current values defined in the Data Sheet. The table below lists the updated values.

Table 17 Power Supply Parameters

Parameter	Symbol	Values			Unit	Note / Test Condition
		Min	Typ	Max		
I_{DDP} current at $\overline{\text{PORST}}$ Low	I_{DDP_PORST} CC	–	5	7	mA	V_{DDP} = 3.3 V, T_A = 25 °C
		–	14	45	mA	V_{DDP} = 3.6 V, T_J= 150 °C

4 Application Hints

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

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

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

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

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

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

   with

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

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

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

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

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

Recommendation 1

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

Recommendation 2

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

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

ADC_AI.H004 Completion of Startup Calibration

Before using the VADC the startup calibration must be completed.

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

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

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

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

ADC_AI.H008 Injected conversion with broken wire detection

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

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

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

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

Recommendation

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

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

ADC_TC.H011 Bit DCMSB in register GLOBCFG

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

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

Subject to Agreement on the Use of Product Information

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.

MultiCAN_AI.H005 TxD Pulse upon short disable request

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

Example for setup of the CAN disable request:

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

Workaround

Set all INIT bits to 1 before requesting module disable.

MultiCAN_AI.H006 Time stamp influenced by resynchronization

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

shorter or longer than nominal bit time length due to the CAN resynchronization events.

Workaround

None.

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

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

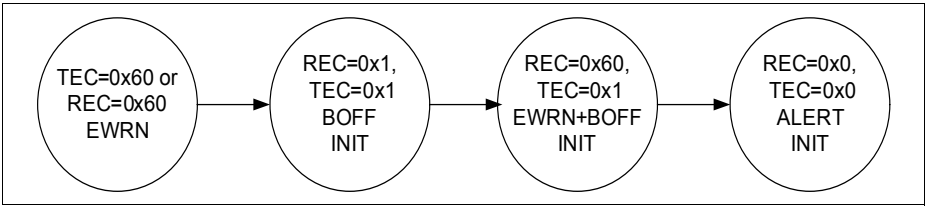

Figure 21 Alert Interrupt Behavior in case of Bus-Off

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

MultiCAN_AI.H008 Effect of CANDIS on SUSACK

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

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

MultiCAN_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**.

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

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

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

Workaround

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

MultiCAN_TC.H004 Double remote request

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

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

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

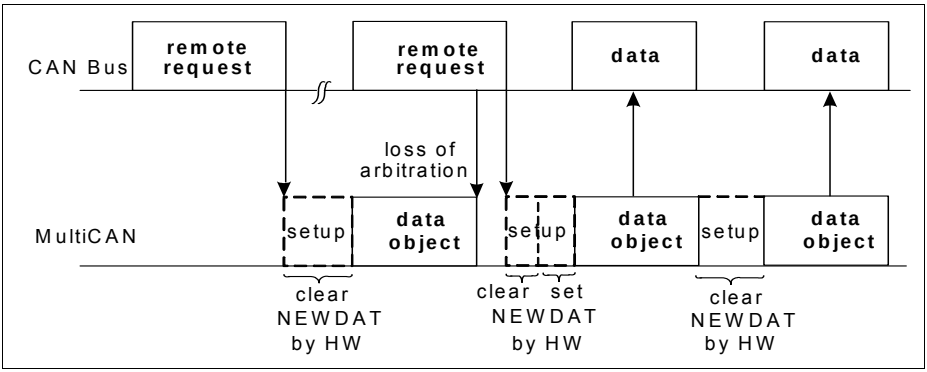

Figure 22 Loss of Arbitration

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 to optimize 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 3.3V LVTTL outputs)
- "Class A1+ pads (medium speed 3.3V LVTTL outputs)
- "Class A2 pads (high speed 3.3V LVTTL outputs, e.g. for EBU or fast serial interfaces)

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

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 25MHz.

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 80MHz.

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 ohms) 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 25MHz signal strong/medium or strong/soft would be suitable lowering the potential overshoots and undershoots.

RESET_CM.H001 Power-on reset release

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

Implications

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

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

Subject to Agreement on the Use of Product Information

Recommendation

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

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

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.

5 Documentation Updates

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

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

Implications

The reference manual is limiting 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.

STARTUP_CM.D003 Alignment of ABM/PSRAM Header

The reference manual is specifying PSRAM/ABM Header of 32 byte size. Only for 20byte of these 32 bytes the functionality is defined. The remaining 12bytes 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 following figure provides a detailed view on the location of the reserved bytes:

Subject to Agreement on the Use of Product Information

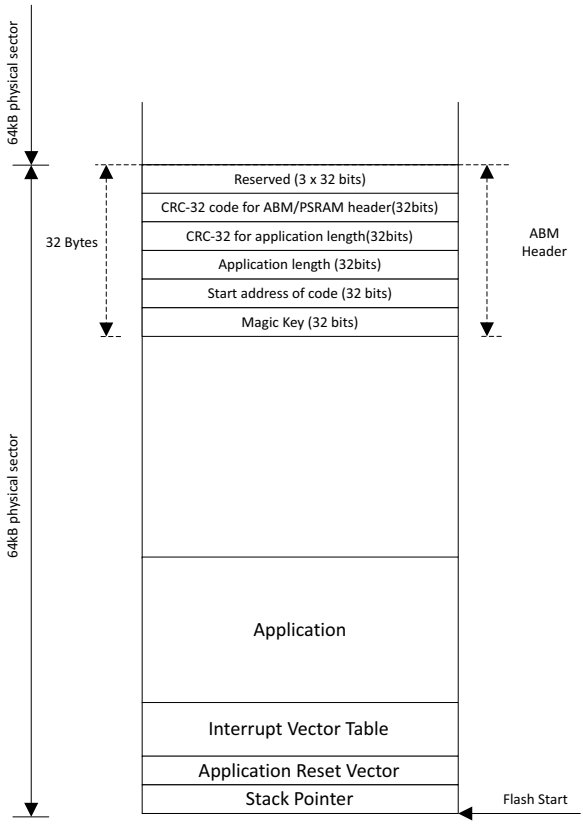

Figure 23 ABM/PSRAM Header - Location of reserved bytes

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.