Never stop thinking

16-Bit

Architecture

XE166N Derivatives

16-Bit Single-Chip
Real Time Signal Controller
XE166 Family / Value Line

Errata Sheet

V1.3 2009-06

Microcontrollers

Edition 2009-06

Published by
Infineon Technologies AG
81726 Munich, Germany

Never stop thinking

16-Bit

Architecture

XE166N Derivatives

16-Bit Single-Chip
Real Time Signal Controller
XE166 Family / Value Line

Errata Sheet

V1.3 2009-06

Microcontrollers

Table of Contents

1 History List / Change Summary

Table 1 History List

Version	Date	Remark[1)]
1.0	30.09.2008	First Errata Sheet release
1.1	30.01.2009	Errata Sheet name is changed from "XE166**x**N Derivatives" to "XE166N Derivatives"
1.2	10.03.2009	New Marking/Step, new Errata Sheet layout
1.3	08.06.2009	Errata No. 01489AERRA, new Marking/Step AA

1) Errata changes to the previous Errata Sheet are marked in **Chapter 5** "**Short Errata Description**".

Trademarks

C166^{TM}, TriCore^{TM} and DAVE^{TM} are trademarks of Infineon Technologies AG.

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing?
Your feedback will help us to continuously improve the quality of this document.
Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

2 General

This Errata Sheet describes the deviations of the XE166N Derivatives from the current user documentation.

Each erratum identifier follows the pattern **Module_Arch.TypeNumber**:

- **Module**: subsystem, peripheral, or function affected by the erratum
- **Arch**: microcontroller architecture where the erratum was firstly detected.
  - **AI**: Architecture Independent
  - **CIC**: Companion ICs
  - **TC**: TriCore
  - **X**: XC166 / XE166 / XC2000 Family
  - **XC8**: XC800 Family
  - **[none]**: C166 Family
- **Type**: category of deviation
  - **[none]**: Functional Deviation
  - **P**: Parametric Deviation
  - **H**: Application Hint
  - **D**: Documentation Update
- **Number**: ascending sequential number within the three previous fields. As this sequence is used over several derivatives, including already solved deviations, gaps inside this enumeration can occur.

This Errata Sheet applies to all temperature and frequency versions and to all memory size variants of this device, unless explicitly noted otherwise.

Note: This device is equipped with a C166S V2 Core. Some of the errata have workarounds which are possibly supported by the tool vendors.
Some corresponding compiler switches need possibly to be set. Please see the respective documentation of your compiler.
For effects of issues related to the on-chip debug system, see also the documentation of the debug tool vendor.

3 Current Documentation

The Infineon XE166 Family comprises device types from the XE162x Series and the XE164x Series.

Device	XE16xxN
Marking/Step	EES-AA, ES-AA, AA
Package	PG-LQFP-64, PG-LQFP-100

This Errata Sheet refers to the following documentation:

- XE166N Derivatives User's Manual
- XE162xN Data Sheet
- XE164xN Data Sheet
- Documentation Addendum (if applicable)

Make sure you always use the corresponding documentation for this device available in category 'Documents' at **www.infineon.com/xe166**.

The specific test conditions for EES and ES are documented in a separate Status Sheet.

Note: Devices marked with EES or ES are engineering samples which may not be completely tested in all functional and electrical characteristics, therefore they should be used for evaluation only.

4 Errata Device Overview

This chapter gives an overview of the dependencies of individual errata to devices and steps. An **X** in the column of the sales codes shows that this erratum is valid.

4.1 Functional Deviations

Table 2 shows the dependencies of functional deviations in the derivatives.

Table 2 Errata Device Overview: Functional Deviations

Functional Deviation	XE16xxN	
	EES-AA ES-AA	AA[1)]
BSL_CAN_X.001	X	X
ECC_X.002	X	
ESR_X.002	X	X
GPT12E_X.002	X	X
OCDS_X.003	X	X
PARITY_X.001	X	X
RESET_X.003	X	X
USIC_AI.004	X	X
WDT_X.002	X	X

1) From EES/ES-AA step to AA step, 2 errata have been fixed.

4.2 Deviations from Electrical and Timing Specification

Table 3 shows the dependencies of deviations from the electrical and timing specification in the derivatives.

Table 3 Errata Device Overview: Deviations from Electrical and Timing Specification

AC/DC/ADC Deviation	XE16xxN	
	EES-AA ES-AA	AA[1)]
SWD_X.P001	X	

1) From EES/ES-AA step to AA step, 2 errata have been fixed.

4.3 Application Hints

Table 4 shows the dependencies of application hints in the derivatives.

Table 4 Errata Device Overview: Application Hints

Hint	XE16xxN	
	EES-AA ES-AA	AA[1)]
CAPCOM12_X.H001	X	X
CC6_X.H001	X	X
ECC_X.H001	X	X
GPT12E_X.H002	X	X
INT_X.H002	X	X
MultiCAN_AI.H005	X	X
MultiCAN_AI.H006	X	X
MultiCAN_TC.H002	X	X
MultiCAN_TC.H003	X	X
MultiCAN_TC.H004	X	X
RESET_X.H003	X	X
RTC_X.H003	X	X
USIC_AI.H001	X	X

1) From EES/ES-AA step to AA step, 2 errata have been fixed.

5 Short Errata Description

This chapter gives an overview on the deviations and application hints. Changes to the last Errata Sheet are shown in the column “Chg”.

5.1 Functional Deviations

Table 5 shows a short description of the functional deviations.

Table 5 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
BSL_CAN_X.001	Quartz Crystal Settling Time after PORST too Long for CAN Bootstrap Loader	New	14
ECC_X.002	Incorrect ECC Error Indication for DPRAM		14
ESR_X.002	ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access	Up-date	15
GPT12E_X.002	Effects of GPT Module Microarchitecture		16
OCDS_X.003	Peripheral Debug Mode Settings cleared by Reset		17
PARITY_X.001	PMTSR Register Initialization		18
RESET_X.003	P2.[2:0] and P10.[12:0] Switch to Input		18
USIC_AI.004	Receive shifter baudrate limitation		19
WDT_X.002	Clearing the Internal Flag which Stores Preceding WDT Reset Request		19

5.2 Deviations from Electrical and Timing Specification

Table 6 shows a short description of the electrical- and timing deviations from the specification.

Table 6 Deviations from Electrical and Timing Specification

AC/DC/ADC Deviation	Short Description	Chg	Pg
SWD_X.P001	Supply Watchdog Level V_{SWD_min} too Low		21

5.3 Application Hints

The Table 7 shows a short description of the application hints.

Table 7 Application Hints

Hint	Short Description	Chg	Pg
CAPCOM12_X.H001	Enabling or Disabling Single Event Operation		22
CC6_X.H001	Modifications of Bit MODEN in Register CCU6x_KSCFG		23
ECC_X.H001	ECC Error Indication Permanently Set		23
GPT12E_X.H002	Reading of Concatenated Timers		24
INT_X.H002	Increased Latency for Hardware Traps		25
MultiCAN_AI.H005	TxD Pulse upon short disable request		26
MultiCAN_AI.H006	Time stamp influenced by resynchronization		26
MultiCAN_TC.H002	Double Synchronization of receive input		26
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		27
MultiCAN_TC.H004	Double remote request		27
RESET_X.H003	How to Trigger a PORST after an Internal Failure		28
RTC_X.H003	Changing the RTC Configuration		28
USIC_AI.H001	FIFO RAM Parity Error Handling		29

6 Detailed Errata Description

This chapter provides a detailed description for each erratum. If applicable a workaround is suggested.

6.1 Functional Deviations

BSL_CAN_X.001 Quartz Crystal Settling Time after PORST too Long for CAN Bootstrap Loader

The startup configuration of the CAN bootstrap loader when called immediately after $\overline{\text{PORST}}$ limits the settling time of the external oscillation to 0.5 ms. For typical quartz crystal this settling time is too short. The CAN bootstrap loader generates a time-out and goes into Startup Error State.

Workaround

- For low performance CAN application a ceramic resonator with settling time less then 0.5 ms can be used.
- An alternative is the Internal Start from on-chip Flash memory as startup mode after $\overline{\text{PORST}}$. Then switch the system clock to external source and trigger a software reset with CAN bootstrap loader mode selected. Now the device starts with a CAN bootstrap loader without limitation of the oscillator settling time.

ECC_X.002 Incorrect ECC Error Indication for DPRAM

Under certain conditions, the ECC error flag for the dual port memory (DPRAM) may indicate an error when none exists.
Conditions under which ECC error is incorrectly indicated:

- ECC memory protection has been selected for DPRAM (`SCU_MCHKCON.SELDP` = 0).
- The ECC check is enabled for DPRAM (`SCU_ECCCON.DPEN` = 1).

- A concurrent read and write access is made to the same byte or word in the DPRAM (possible due to instruction pipeline).

Under the above conditions, an ECC error may be indicated for DPRAM (flag `SCU_ECCSTAT.DP` = 1), although there is no error in DPRAM.

It should be noted that despite the incorrect ECC error indication, the data are delivered error free, and the ECC logic still functions correctly (correcting data single bit errors, if present).

There is no data corruption in this case. The ECC bits that were written are generated correctly, as well as check and correction is not affected for that was read.

This problem is limited to the DPRAM. All other SRAM memories cannot perform concurrent read and write accesses, and therefore cannot have this issue.

Workaround

Use parity protection for DPRAM (`SCU_MCHKCON.SELDP` = 1).

Single-bit errors will be correctly indicated by parity logic, but will not be corrected.

ESR_X.002 ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access

During a write access to any register, bits in registers `ESREXSTAT1/2` can be cleared inadvertently.

`ESREXSTAT1/2` store event(s) that can trigger various ESR functions.

Workaround

Make sure that the trigger signals are still active when the associated service routine runs, so the trigger source can be evaluated by software.

GPT12E_X.002 Effects of GPT Module Microarchitecture

The present GPT module implementation provides some enhanced features (e.g. block prescalers `BPS1`, `BPS2`) while still maintaining timing and functional compatibility with the original implementation in the C166 Family of microcontrollers.

Both the GPT1 and GPT2 blocks use a finite state machine to control the actions within each block. Since multiple interactions are possible between the timers (T2 .. T6) and register CAPREL, these elements are processed sequentially within each block in different states. However, all actions are normally completed within one basic clock cycle.

The GPT2 state machine has 4 states (2 states when `BPS2` = 01_B) and processes T6 before T5. The GPT1 state machine has 8 states (4 states when `BPS1` = 01_B) and processes the timers in the order T3 - T2 (all actions except capture) - T4 - T2 (capture).

In the following, two effects of the internal module microarchitecture that may require special consideration in an application are described in more detail.

1.) Reading T3 by Software with T2/T4 in Reload Mode

When T2 or T4 are used to reload T3 on overflow/underflow, and T3 is read by software on the fly, the following unexpected values may be read from T3:

- when T3 is counting **up**, 0000_H or 0001_H may be read from T3 directly after an overflow, although the reload value in T2/T4 is higher (0001_H may be read in particular if `BPS1` = 01_B and `T3I` = 000_B),
- when T3 is counting **down**, $FFFF_H$ or $FFFE_H$ may be read from T3 directly after an underflow, although the reload value in T2/T4 is lower ($FFFE_H$ may be read in particular if `BPS1` = 01_B and `T3I` = 000_B).

Note: All timings derived from T3 in this configuration (e.g. distance between interrupt requests, PWM waveform on T3OUT, etc.) are accurate except for the specific case described under 2.) below.

Workaround:

- When T3 counts **up**, and value_x < reload value is read from T3, value_x should be replaced with the reload value for further calculations.

- When T3 counts **down**, and value_x > reload value is read from T3, value_x should be replaced with the reload value for further calculations.

Alternatively, if the intention is to identify the overflow/underflow of T3, the T3 interrupt request may be used.

2.) Reload of T3 from T2 with setting `BPS1` = 01_B and `T3I` = 000_B

When T2 is used to reload T3 in the configuration with `BPS1` = 01_B and `T3I` = 000_B (i.e. fastest configuration/highest resolution of T3), the reload of T3 is performed with a delay of one basic clock cycle.

Workaround 1:

To compensate the delay and achieve correct timing,

- increment the reload value in T2 by 1 when T3 is configured to count **up,**
- decrement the reload value in T2 by 1 when T3 is configured to count **down**.

Workaround 2:

Alternatively, use T4 instead of T2 as reload register for T3. In this configuration the reload of T3 is not delayed, i.e. the effect described above does not occur with T4.

<u>OCDS_X.003</u> Peripheral Debug Mode Settings cleared by Reset

The behavior (run/stop) of the peripheral modules in debug mode is defined in bitfield SUMCFG in the KSCCFG registers. The intended behavior is, that after an application reset has occurred during a debug session, a peripheral re-enters the mode defined for debug mode.

For some peripherals, the debug mode setting in SUMCFG is erroneously set to normal mode upon any reset (instead upon a debug reset only). It remains in this state until SUMCFG is written by software or the debug system.

Some peripherals will **not** re-enter the state defined for debug mode after an application reset:

GPT12, CAPCOM2, and MultiCAN will resume normal operation like after reset, i.e. they are inactive until they are initialized by software.

In case the **RTC** has been running before entry into debug mode, and it was configured in SUMCFG to stop in debug mode, it will resume operation as before entry into debug mode instead.

All other peripheral modules, i.e. ADC0, ADC1, CC60...CC63, USIC0...USIC2, will correctly re-enter the state defined for debug mode after an application reset in debug mode.

For **Flash** and **CPU**, bitfield SUMCFG must be configured to normal mode anyway, since they are required for debugging.

Workaround

None.

PARITY_X.001 `PMTSR` Register Initialization

The `PMTSR` register content after start-up is 0100_H, meaning the parity logic for SBRAM is not in standard mode of operation.

Workaround

If parity will be used as Memory Control mechanism for SBRAM, it must be enabled by initializing the `PMTSR` register with 8000_H.

RESET_X.003 `P2.[2:0]` and `P10.[12:0]` Switch to Input

During the execution of an Application Reset and Debug Reset the pins `P2.[2:0]` and `P10.[12:0]` are intermediately switched to input.

These pins return to their previous mode approximately 35 system clock cycles after the application reset counter has expired (approx. 0.6 µs with default reset delay at 80 MHz).

If such a pin is used as output, make sure that this short interruption does not lead to critical system conditions.

Workaround

External pull devices can be added to have a defined level on these pins during Application and Debug Reset.

USIC_AI.004 Receive shifter baudrate limitation

If the frame length of `SCTRH.FLE` does not match the frame length of the master, then the baudrate of the SSC slave receiver is limited to $f_{sys}/2$ instead of f_{sys}.

Workaround

None.

WDT_X.002 Clearing the Internal Flag which Stores Preceding WDT Reset Request

The information that the WDT has already been exceeded once is stored in an internal flag. In contrary to the documentation, that this flag can be cleared by writing a 1_B to bit `WDTCS.CLRIRF` at any time, clearing of the internal flag is only possible, when the WDT is in Prewarning Mode.

Workaround 1

Applications following the proposal of Application Note **AP16103** (section `Using ESR pins to trigger a PORST reset`) to trigger a Power-on Reset upon a WDT event will find the internal flag cleared upon the Power-on Reset and thus will have no issue with this limitation.

Workaround 2

In case the WDT triggers a User Reset upon a WDT overflow, the internal flag will not be cleared by the reset itself. Any further overflow of the WDT will lead to a permanent reset of the device.

Applications which intentionally let the WDT exceed once, e.g. in conjunction with an initial self test, might want to have the internal flag cleared to prevent a permanent reset upon a real WDT overflow.

If the internal flag shall be cleared by software, this must be done as a reaction on a WDT overflow in the time frame the WDT is in Prewarning Mode before the permanent User Reset will be triggered. The CPU is notified upon the WDT entering Prewarning Mode by issuing an interrupt request. The application can react on this request and clear the internal flag now by writing a 1_B to bit `WDTCS.CLRIRF` e.g. within an ISR.

Workaround 3

Some applications may not want to use or rely on the interrupt logic in conjunction with a WDT overflow event. The proposed remedy in this case is, to initiate a Power Reset to clear the internal flag by changing the settings of the active Supply Watchdog (SWD) as follows:

1. Disable SFR protection.
2. Write the inverted value of bit `LxALEV` to register `SWDCON0`, where x stands for the number of the comparator which currently would trigger a Power Reset.

In doing so, a Power Reset for VDDI_1 and VDDI_M will be activated clearing the internal flag. The application may store information on preceding WDT events in the Standby-SRAM. This can be done any time after the WDT reset without timing limitations or the need to use the interrupt logic.

Note: Although the supply for the DPRAM, DSRAM and PSRAM will be switched off during the active reset phase, it depends on the external buffer capacitance at the VDDI_1 pins, the actual system clock frequency and the environmental conditions, whether the content of these RAMs will be preserved in this case or not. However, the Standby-RAM itself is not cleared upon this reset.

6.2 Deviations from Electrical and Timing Specification

SWD_X.P001 Supply Watchdog Level V_{SWD_min} too Low

The supply watchdog (SWD) has a built-in hysteresis. In the affected products, this hysteresis is increased.

This leads to a decreased lower level, i.e. the threshold is lower than selected (e.g. < 4.5 V for `SWDCON.LEVxV` = 1001_B, or < 3.0 V for `SWDCON.LEVxV` = 0001_B).

The functionality of the on-chip modules is not affected, as it is ensured by the power validation circuits (PVC).

The IO timing can be marginally slower if V_{DDP} is below the specified minimum value.

Workaround

None.

6.3 Application Hints

CAPCOM12_X.H001 Enabling or Disabling Single Event Operation

The single event operation mode of the CAPCOM1/2 unit eliminates the need for software to react after the first compare match when only one event is required within a certain time frame. The enable bit SEEy for a channel CCy is cleared by hardware after the compare event, thus disabling further events for this channel.

One Channel in Single Event Operation

As the Single Event Enable registers CC1_SEE, CC2_SEE are not located in the bit-addressable SFR address range, they can only be modified by instructions operating on data type WORD. This is no problem when only one channel of a CAPCOM unit is used in single event mode.

Two or more Channels in Single Event Operation

When two or more channels of a CAPCOM unit are independently operating in single event mode, usually an OR instruction is used to enable one or more compare events in register CCn_SEE, while an AND instruction may be used to disable events before they have occurred. In these cases, the timing relation of the channels must be considered, otherwise the following typical problem may occur:

- In the Memory stage, software reads register CCn_SEE with bit SEEy = 1_B (event for channel CCy has not yet occurred)
- Meanwhile, event for CCy occurs, and bit SEEy is cleared to 0_B by hardware
- In the Write-Back stage, software writes CCn_SEE with bit SEEx = 1_B (intended event for CCx enabled via OR instruction) **and** bit SEEy = 1_B
- or, as inverse procedure, software writes CCn_SEE with bit SEEx = 0_B (intended event for CCx disabled via AND instruction) **and** bit SEEy = 1_B

In these cases, another unintended event for channel CCy is enabled.

To avoid this effect, one of the following solutions - depending on the characteristics of the application - is recommended to enable or disable further compare events for CAPCOM channels concurrently operating in single event mode:

- Modify register CCn_SEE only when it is ensured that no compare event in single event mode can occur, i.e. when CCn_SEE = 0x0000, or
- Modify register CCn_SEE only when it is ensured that there is a sufficient time distance to the events of all channels operating in single event mode, such that none of the bits in CCn_SEE can change in the meantime, or
- Use single event operation for one channel only (i.e. only one bit SEMx may be = 1_B), and/or
- Use one of the standard compare modes, and emulate single event operation for a channel CCs by disabling further compare events in bit field MODs (in register CCn_Mz) in the corresponding interrupt service routine. Writing to register CCn_Mz is uncritical, as this register is not modified by hardware.

CC6_X.H001 Modifications of Bit MODEN in Register CCU6x_KSCFG

For each module, setting bit MODEN = 0 immediately switches off the module clock. Care must be taken that the module clock is only switched off when the module is in a defined state (e.g. stop mode) in order to avoid undesired effects in an application.

In addition, for a CCU6 module in particular, if bit MODEN is changed to 0 while the internal functional blocks have not reached their defined stop conditions, and later MODEN is set to 1 and the mode is not set to run mode, this leads to a lock situation where the module clock is not switched on again.

ECC_X.H001 ECC Error Indication Permanently Set

The ECC error flag of the ECCSTAT register for the DPRAM, DSRAM, PSRAM and SBRAM can not be cleared, if a memory location with an ECC error is selected and the ECC is enabled. The memory can be selected by an active or by the latest read or write access.

Workaround

Select a memory location without ECC error in the respective memory (e.g. make a read to another address) and then clear the ECC error flag. Be aware that the new selected address may also have an ECC error.

GPT12E_X.H002 Reading of Concatenated Timers

For measuring longer time periods, a core timer (T3 or T6) may be concatenated with an auxiliary timer (T2/T4 or T5) of the same timer block. In this case, the core timer contains the low part, and the auxiliary timer contains the high part of the extended timer value.

When reading the low and high parts of concatenated timers, care must be taken to obtain consistent values in particular after a timer overflow/underflow (e.g. one part may already have considered an overflow, while the other has not). This is a general issue when reading multi-word results with consecutive instructions, and not necessarily unique to the GPT module microarchitecture.

The following algorithm may be used to read concatenated GPT timers, represented by Timer_high (for auxiliary timer, here: T2) and Timer_low (for core timer, here: T3). In this example, the high part is read twice, and reading of the low part is repeated if two different values were read for the high part.

- read Timer_high_temp = T2
- read Timer_low = T3
- wait two basic clock cycles (to allow increment/decrement of auxiliary timer in case of core timer overflow/underflow) - see **Table 8** below
- read Timer_high = T2
  - if Timer_high is not equal to Timer_high_temp: read Timer_low = T3

After execution of this algorithm, Timer_high and Timer_low represent a consistent time stamp of the concatenated timers.

The equivalent number of system clock cycles corresponding to two basic clock cycles is shown in the following **Table 8**:

Table 8 Equivalent Number of System Clock Cycles Required to Wait for Two Basic Clock Cycles

Setting of BPS1	**BPS1 = 01**	**BPS1 = 00**	**BPS1 = 11**	**BPS1 = 10**
Required Number of System Clocks	8	16	32	64
Setting of BPS2	**BPS2 = 01**	**BPS2 = 00**	**BPS2 = 11**	**BPS2 = 10**
Required Number of System Clocks	4	8	16	32

In case the required timer resolution can be achieved with different combinations of the Block Prescaler BPS1/BPS2 and the Individual Prescalers TxI, the variant with the smallest value for the Block Prescaler may be chosen to minimize the waiting time. E.g. in order to run T6 at $f_{SYS}/512$, select BPS2 = 00_B, T6I = 111_B, and insert 8 NOPs (or other instructions) to ensure the required waiting time before reading Timer_high the second time.

INT_X.H002 Increased Latency for Hardware Traps

When a condition for a HW trap occurs (i.e. one of the bits in register TFR is set to 1_B), the next valid instruction that reaches the Memory stage is replaced with the corresponding TRAP instruction. In some special situations described in the following, a valid instruction may not immediately be available at the Memory stage, resulting in an increased delay in the reaction to the trap request:

1. When the CPU is in break mode, e.g. single-stepping over such instructions as SBRK or BSET TFR.x (where x = one of the trap flags in register TFR) will have no (immediate) effect until the next instruction enters the Memory stage of the pipeline (i.e. until a further single-step is performed).
2. When the pipeline is running empty due to (mispredicted) branches and a relatively slow program memory (with many wait states), servicing of the trap is delayed by the time for the next access to this program memory, even if vector table and trap handler are located in a faster memory. However, the situation when the pipeline/prefetcher are completely empty is quite rare due to the advanced prefetch mechanism of the C166S V2 core.

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:

`MCAN_KSCCFG.MODEN` = 0 and then `MCAN_KSCCFG.MODEN` = 1

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_TC.H002 Double Synchronization of receive input

The MultiCAN module has a double synchronization stage on the CAN receive inputs. This double synchronization, delays the receive data by 2 module clock cycles. If the MultiCAN is operating at a low module clock frequencies and high CAN baudrate, this delay may become significant and has to be taken into account when calculating the overall physical delay on the CAN bus (transceiver delay...).

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.

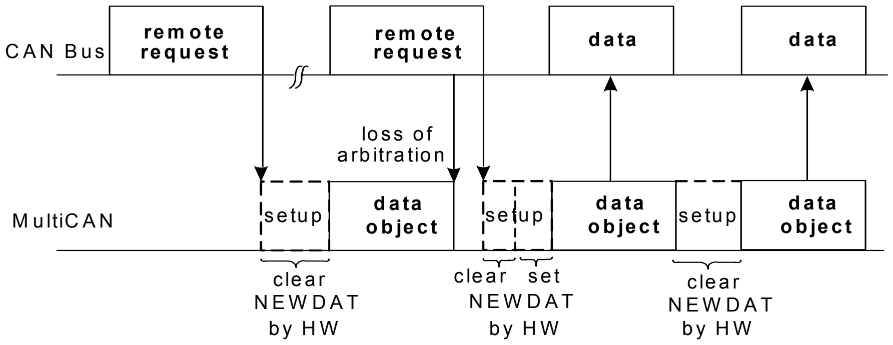

Figure 1 Loss of Arbitration

RESET_X.H003 How to Trigger a $\overline{\text{PORST}}$ after an Internal Failure

There is no internal User Reset that restores the complete device including the power system like a Power-On Reset. In some applications it is possible to connect $\overline{\text{ESR1}}$ or $\overline{\text{ESR2}}$ with the $\overline{\text{PORST}}$ pin and set the used $\overline{\text{ESR}}$ pin as Reset output. With this a WDT or Software Reset can trigger a Power-On Reset.

A detailed description is in the Application Note **AP16103**.

RTC_X.H003 Changing the RTC Configuration

The count input clock f_{RTC} for the Real Time Clock module (RTC) can be selected via bit field RTCCLKSEL in register `RTCCLKCON`. Whenever the system clock is less than 4 times faster than the RTC count input clock ($f_{SYS} < f_{RTC} \times 4$), Asynchronous Mode must be selected (bit RTCCM = 1_B in register `RTCCLKCON`).

To assure data consistency in the count registers `T14`, `RTCL`, `RTCH`, the RTC module must be temporarily switched off by setting bit MODEN = 0_B in register `RTC_KSCCFG` before register `RTCCLKCON` is modified, i.e. whenever

- changing the operating mode (Synchronous/Asynchronous) Mode in bit RTCCM, or
- changing the RTC count source in bit field RTCCLKSEL.

In case power domain DMP_1 is switched off, it is not required to switch the RTC to Asynchronous Mode, since it will receive a reset in any case.

USIC_AI.H001 FIFO RAM Parity Error Handling

A false RAM parity error may be signalled by the USIC module, which may optionally lead to a trap request (if enabled) for the USIC RAM, under the following conditions:

- a receive FIFO buffer is configured for the USIC module, and
- after the last power-up, less data elements than configured in bit field `SIZE` have been received in the FIFO buffer, and
- the last data element is read from the receiver buffer output register `OUTRL` (i.e. the buffer is empty after this read access).

Once the number of received data elements is greater than or equal to the receive buffer size configured in bit field `SIZE`, the effect described above can no longer occur.

To avoid false parity errors, it is recommended to initialize the USIC RAM before using the receive buffer FIFO. This can be achieved by configuring a 64-entry transmit FIFO and writing 64 times the value 0x0 to the FIFO input register `IN00` to fill the whole FIFO RAM with 0x0.