16/32-Bit

Architecture

XC2200 Derivatives

16/32-Bit Single-Chip Microcontroller
with 32-Bit Performance
XC2000 Family / Alpha Line

Errata Sheet

V1.11 2014-10

Microcontrollers

Edition 2014-10

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

Information

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office (**www.infineon.com**).

Warnings

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

16/32-Bit

Architecture

XC2200 Derivatives

16/32-Bit Single-Chip Microcontroller
with 32-Bit Performance
XC2000 Family / Alpha Line

Errata Sheet

V1.11 2014-10

Microcontrollers

Table of Contents

1 History List / Change Summary

Table 1 History List

Version	Date	Remark[1)]
1.0	09.06.2006	Errata Sheet XC2287 v1.0 (PG-LQFP-144)
1.1	28.02.2007	Errata Sheet XC2267 v1.0 (PG-LQFP-100) Errata Sheet XC2287 v1.1 (PG-LQFP-144)
1.2	30.03.2007	Errata Sheet XC2267 v1.1 (PG-LQFP-100) Errata Sheet XC2287 v1.2 (PG-LQFP-144)
1.3	23.07.2007	Errata Sheet XC22xx v1.3 (all package)
1.4	19.10.2007	Errata Sheet XC22xx v1.4 (all package)
1.5	11.04.2008	Errata Sheet for all product steps.
1.6	20.08.2008	Cancelled release due missing note.
1.7	22.08.2008	
1.8	30.01.2009	
1.9	08.07.2010	Errata No. 01537AERRA, new Errata Sheet layout
1.10	16.04.2013	Errata No. 02595AERRA. Removed EES-AA, EES-AB, EE-AB, EES-AC, ES-AC references from Marking/Step. Removed chapter 4 “Errata Device Overview”
1.11	09.10.2014	Errata No. 03263AERRA. Removed Errata BSL_X.003, CPU_X.004, FLASH_X.007, POWER_X.003, POWER_X.005, RESET_X.H002, WDT_X.H001 (already fixed in Marking/Step AB)

1) Errata changes to the previous Errata Sheet are marked in chapter Short Errata Description

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 XC2200 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 initially detected.
  - **AI**: Architecture Independent
  - **TC**: TriCore
  - **X**: XC166 / XE166 / XC2000 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 XC2000 Family comprises device types from the XC2200 group, the XC2300 group and the XC2700 group. The XC22xx device types belong to the XC2200 group.

Device	XC22xx
Marking/Step	AB, AC
Package	PG-LQFP-100**,** PG-LQFP-144

This Errata Sheet refers to the following documentation:

- XC2200 Derivatives User's Manual
- XC226x Data Sheet
- XC228x 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/xc2200** .

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 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”.

4.1 Functional Deviations

Table 2 shows a short description of the functional deviations.

Table 2 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
ADC_AI.001	Conversions requested in Slot 0 started twice		18
BROM_TC.006	Baud Rate Detection for CAN Bootstrap Loader		18
BSL_CAN_X.001	Quartz Crystal Settling Time after PORST too Long for CAN Bootstrap Loader		19
BSL_X.004	Evaluation of UART Bootstrap Loader Identification Byte in Single Wire Configuration		19
DPRAM_X.001	Parity Error Flag for DPRAM		19
EBC_X.007	Bus Arbitration not Properly Working		20
ESR_X.004	Wrong Value of SCU_RSTCONx Registers after ESRy Application Reset		20
FLASH_X.008	Flash Read after Flash Erase Command		21
GPT12E_X.001	T5/T6 in Counter Mode with BPS2 = $1X_B$		22
GPT12E_X.002	Effects of GPT Module Microarchitecture		22
GSC_X.001	Clearing of Request Triggers by the GSC		23
INT_X.007	Interrupt using a Local Register Bank during execution of IDLE		24
INT_X.008	HW Trap during Context Switch in Routine using a Local Bank		25
INT_X.009	Delayed Interrupt Service of Requests using a Global Bank		26
INT_X.010	HW Traps and Interrupts may get postponed		27
MultiCAN_AI.040	Remote frame transmit acceptance filtering error		28
MultiCAN_AI.041	Dealloc Last Obj		29

Table 2 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
MultiCAN_AI.042	Clear MSGVAL during transmit acceptance filtering		29
MultiCAN_AI.043	Dealloc Previous Obj		29
MultiCAN_AI.044	RxFIFO Base SDT		30
MultiCAN_AI.045	OVIE Unexpected Interrupt		30
MultiCAN_AI.046	Transmit FIFO base Object position		31
MultiCAN_TC.025	RXUPD behavior		31
MultiCAN_TC.026	MultiCAN Timestamp Function		32
MultiCAN_TC.027	MultiCAN Tx Filter Data Remote		32
MultiCAN_TC.028	SDT behavior		32
MultiCAN_TC.029	Tx FIFO overflow interrupt not generated		33
MultiCAN_TC.030	Wrong transmit order when CAN error at start of CRC transmission		34
MultiCAN_TC.031	List Object Error wrongly triggered		35
MultiCAN_TC.032	MSGVAL wrongly cleared in SDT mode		35
MultiCAN_TC.035	Different bit timing modes		36
MultiCAN_TC.037	Clear MSGVAL		37
MultiCAN_TC.038	Cancel TXRQ		38
OCDS_X.003	Peripheral Debug Mode Settings cleared by Reset		38
RESET_X.002	Startup Mode Selection is not Valid in SCU_STSTAT.HWCFG		39
RESET_X.003	P2.[2:0] and P10.[12:0] Switch to Input		39
RESET_X.004	Sticky "Register Access Trap" forces device into power-save mode after reset.		40
RTC_X.003	Interrupt Generation in Asynchronous Mode		40
USIC_AI.003	TCSRL.SOF and TCSRL.EOF not cleared after a transmission is started		41
USIC_AI.004	Receive shifter baudrate limitation		41
USIC_AI.005	Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time		41

Table 2 Functional Deviations (cont'd)

Functional Deviation	Short Description	Chg	Pg
USIC_AI.016	Transmit parameters are updated during FIFO buffer bypass		42
USIC_AI.018	Clearing PSR.MSLS bit immediately deasserts the SELOx output signal	New	42

4.2 Deviations from Electrical and Timing Specification

Table 3 shows a short description of the electrical- and timing deviations from the specification.

Table 3 Deviations from Electrical- and Timing Specification

AC/DC/ADC Deviation	Short Description	Chg	Pg
SWD_X.P002	Supply Watchdog (SWD) Supervision Level in Data Sheet.		44

4.3 Application Hints

Table 4 shows a short description of the application hints.

Table 4 Application Hints

Hint	Short Description	Chg	Pg
ADC_AI.H002	Minimizing Power Consumption of an ADC Module		45
CAPCOM12_X.H001	Enabling or Disabling Single Event Operation		45
CC6_X.H001	Modifications of Bit MODEN in Register CCU6x_KSCFG		46
GPT12_AI.H001	Modification of Block Prescalers BPS1 and BPS2		47
GPT12E_X.H002	Reading of Concatenated Timers		47
INT_X.H002	Increased Latency for Hardware Traps		48
INT_X.H004	SCU Interrupts Enabled After Reset		49
JTAG_X.H001	JTAG Pin Routing		49
LXBUS_X.H001	Do Not Access Reserved Locations on the LXBus		50
MultiCAN_AI.H005	TxD Pulse upon short disable request		50
MultiCAN_AI.H006	Time stamp influenced by resynchronization		51
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off		51
MultiCAN_AI.H008	Effect of CANDIS on SUSACK		51
MultiCAN_TC.H002	Double Synchronization of receive input		52
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		52
MultiCAN_TC.H004	Double remote request		52
OCDS_X.H002	Suspend Mode Behavior for MultiCAN		53
PVC_X.H001	PVC Threshold Level 2		54
RESET_X.H003	How to Trigger a PORST after an Internal Failure		55
RTC_X.H003	Changing the RTC Configuration		55
StartUp_X.H002	FCONCS0..FCONCS4 Registers are Always Configured in External Start-Up Mode		55
USIC_AI.H001	FIFO RAM Parity Error Handling		56

Table 4 Application Hints (cont'd)

Hint	Short Description	Chg	Pg
USIC_AI.H002	Configuration of USIC Port Pins		56
USIC_AI.H003	PSR.RXIDLE Cleared by Software		57

4.4 Documentation Updates

Table 5 gives a short description of the documentation updates.

Table 5 Documentation Updates

Documentation Updates	Short Description	Chg	Pg
EBC_X.D001	Visibility of Internal LXBus Cycles on External Address Bus		58
ID_X.D001	Identification Register		58
RESET_X.D001	Reset Types of Trap Registers		59
SCU_X.D007	SCU Interrupts Enabled After Reset		59
StartUp_X.D002	External Start-Up Mode Selection by Configuration Pins		59
USIC_X.D003	USIC0 Channel 1 Connection DX0D and DOUT		60
XTAL_X.D001	Input Voltage Amplitude V_{AX1} on XTAL1		60

5 Detailed Errata Description

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

5.1 Functional Deviations

ADC_AI.001 Conversions requested in Slot 0 started twice

A conversion n+1 requested in arbiter slot 0 will be started twice if all configuration and timing conditions of the following sequence are met:

1. A conversion n of a channel is currently running.
2. Slot 0 has won the arbitration while conversion n is in progress.
3. Conversion n ends one f_{ADCD} clock cycle before the end of an arbitration cycle.
4. The conversion n+1 initiated in slot 0 is started exactly in the last f_{ADCD} clock cycle of this arbitration-cycle (see 3.).
5. The conversion time of conversion n+1 is shorter than 2 arbitration cycles.

If all these conditions are met, then the request of slot 0 cannot be cleared in time by the arbiter, and conversion n+1 is requested a second time.

Workaround

The conversion time for channels requested in slot 0 must not be shorter than two arbitration cycles.

BROM_TC.006 Baud Rate Detection for CAN Bootstrap Loader

In a specific corner case, the baud rate detected during reception of the initialization frame for the CAN bootstrap loader may be incorrect. The probability for this sporadic problem is relatively low, and it decreases with decreasing CAN baud rate and increasing module clock frequency.

Workaround:

If communication fails, the host should repeat the CAN bootstrap loader initialization procedure after a reset of the device.

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 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 applications a ceramic resonator with settling time less than 0.5 ms can be used.
- An alternative is the Internal Start from on-chip Flash memory as startup mode after 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.

BSL_X.004 Evaluation of UART Bootstrap Loader Identification Byte in Single Wire Configuration

In the current implementation, transmission of the start bit of the identification byte ($D5_H$) partially overlaps with the stop bit time slot of the zero byte sent by the host. This does not present any problem in a duplex (2-wire) configuration.

If the UART bootstrap loader is used in a single wire configuration (RxD/TxD externally connected, e.g. K-line environment), depending on the baudrate, the start bit of the identification byte may not be correctly recognized by the host. At 9600 Baud, the host typically interprets the identification byte as $F5_H$.

Workaround

The host software either should not evaluate the received identification byte, or should also tolerate values other than $D5_H$.

DPRAM_X.001 Parity Error Flag for DPRAM

The parity error flag for the dual port memory (DPRAM) does not work correctly. Under certain conditions bit PECON.PEFDP is set, although there is no error in the DPRAM.

Workaround

Do not enable the parity error trap for the dual port memory, i.e. leave bit PECON.PEENDP = 0 (default after power-on reset).

EBC_X.007 Bus Arbitration not Properly Working

Due to a mismatch of pad propagation delays and internal operation cycle time the arbitration feature of the External Bus Controller (EBC) can only be used with severe restrictions. It is recommended not to use this feature, as future members of the XC2000/XE166 Family will no longer support bus arbitration.

Workaround

The usable conditions also depend on the application system and can only be defined for a specific use case.

ESR_X.004 Wrong Value of SCU_RSTCONx Registers after ESRy Application Reset

SCU_RSTCONx registers are reset only by Power-On, but they may be wrongly affected after a second application reset requested by an ESRy pin. This may lead to the SCU_RSTCONx register values being set to zero, which could unexpectedly disable reset sources within the user application. The conditions which lead to this behavior are:

1. First, an application reset by SW (software), CPU (Central Processing Unit), MP (Memory), WDT (Watchdog Timer) or ESRy (External Service Request y) occurs.
2. Following this, an application reset on an ESRy pin occurs.
3. If the above mentioned ESRy reset occurs during a critical time window of the SSW (startup software), then it's possible that the application will operate with the wrong SCU_RSTCONx register value. The critical time window occurs when the SSW is writing the SCU_RSTCONx registers, and at the same time, the ESRy reset request is processed by the reset circuitry. The width of this critical window $f_{critical\ window}$ is less than 13 cycles.

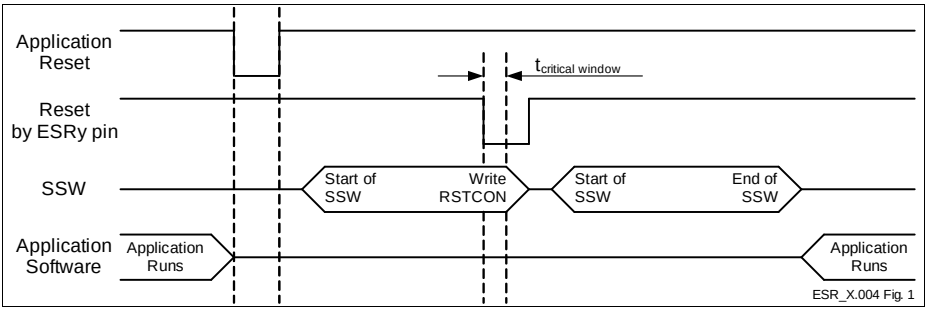

Figure 1 Critical application reset sequence

Workaround

- Initialize SCU_RSTCONx registers by user software after any reset, or
- assure that a second application reset request with an ESR pin does not occur during the critical time window.

FLASH_X.008 Flash Read after Flash Erase Command

Under certain conditions all Flash erase commands do not work correctly. After erasing, all erased bits must be programmed with new data or with all-zero data before reading any data from the addressed sector is allowed.

Workarounds

1. Erase a range of Flash memory and program it completely with new data before reading. This is the fastest solution.
   Additional hint: A Flash driver could implement a programming function that performs first an “Erase Page” and uses directly thereafter “Program Page” to program the data of this page. The Flash driver wouldn’t need any separate erase function.
2. Erase a range of Flash memory and program it completely with all-zero data. Only after this the range may be read. Data can be programmed later[1)].

1) Please note: only in order to implement this workaround for the noted device steps it is allowed to execute two program commands before erasing it.

GPT12E_X.001 T5/T6 in Counter Mode with BPS2 = $1X_B$

When T5 and/or T6 are configured for counter mode (bit field TxM = 001_B in register GPT12E_TxCON, x = 5, 6), **and** bit field **BPS2 = $1X_B$** in register GPT12E_T6CON, then edge detection for the following count input and control signals does not work correctly:

T5IN, T6IN, T5EUD, T6EUD.

Note: The configuration where T5 counts the overflow/underflow events of T6 is not affected by this problem.

Workaround

Do not set bit field BPS2 = $1X_B$ in register GPT12E_T6CON when T5 and/or T6 are configured for counter mode. Use only settings BPS2 = $0X_B$ when T5 and/or T6 are configured for counter mode.

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 of 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.

GSC_X.001 Clearing of Request Triggers by the GSC

After a request from sources with priority 5..10 ($\overline{ESR0}$...GPT12E, see table in Chapter “Global State Controller (GSC)” of the current User’s Manual), the following problem will occur:

A trigger for a command request (Wake-up, Clock-off, Suspend Mode) that is enabled in register GSCEN remains pending in the GSC after the arbitration has been finished and the command has been requested. As a consequence, further request triggers with the same or a lower priority will be ignored (14 = lowest priority).

Example

A request from the OCDS to enter Suspend Mode (request source OCDS entry, priority 14) will be ignored if (at any time before) an interrupt request has occurred (request source ITC, priority 9), and the ITC request trigger is enabled in register GSCEN. In this case, modules that are programmed to stop in Suspend Mode (selected in bit field SUMCFG) will continue to run.

Workaround

Disable triggers from request sources that are not used by the application in register GSCEN.
For other sources that shall trigger the GSC, clear and then set again the respective trigger enable bits in register GSCEN each time the GSC logic shall be armed.

INT_X.007 Interrupt using a Local Register Bank during execution of IDLE

During the execution of the IDLE instruction, if an interrupt which uses a local register bank is acknowledged, the CPU may stall, preventing further code execution. Recovery from this condition can only be made through a hardware or watchdog reset.

All of the following conditions must be present for the problem to occur:

- The IDLE instruction is executed while the **global** register bank is selected (bit field BANK = 00_B in register PSW),
- The interrupting routine is using one of the **local** register banks (BANK = 10_B or 11_B), and the local register bank is selected automatically via the bank selection registers BNKSEL0...3, (i.e. the interrupting routine has a priority level ≥12),
- The system stack is located in the internal dual-ported RAM (DPRAM, locations $0F600_H$ ... $0FDFF_H$),
- The interrupt is acknowledged during the first 8 clock cycles of the IDLE instruction execution.

Workaround 1

Disable interrupts (either globally, or only interrupts using a local register bank) before execution of IDLE:

```
BCLR IEN   ; Disable interrupts globally
IDLE       ; CPU enters idle mode
BSET IEN   ; After exit from idle mode
              ;   re-enable interrupts
```

If an interrupt request is generated during this sequence, the CPU leaves idle mode and acknowledges the interrupt after BSET IEN.

Workaround 2

Do not use local register banks, use only global register banks.

Workaround 3

Locate the system stack in a memory other than the DPRAM, e.g. in DSRAM.

INT_X.008 HW Trap during Context Switch in Routine using a Local Bank

When a hardware trap occurs under specific conditions in a routine using a local register bank, the CPU may stall, preventing further code execution. Recovery from this condition can only be made through a hardware or watchdog reset.

All of the following conditions must be present for this problem to occur:

- The routine that is interrupted by the hardware trap is using one of the **local** register banks (bit field PSW.BANK = 10_B or 11_B)
- The system stack is located in the internal dual-ported RAM (DPRAM, locations $0F600_H$ ... $0FDFF_H$)
- The hardware trap occurs in the second half (load phase) of a context switch operation triggered by one of the following actions:
  - a) Execution of the IDLE instruction, or
  - b) Execution of an instruction writing to the Context Pointer register CP (untypical case, because this would mean that the routine using one of the local banks modifies the CP contents of a global bank)

Workaround 1

Locate the system stack in a memory other than the DPRAM, e.g. in DSRAM.

Workaround 2

Do not use local register banks, use only global register banks.

Workaround 3

Condition b) (writing to CP while a local register bank context is selected) is not typical for most applications. If the application implementation already eliminates the possibility for condition b), then only a workaround for condition a) is required.
The workaround for condition a) is to make sure that the IDLE instruction is executed within a code sequence that uses a global register bank context.

INT_X.009 Delayed Interrupt Service of Requests using a Global Bank

Service of an interrupt request using a global register bank is delayed - regardless of its priority - if it would interrupt a routine using one of the local register banks in the following situations:

Case 1:

- The Context Pointer CP is written to (e.g. by POP, MOV, SCXT ... instructions) within a routine that uses one of the **local** register banks (bit field PSW.BANK = 10_B or 11_B),
- Then an interrupt request occurs which is programmed (with GPRSELx = 00_B) to automatically use the **global** bank via the bank selection registers BNKSEL0...3 (i.e. the interrupting routine has a priority level ≥12).

Note that this scenario is regarded as untypical case, because this would mean that the routine using one of the local banks modifies the CP contents of a global bank.

In this case service of the interrupt request is delayed until bit field PSW.BANK becomes 00_B, e.g. by explicitly writing to the PSW, or by an implicit update from the stack when executing the RETI instruction at the end of the routine using the local bank.

Case 2 (see also Figure 1):

- The Context Pointer CP is written to (e.g. by POP, MOV, SCXT ... instructions) within a routine (Task A) that uses a **global** register bank (bit field PSW.BANK = 00_B), i.e. the context for this routine will be modified,
- This context switch procedure (19 cycles) is interrupted by an interrupt request (Task B) which is programmed (with GPRSELx = $1X_B$) to automatically use one of the **local** banks via the bank selection registers BNKSEL0...3 (i.e. the interrupting routine has a priority level ≥12),
- Before the corresponding interrupt service routine is finished, another interrupt request (Task C) occurs which is programmed (with GPRSELx = 00_B) to automatically use the **global** bank via the bank selection registers BNKSEL0...3 (i.e. the interrupting routine has a priority level ≥13)

In this case service of this interrupt request (for Task C) is delayed until bit field PSW.BANK becomes 00_B after executing the RETI instruction at the end of the routine (Task B) using the local bank.

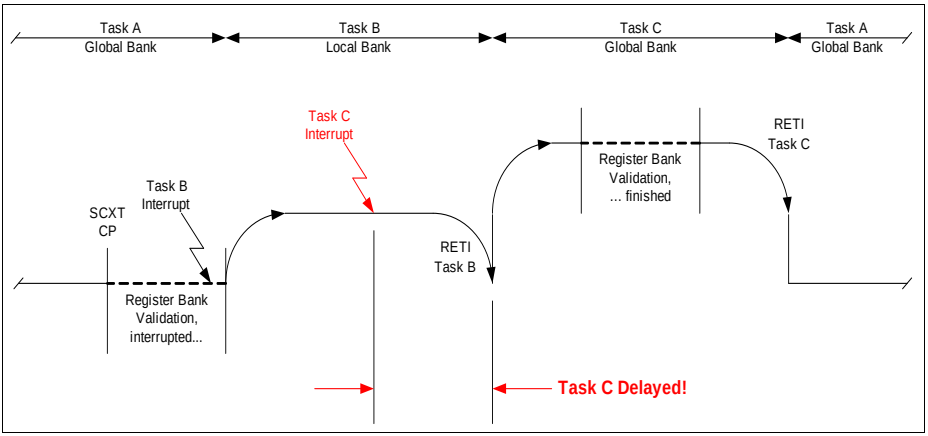

Figure 2 Example for Case 2: Interrupt Service for Task C delayed

Workaround for Case 1

Do not write to the CP register (i.e. modify the context of a global bank) while a local register bank context is selected.

Workaround for Case 2

When using both local and global register banks via the bank selection registers BNKSEL0...3 for interrupts on levels ≥12, ensure that there is no interrupt using a global register bank that has a higher priority than an interrupt using a local register bank.

Example 1:
Local bank interrupts are used on levels 14 and 15, no local bank interrupts on level 12 and 13. In this case, global bank interrupts on level 15 must not be used.

Example 2:
Local bank interrupts are used on level 12. In this case, no global bank interrupts must be used on levels 13, 14, 15.

INT_X.010 HW Traps and Interrupts may get postponed

Under the special conditions described below, a hardware trap (HWTx) and subsequent interrupts, PEC transfers, OCDS service requests (on priority level < 11_H) or class B and class A traps (if HWTx also was class A) may get postponed until the next RETI instruction is executed. If no RETI is executed, these requests may get postponed infinitely.

Both of the following conditions must be fulfilled at the same time when the trigger for the hardware trap HWTx occurs in order to cause the problem:

1. The pipeline is cancelled due to one of the following reasons:
   a) a multiply or divide instruction is followed by a mispredicted conditional (zero-cycle) jump.
   b) a class A hardware trap is triggered quasi-simultaneously with the request for a class B trap (= HWTx), i.e. the trigger for the class A trap arrives before the previously injected TRAP instruction for the class B trap has reached the Execute stage of the pipeline.
      In this case, the class A trap is entered, but when the RETI instruction at the end of the class A trap routine is executed, the pending class B trap (HWTx) is **not** entered, and subsequent interrupts/PECs/class B traps are postponed until the next RETI.
   c) a break is requested by the debugger.
2. The pipeline is stalled in the Execute or Write Back stage due to consecutive writes, or due to a multi-cycle write that is performed to a memory area with wait states (PSRAM, external memory).

Workaround

Disable overrun of pipeline bubbles by setting bit OVRUN (CPUCON2.4) = 0.

MultiCAN_AI.040 Remote frame transmit acceptance filtering error

Correct behaviour:

Assume the MultiCAN message object receives a remote frame that leads to a valid transmit request in the same message object (request of remote answer), then the MultiCAN module prepares for an immediate answer of the remote request. The answer message is arbitrated against the winner of transmit acceptance filtering (without the remote answer) with a respect to the priority class (`MOARn.PRI`).

Wrong behaviour:

Assume the MultiCAN message object receives a remote frame that leads to a valid transmit request in the same message object (request of remote answer), then the MultiCAN module prepares for an immediate answer of the remote request. The answer message is arbitrated against the winner of transmit acceptance filtering (without the remote answer) with a respect to the CAN arbitration rules and not taking the PRI values into account.

If the remote answer is not sent out immediately, then it is subject to further transmit acceptance filtering runs, which are performed correctly.

Workaround

Set MOFCRn.FRREN=1_B and MOFGPRn.CUR to this message object to disable the immediate remote answering.

MultiCAN_AI.041 Dealloc Last Obj

When the last message object is deallocated from a list, then a false list object error can be indicated.

Workaround

- Ignore the list object error indication that occurs after the deallocation of the last message object.

or

- Avoid deallocating the last message object of a list.

MultiCAN_AI.042 Clear MSGVAL during transmit acceptance filtering

Assume all CAN nodes are idle and no writes to MOCTRn of any other message object are performed. When bit MOCTRn.MSGVAL of a message object with valid transmit request is cleared by software, then MultiCAN may not start transmitting even if there are other message objects with valid request pending in the same list.

Workaround

- Do not clear MOCTRn.MSGVAL of any message object during CAN operation. Use bits MOCTRn.RXEN, MOCTRn.TXEN0 instead to disable/reenable reception and transmission of message objects.

or

- Take a dummy message object, that is not allocated to any CAN node. Whenever a transmit request is cleared, set MOCTRm.TXRQ of the dummy message object thereafter. This retriggers the transmit acceptance filtering process.

MultiCAN_AI.043 Dealloc Previous Obj

Assume two message objects m and n (message object n = MOCTRm.PNEXT, i.e. n is the successor of object m in the list) are allocated. If message m is reallocated to another list or to another position while the transmit or receive acceptance filtering run is

performed on the list, then message object n may not be taken into account during this acceptance filtering run. For the frame reception message object n may not receive the message because n is not taken into account for receive acceptance filtering. The message is then received by the second priority message object (in case of any other acceptance filtering match) or is lost when there is no other message object configured for this identifier.For the frame transmission message object n may not be selected for transmission, whereas the second highest priority message object is selected instead (if any). If there is no other message object in the list with valid transmit request, then no transmission is scheduled in this filtering round. If in addition the CAN bus is idle, then no further transmit acceptance filtering is issued unless another CAN node starts a transfer or one of the bits MSGVAL, TXRQ, TXEN0, TXEN1 is set in the message object control register of any message object.

Workaround

- After reallocating message object m, write the value one to one of the bits MSGVAL, TXRQ, TXEN0, TXEN1 of the message object control register of any message object in order to retrigger transmit acceptance filtering.
- For frame reception, make sure that there is another message object in the list that can receive the message targeted to n in order to avoid data loss (e.g. a message object with an acceptance mask=0_D and PRI=3_D as last object of the list).

MultiCAN_AI.044 RxFIFO Base SDT

If a receive FIFO base object is located in that part of the list, that is used for the FIFO storage container (defined by the top and bottom pointer of this base object) and bit SDT is set in the base object (CUR pointer points to the base object), then MSGVAL of the base object is cleared after storage of a received frame in the base object without taking the setting of MOFGPRn.SEL into account.

Workaround

Take the FIFO base object out of the list segment of the FIFO slave objects, when using Single Data Transfer.

MultiCAN_AI.045 OVIE Unexpected Interrupt

When a gateway source object or a receive FIFO base object with MOFCRn.OVIE set transmits a CAN frame, then after the transmission an unexpected interrupt is generated on the interrupt line as given by MOIPRm.RXINP of the message object referenced by m=MOFGPRn.CUR.

Workaround

Do not transmit any CAN message by receive FIFO base objects or gateway source objects with bit MOFCRn.OVIE set.

MultiCAN_AI.046 Transmit FIFO base Object position

If a message object n is configured as transmit FIFO base object and is located in the list segment that is used for the FIFO storage container (defined by MOFGPRn.BOT and MOFGPRn.TOP) but not at the list position given by MOFGPRn.BOT, then the MultiCAN uses incorrect pointer values for this transmit FIFO.

Workaround

The transmit FIFO works properly when the transmit FIFO base object is either at the bottom position within the list segment of the FIFO (MOFGPRn.BOT=n) or outside of the list segment as described above.

MultiCAN_TC.025 RXUPD behavior

When a CAN frame is stored in a message object, either directly from the CAN node or indirectly via receive FIFO or from a gateway source object, then bit MOCTR.RXUPD is set in the message object before the storage process and is automatically cleared after the storage process.

Problem description

When a standard message object (MOFCR.MMC) receives a CAN frame from a CAN node, then it processes its own RXUPD as described above (correct).

In addition to that, it also sets and clears bit RXUPD in the message object referenced by pointer MOFGPR.CUR (wrong behavior).

Workaround

The “foreign” RXUPD pulse can be avoided by initializing MOFGPR.CUR with the message number of the object itself instead of another object (which would be message object 0 by default, because MOFGPR.CUR points to message object 0 after reset initialization of MultiCAN).

MultiCAN_TC.026 MultiCAN Timestamp Function

The timestamp functionality does not work correctly.

Workaround

Do not use timestamp.

MultiCAN_TC.027 MultiCAN Tx Filter Data Remote

Message objects of priority class 2 (MOAR.PRI = 2) are transmitted in the order as given by the CAN arbitration rules. This implies that for 2 message objects which have the same CAN identifier, but different DIR bit, the one with DIR = 1 (send data frame) shall be transmitted before the message object with DIR = 0, which sends a remote frame. The transmit filtering logic of the MultiCAN leads to a reverse order, i.e the remote frame is transmitted first. Message objects with different identifiers are handled correctly.

Workaround

None.

MultiCAN_TC.028 SDT behavior

Correct behavior

Standard message objects:

MultiCAN clears bit MOCTR.MSGVAL after the successful reception/transmission of a CAN frame if bit MOFCR.SDT is set.

Transmit Fifo slave object:

MultiCAN clears bit MOCTR.MSGVAL after the successful reception/transmission of a CAN frame if bit MOFCR.SDT is set. After a transmission, MultiCAN also looks at the respective transmit FIFO base object and clears bit MSGVAL in the base object if bit SDT is set in the base object and pointer MOFGPR.CUR points to MOFGPR.SEL (after the pointer update).

Gateway Destination/Fifo slave object:

MultiCAN clears bit MOCTR.MSGVAL after the storage of a CAN frame into the object (gateway/FIFO action) or after the successful transmission of a CAN frame if bit MOFCR.SDT is set. After a reception, MultiCAN also looks at the respective FIFO base/Gateway source object and clears bit MSGVAL in the base object if bit SDT is set in

the base object and pointer MOFGPR.CUR points to MOFGPR.SEL (after the pointer update).

Problem description

Standard message objects:

After the successful transmission/reception of a CAN frame, MultiCAN also looks at message object given by MOFGPR.CUR. If bit SDT is set in the referenced message object, then bit MSGVAL is cleared in the message object CUR is pointing to.

Transmit FIFO slave object:

Same wrong behaviour as for standard message object. As for transmit FIFO slave objects CUR always points to the base object, the whole transmit FIFO is set invalid after the transmission of the first element instead after the base object CUR pointer has reached the predefined SEL limit value.

Gateway Destination/Fifo slave object:

Correct operation of the SDT feature.

Workaround

Standard message object:

Set pointer MOFGPR.CUR to the message number of the object itself.

Transmit FIFO:

Do not set bit MOFCR.SDT in the transmit FIFO base object. Then SDT works correctly with the slaves, but the FIFO deactivation feature by CUR reaching a predefined limit SEL is lost.

MultiCAN_TC.029 Tx FIFO overflow interrupt not generated

Specified behaviour

After the successful transmission of a Tx FIFO element, a Tx overflow interrupt is generated if the FIFO base object fulfils these conditions:

- Bit MOFCR.OVIE=1, AND
- MOFGPR.CUR becomes equal to MOFGPR.SEL

Real behaviour

A Tx FIFO overflow interrupt will not be generated after the transmission of the Tx FIFO base object.

Workaround

If Tx FIFO overflow interrupt needed, take the FIFO base object out of the circular list of the Tx message objects. That is to say, just use the FIFO base object for FIFO control, but not to store a Tx message.

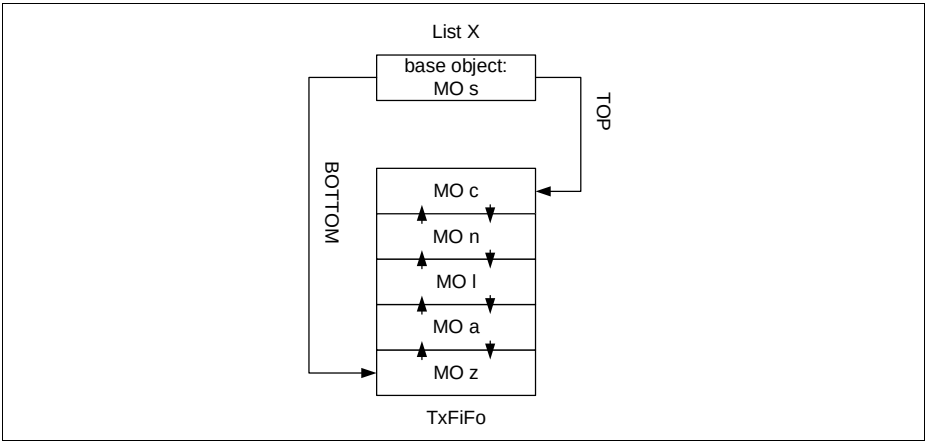

Figure 3 FIFO structure

MultiCAN_TC.030 Wrong transmit order when CAN error at start of CRC transmission

The priority order defined by acceptance filtering, specified in the message objects, define the sequential order in which these messages are sent on the CAN bus. If an error occurs on the CAN bus, the transmissions are delayed due to the destruction of the message on the bus, but the transmission order is kept. However, if a CAN error occurs when starting to transmit the CRC field, the arbitration order for the corresponding CAN node is disturbed, because the faulty message is not retransmitted directly, but after the next transmission of the CAN node.

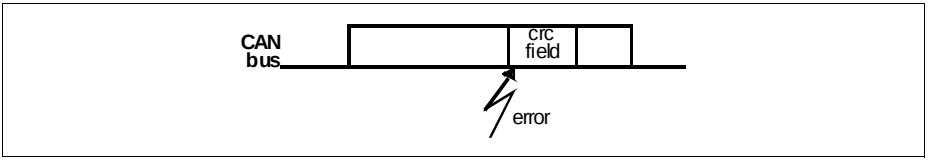

Figure 4

Workaround

None.

MultiCAN_TC.031 List Object Error wrongly triggered

If the first list object in a list belonging to an active CAN node is deallocated from that list position during transmit/receive acceptance filtering (happening during message transfer on the bus), then a "list object" error may occur (NSRx.LOE=1_B), which will cause that effectively no acceptance filtering is performed for this message by the affected CAN node.

As a result:

- for the affected CAN node, the CAN message during which the error occurs will not be stored in a message object. This means that although the message is acknowledged on the CAN bus, its content will be ignored.
- the message handling of an ongoing transmission is not disturbed, but the transmission of the subsequent message will be delayed, because transmit acceptance filtering has to be started again.
- message objects with pending transmit request might not be transmitted at all due to failed transmit acceptance filtering.

Workaround

EITHER:

- Avoid deallocation of the first element on active CAN nodes. Dynamic reallocations on message objects behind the first element are allowed, OR
- Avoid list operations on a running node. Only perform list operations, if CAN node is not in use (e.g. when NCRx.INIT=1_B)

MultiCAN_TC.032 MSGVAL wrongly cleared in SDT mode

When Single Data Transfer Mode is enabled (MOFCRn.SDT=1_B), the bit MOCTRn.MSGVAL is cleared after the reception of a CAN frame, no matter if it is a data frame or a remote frame.

In case of a remote frame reception and with MOFCR.FRREN = 0_B, the answer to the remote frame (data frame) is transmitted despite clearing of MOCTRn.MSGVAL (incorrect behaviour). If, however, the answer (data frame) does not win transmit acceptance filtering or fails on the CAN bus, then no further transmission attempt is made due to cleared MSGVAL (correct behaviour).

Workaround

- To avoid a single trial of a remote answer in this case, set MOFCR.FRREN = 1_B and MOFGPR.CUR = this object.

MultiCAN_TC.035 Different bit timing modes

Bit timing modes (NFCRx.CFMOD=10_B) do not conform to the specification.

When the modes 001_B-100_B are set in register NFCRx.CFSEL, the actual configured mode and behaviour is different than expected.

Table 6

Bit timing mode (NFCR.CFSEL) according to spec	Value to be written to NFCR.CFSEL instead	Measurement
001_B	Mode is missing (not implemented) in MultiCAN	Whenever a recessive edge (transition from 0 to 1) is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent dominant edge is stored in CFC.
010_B	011_B	Whenever a dominant edge is received as a result of a transmitted dominant edge the time (clock cycles) between both edges is stored in CFC.
011_B	100_B	Whenever a recessive edge is received as a result of a transmitted recessive edge the time (clock cycles) between both edges is stored in CFC.
100_B	001_B	Whenever a dominant edge that qualifies for synchronization is monitored on the receive input the time (measured in clock cycles) between this edge and the most recent sample point is stored in CFC.

Workaround

None.

MultiCAN_TC.037 Clear MSGVAL

Correct behaviour:

When MSGVAL is cleared for a message object in any list, then this should not affect the other message objects in any way.

Message reception (wrong behaviour):

Assume that a received CAN message is about to be stored in a message object A, which can be a standard message object, FIFO base, FIFO slave, gateway source or gateway destination object.

If during of the storage action the user clears MOCTR.MSGVAL of message object B in any list, then the MultiCAN module may wrongly interpret this temporarily also as a clearing of MSGVAL of message object A. The result of this is that the message is not stored in message object A and is lost. Also no status update is performed on message object A (setting of NEWDAT, MSGLST, RXPND) and no message object receive interrupt is generated. Clearing of MOCTR.MSGVAL of message object B is performed correctly.

Message transmission (wrong behaviour):

Assume that MultiCAN is about to copy the message content of a message object A into the internal transmit buffer of the CAN node for transmission.

If during of the copy action the user clears MOCTR.MSGVAL of message object B in any list, then the MultiCAN module may wrongly interpret this also as a clearing of MSGVAL of message object A. The result of this is that the copy action for message A is not performed, bit NEWDAT is not cleared and no transmission takes place (clearing MOCTR.MSGVAL of message object B is performed correctly). In case of idle CAN bus and the user does not actively set the transmit request of any message object, this may lead to not transmitting any further message object, even if they have a valid transmit request set.

Single data transfer feature:

When the MultiCAN module clears MSGVAL as a result of a single data transfer (MOFCR.SDT = 1 in the message object), then the problem does not occur. The problem only occurs if MSGVAL of a message object is cleared via CPU.

Workaround

Do not clear MOCTR.MSGVAL of any message object during CAN operation. Use bits MOCTR.RXEN, MOCTR.TXEN0 instead to disable/reenable reception and transmission of message objects.

MultiCAN_TC.038 Cancel TXRQ

When the transmit request of a message object that has won transmit acceptance filtering is cancelled (by clearing MSGVAL, TXRQ, TXEN0 or TXEN1), the CAN bus is idle and no writes to MOCTR of any message object are performed, then MultiCAN does not start the transmission even if there are message objects with valid transmit request pending.

Workaround

To avoid that the CAN node ignores the transmission:

- take a dummy message object, that is not allocated to any CAN node. Whenever a transmit request is cleared, set TXRQ of the dummy message object thereafter. This retriggers the transmit acceptance filtering process.

or:

- whenever a transmit request is cleared, set one of the bits TXRQ, TXEN0 or TXEN1, which is already set, again in the message object for which the transmit request is cleared or in any other message object. This retriggers the transmit acceptance filtering process.

OCDS_X.003 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. ADC, CCU6 and USIC**,** 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.

RESET_X.002 Startup Mode Selection is not Valid in SCU_STSTAT.HWCFG

Reading from SCU_STSTAT.HWCFG-bitfield returns all zeros instead of the information which startup mode has been entered after the last reset.

Workaround

Read the initial value from VECSEG register to evaluate where from the user code is started:

- VECSEG[7:0]=00_H - start from an off-chip memory, external startup mode
- VECSEG[7:0]=$C0_H$ - start from on-chip flash, internal startup mode
- VECSEG[7:0]=$E0_H$ - start from on-chip PSRAM, bootstrap loader mode (UART, CAN or SSC)

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.

RESET_X.004 Sticky "Register Access Trap" forces device into power-save mode after reset.

The system control unit (SCU) provides trap generation, to respond to certain system level events or faults. Certain trap sources maintain sticky trap flags which are only cleared explicitly by software, or by a power-on reset. These sticky trap flags are contained in the SCU register DMPMIT.

In case the "Register Access Trap" flag (DMPMIT.RAT) becomes set, but is not cleared before a debug, internal application, or application reset occurs, then the microcontroller will reset, but will fail to start-up correctly. The microcontroller start-up software will detect that the sticky trap flag is set, and will force the device into power-save mode with DMP_1 shut down and DMP_M powered.

Workaround

In response to the trap event, software must explicitly clear the sticky trap flag using the SCU register DMPMITCLR, before executing a debug, internal application, or application reset.

Note that this workaround does not address unexpected debug, internal application, or application resets which occur between the sticky trap event and the clearing of the sticky flags by software. To keep this exposure period as short as possible, it is recommended to clear the flag early in the trap routine.

Note: Register DMPMITCLR is protected by the register security mechanism after execution of the EINIT instruction and must be unlocked before accessing.

RTC_X.003 Interrupt Generation in Asynchronous Mode

Asynchronous Mode must be selected (bit RTCCM = 1_B in register RTCCLKCON) whenever the system clock is less than 4 times faster than the RTC count input clock ($f_{SYS} < f_{RTC} \times 4$). While in Asynchronous Mode, generation of the RTC interrupt via flag RTCIR in register RTC_IC does not work correctly.

Workaround 1

Select the system clock such that it is at least 4 times faster than the RTC count input clock ($f_{SYS} \geq f_{RTC} \times 4$) and operate the RTC in Synchronous Mode.

Workaround 2

Before switching from Synchronous to Asynchronous Mode, clear the individual interrupt enable bits (CNTxIE, T14IE) in register RTC_ISNC. After returning to Synchronous Mode, set the individual interrupt enable bits as required by the application. Then flag

RTCIR will get set if at least one of the individual interrupt requests (flags CNTxIR, T14IR) was pending.

USIC_AI.003 TCSRL.SOF and TCSRL.EOF not cleared after a transmission is started

The Start of Frame (SOF) and End of Frame (EOF) bit in the Transmit Control/Status Register (TCSRL) will not be cleared by hardware when the data is transfered from the transmit buffers (TBUFx) to the transmit shift register, i.e. the transmission of a new word starts.

Workaround

Clear TCSRL.SOF and TCSRL.EOF by software.

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.

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

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

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

Workaround

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

USIC_AI.016 Transmit parameters are updated during FIFO buffer bypass

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

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

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

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

Workaround

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

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

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

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

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

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

Workaround

There are two possible workarounds:

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

5.2 Deviations from Electrical- and Timing Specification

SWD_X.P002 Supply Watchdog (SWD) Supervision Level in Data Sheet.

The Supply Watchdog (SWD) Supervision Level V_{SWD} tolerance boundaries for 5.5 V are changed from ± 0.15 V to ± 0.30 V.

5.3 Application Hints

ADC_AI.H002 Minimizing Power Consumption of an ADC Module

For a given number of A/D conversions during a defined period of time, the total energy (power over time) required by the ADC analog part during these conversions via supply V_{DDPA} is approximately proportional to the converter active time.

Recommendation for Minimum Power Consumption:

In order to minimize the contribution of A/D conversions to the total power consumption, it is recommended

1. to select the internal operating frequency of the analog part (f_{ADCI}) near the **maximum** value specified in the Data Sheet, and
2. to switch the ADC to a power saving state (via ANON) while no conversions are performed. Note that a certain wake-up time is required before the next set of conversions when the power saving state is left.

Note: The selected internal operating frequency of the analog part that determines the conversion time will also influence the sample time t_S. The sample time t_S can individually be adapted for the analog input channels via bit field STC.

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.

GPT12_AI.H001 Modification of Block Prescalers BPS1 and BPS2

The block prescalers BPS1 and BPS2, controlled via bit fields T3CON.BSP1 and T6CON.BPS2, determine the basic clock for the GPT1 and GPT2 block, respectively.

After reset, when initializing a block prescaler BPSx to a value different from its default value (00_B), it must be initialized first before any mode involving external trigger signals is configured for the associated GPTx block. These modes include counter, incremental interface, capture, and reload mode. Otherwise, unintended count/capture/reload events may occur.

In case a block prescaler BPSx needs to be modified during operation of the GPTx block, disable related interrupts before modification of BPSx, and afterwards clear the corresponding service request flags and re-initialize those registers (T2, T3, T4 in block GPT1, and T5, T6, CAPREL in block GPT2) that might be affected by an unintended count/capture/reload event.

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 7** 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 7**:

Table 7 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.

INT_X.H004 SCU Interrupts Enabled After Reset

Following a reset, the SCU interrupts are enabled by default (register SCU_INTDIS = 0000_H). This may lead to interrupt requests being triggered in the SCU immediately, even before user software has begun to execute. In the SCU, multiple interrupt sources are `ORed` to a common interrupt node of the CPU interrupt controller. Due to the "ORing" of multiple interrupt sources, only one interrupt request to the interrupt controller will be generated if multiple sources at the input of this OR gate are active at the same time. If user software enables an interrupt in the interrupt controller (SCU_xIC) which shares the same node as the SCU interrupt request active after reset, it may lead to the effect of suppressing the intended interrupt source. So, for all SCU interrupt sources which will not be used, make sure to disable the interrupt source (SCU_INTDIS) and clear any pending request flags (SCU_xIC.IR) before enabling interrupts in interrupt controller.

JTAG_X.H001 JTAG Pin Routing

In the current device, the pins connected to the JTAG interface can be selected by software (write to register DBGPRR). After a reset, the JTAG interface is connected to position A (see **Table 8**). If connected to these pins, the debugger will work without any restrictions.

Table 8 JTAG Position A

Pin LQFP-100	Pin LQFP-144	Symbol	Signal
23	34	P5.2	TDI_A
6	8	P7.0	TDO_A
57	82	P2.9	TCK_A
28	39	P5.4	TMS_A
5	6	$\overline{\text{TRST}}$	$\overline{\text{TRST}}$

To use other pins for the JTAG interface, the following sequence of steps must be executed:

- $\overline{\text{TRST}}$ must be high at the rising edge of $\overline{\text{PORST}}$. Usually debuggers provide that.
- Debuggers must be set to do a so called `hot attach`. This is connecting the microcontroller without executing a reset.
- Execute a write to DBGPRR register with the desired selection of pins to be used with one of the first instructions out of Flash.

- TCK_A must be stable on a valid low or high level until the change of the JTAG interface was executed.
- A Halt after reset can be achieved by a program loop that is left by control through the debugger.

Verify the correct operation of the sequence within the actual application.

LXBUS_X.H001 Do Not Access Reserved Locations on the LXBus

Some of the on-chip peripherals are connected via the LXBus. The EBC controls this access by using CS7 to define the LXBus area.

The memory map lists several sections occupied by the on-chip LXBus peripherals MultiCAN and the USIC modules.

The reserved sections within the address range $20'0000_H$ ... $20'FFFF_H$ shown in the memory map are designated to additional peripherals for future derivatives. The sizes of the reserved sections depend on the chosen device type.

These reserved sections must not be accessed by user software nor by debuggers. Access to these sections may lead to a CPU lock situation caused by a bus lock and also makes the software incompatible with other derivatives. The error mode can only be left by a reset.

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

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

`PMCON1.CAN_DIS` = 1 and then `PMCON1.CAN_DIS` = 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:

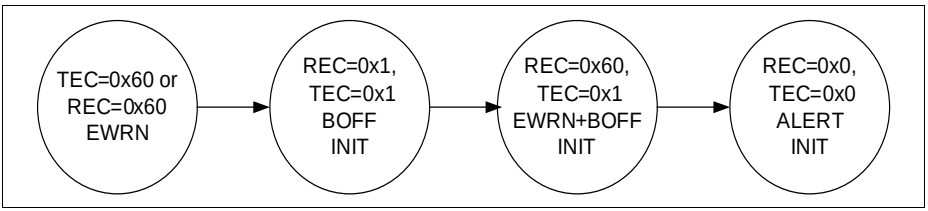

Figure 5 Alert Interrupt Behavior in case of Bus-Off

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

MultiCAN_AI.H008 Effect of CANDIS on SUSACK

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

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

MultiCAN_TC.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 frequency 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 etc.).

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.

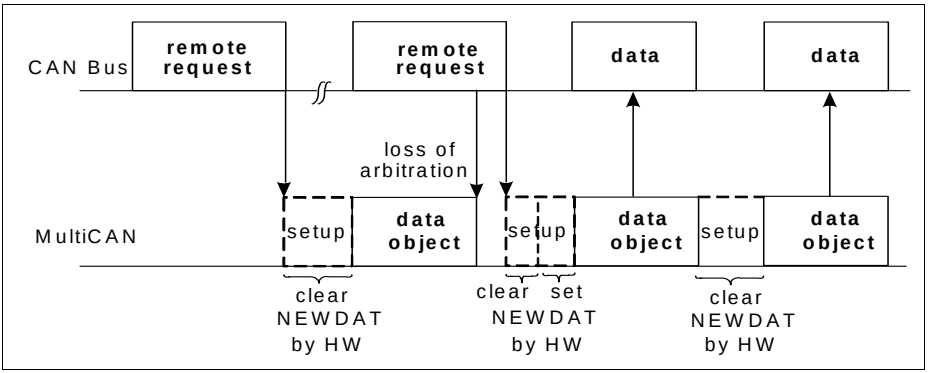

Figure 6 Loss of Arbitration

OCDS_X.H002 Suspend Mode Behavior for MultiCAN

The MultiCAN module basically provides two mechanisms to stop participation in CAN bus communication when a suspend request is issued by the OCDS:

Suspend operation of selected CAN nodes

The sensitivity to a suspend request can be individually enabled/disabled for each CAN node via bit SUSEN in its associated Node Control Register NCRx. With SUSEN = 1_B, upon a suspend request bit INIT is internally forced to 1_B to disable the CAN node as soon as it becomes BUS IDLE or BUS OFF. This way, a CAN node correctly finishes a running CAN frame, but does not start a new one. The network is not blocked due to the suspend state of one communication partner. All CAN registers can be read and written in this state since the module clock is not switched off.

Notes

1. *Depending on CAN activity and bus speed, the contents of some MultiCAN registers may still change if the debugger immediately reads them before the CAN node has reached BUS IDLE or BUS OFF state, i.e. before bit SUSACK = 1_B.*
2. *Bit field SUMCFG in register KSCCFG for the MultiCAN module must be set to 00_B to avoid an immediate stop (see below).*

Immediately stop operation of MultiCAN module

When bit field SUMCFG in register KSCCFG for the MultiCAN module is set to $1X_B$, the clock for the MultiCAN module is switched off as soon as the suspend request from the OCDS becomes active. As a consequence, the module immediately stops all CAN activity (even within a running frame) and sets all transmit outputs to 1_B (recessive state). In this state, write accesses to the module in general, and read accesses to the CAN

RAM and most of the MultiCAN registers are no longer supported. A normal continuation when the suspend mode is left may not always be possible and may require a reset (e.g. depending on error counters).

PVC_X.H001 PVC Threshold Level 2

The Power Validation Circuits (PVCM, PVC1) compare the supply voltage of the respective domain (DMP_M, DMP_1) with programmable levels (LEV1V and LEV2V in register SCU_PVCMCON0 or SCU_PVC1CON0).

The default value of LEV1V is used to generate a reset request in the case of low core voltage.

LEV2V can generate an interrupt request at a higher voltage, to be used as a warning. Due to variations of the tolerance of both the Embedded Voltage Regulators (EVR) and the PVC levels, this interrupt can be triggered inadvertently, even though the core voltage is within the normal range. It is, therefore, recommended not to use this warning level.

LEV2V can be disabled by executing the following sequence:

1. Disable the PVC level threshold 2 interrupt request SCU_PVCMCON0.L2INTEN and SCU_ PVC1CON0.L2INTEN.
2. Disable the PVC interrupt request flag source SCU_INTDIS.PVCMI2 and SCU_INTDIS.PVC1I2.
3. Clear the PVC interrupt request flag source SCU_DMPMITCLR.PVCMI2 and SCU_ DMPMITCLR.PVC1I2.
4. Clear the PVC interrupt request flag by writing to SCU_INTCLR.PVCMI2 and SCU_INTCLR.PVC1I2.
5. Clear the selected SCU request flag (default is SCU_1IC.IR).

The Power Validation Circuits (PVCM) compare the supply voltage of the respective domain (DMP_M) with programmable levels (LEV1V and LEV2V in register SCU_PVCMCON0).

The default value of LEV1V is used to generate a reset request in the case of low core voltage.

LEV2V can generate an interrupt request at a higher voltage, to be used as a warning. Due to variations of the tolerance of both the Embedded Voltage Regulators (EVR) and the PVC levels, this interrupt can be triggered inadvertently, even though the core voltage is within the normal range. It is, therefore, recommended not to use this warning level.

LEV2V can be disabled by executing the following sequence:

1. Disable the PVC level threshold 2 interrupt request SCU_PVCMCON0.L2INTEN.
2. Disable the PVC interrupt request flag source SCU_INTDIS.PVCMI2.

3. Clear the PVC interrupt request flag source SCU_DMPMITCLR.PVCMI2.
4. Clear the PVC interrupt request flag by writing to SCU_INTCLR.PVCMI2.
5. Clear the selected SCU request flag (default is SCU_1IC.IR).

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.

StartUp_X.H002 FCONCS0..FCONCS4 Registers are Always Configured in External Start-Up Mode

The Start-Up procedure in External Start Mode writes all the FCONCSx (x=0..4) registers, independently on the number of CS-outputs selected.This has no effect for the user because for unused CS lines the Start-Up procedure configures only the external bus type into the registers but does not enable the output(s) for CS-functionality and they are free available.

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.

USIC_AI.H002 Configuration of USIC Port Pins

Setting up alternate output functions of USIC port pins through Pn.IOCRy registers before enabling the USIC protocol (CCR.MODE = 0001_B, 0010_B, 0011_B or 0100_B) might lead to unintended spikes on these port pins. To avoid the unintended spikes, either of the following two sequences can be used to enable the protocol:

- Sequence 1:
  - Write the initial output value to the port pin through Pn_OMR
  - Enable the output driver for the general purpose output through Pn_IOCRx
  - Enable USIC protocol through CCR.MODE
  - Select the USIC alternate output function through Pn_IOCRx
- Sequence 2:
  - Enable USIC protocol through CCR.MODE
  - Enable the output driver for the USIC alternate output function through Pn_IOCRx

Similarly, after the protocol is established, switching off the USIC channel by reseting CCR.MODE directly might cause undesired transitions on the output pin. The following sequence is recommended:

- Write the passive output value to the port pin through Pn_OMR
- Enable the output driver for the general purpose output through Pn_IOCRx
- Disable USIC protocol through CCR.MODE

USIC_AI.H003 PSR.RXIDLE Cleared by Software

If PSR.RXIDLE is cleared by software, the USIC is not able to receive until the receive line is detected IDLE again (see User's Manual chapter Idle Time).

For UART based busses with higher traffic e.g. LIN it is possible that sometimes the next frame starts sending before PSR.RXIDLE is set 1_B by hardware again. This generates an error.

A solution is, that the PSR.RXIDLE bit is not cleared in software.

5.4 Documentation Updates

EBC_X.D001 Visibility of Internal LXBus Cycles on External Address Bus

EBC chapter “Access Control to LXBus Modules” receives the following correction:

In the first paragraph the term “read mode” is replaced by “tri-state mode”.

The following is added:

Despite the above mentioned measures, accesses to internal LXBus modules are to some extent reflected on the non-multiplexed address pins A[23:0] of the external bus.

1. During an internal LXBus access, the external address bus is tri-stated. The switch to tri-state mode occurs in the same cycle as the internal LXBus access. This may induce residual voltage which can lead to undefined logic levels on the address bus pins. Those in turn can cause unwanted switching activity on attached device input stages. Therefore attached devices should be equipped with an input hysteresis filter to avoid unwanted switching activity.
2. After an internal LXBus access is completed the address of the location accessed last on the LXBus becomes visible on the external address bus, unless an external bus cycle immediately follows the LXBus cycle. Due to this behavior, switching activity on the address bus can be observed even if no external access is active.

Note: A functional impact due to this behavior is not expected because external bus control signals are held inactive during the internal LXBus access.

ID_X.D001 Identification Register

Additional “Identification Registers” chapter for the Data Sheet.

Table 9 Identification Registers

Short Name	Value	Address	Notes
SCU_IDMANUF	1820_H	$00'F07E_H$	
SCU_IDCHIP	2601_H	$00'F07C_H$	
SCU_IDMEM	$30BF_H$	$00'F07A_H$	
SCU_IDPROG	1313_H	$00'F078_H$	
JTAG_ID	$0010'A083_H$	---	marking EES-AA
	$1010'A083_H$	---	marking EES-AB, ES-AB, AB, EES-AC, ES-AC or AC

RESET_X.D001 Reset Types of Trap Registers

The reset type of SCU registers TRAPDIS, TRAPSET and TRAPNP is an Application Reset.

In the next revision of the user's manual the reset type of this registers will be changed from a Power-on Reset to an Application Reset.

SCU_X.D007 SCU Interrupts Enabled After Reset

Additional information to the "SCU Interrupt Generation" chapter of the User's Manual:

Following a reset, the SCU interrupts are enabled by default (register SCU_INTDIS = 0000_H). This may lead to interrupt requests being triggered in the SCU immediately, even before user software has begun to execute. In the SCU, multiple interrupt sources are `ORed` to a common interrupt node of the CPU interrupt controller. Due to the "ORing" of multiple interrupt sources, only one interrupt request to the interrupt controller will be generated if multiple sources at the input of this OR gate are active at the same time. If user software enables an interrupt in the interrupt controller (SCU_xIC) which shares the same node as the SCU interrupt request active after reset, it may lead to the effect of suppressing the intended interrupt source. So, for all SCU interrupt sources which will not be used, make sure to disable the interrupt source (SCU_INTDIS) and clear any pending request flags (SCU_xIC.IR) before enabling interrupts in interrupt controller.

StartUp_X.D002 External Start-Up Mode Selection by Configuration Pins

In "Start-Up Mode Selection" section of chapter 10.1 of the user manual version 2.0 the following table should updated (changes are marked **red**).

Table 10 Start-Up Mode Configuration

Start-Up Mode	STSTAT.HWCFG Value[1]	Configuration Pins P10.[4:0][2]				
External Start	**0000'0000$_B$**	**0**	**x**	0	0	0

1) Bitfield HWCFG can be loaded from Port 10 or from bitfield SWCFG in register SWRSTCON.

2) x means that the level on the corresponding pin is irrelevant.

USIC_X.D003 USIC0 Channel 1 Connection DX0D and DOUT

The connection for USIC0 channel 1 **U0C1_DX0D** to **P2.3** and USIC0 channel 1 **U0C1_DOUT** to **P2.4** are not available. These connections are erroneously listed in the User's Manual V2.1 and Data Sheet V2.1 tables "Pin Definitions and Functions".

XTAL_X.D001 Input Voltage Amplitude V_{AX1} on XTAL1

In the Data Sheet section "External Clock Input Parameters" the following table and figure should be updated (major changes are marked **red**).

Table 11 Start-Up Mode Configuration

Parameter	Symbol	Values			Unit	Note / Test Condition
		Min.	Typ.	Max.		
Oscillator frequency	f_{osc} SR	4	-	40	MHz	Input=Clock Signal
		4	-	16	MHz	Input=crystal or ceramic resonator
Input voltage amplitude on XTAL1 1)	V_{AX1} SR	0.3 × V_{DDIM}	-	-	V	4 to 16 MHz
		0.4 × V_{DDIM}	-	-	V	16 to 25 MHz
		0.5 × V_{DDIM}	-	-	V	25 to 40MHz

1) The amplitude voltage V_{AX1} refers to the offset voltage V_{OFF}. This offset voltage must be stable during the operation and the resulting voltage peaks must remain within the limits defined by V_{IX1}.

Note: For crystal or ceramic resonator operation, it is strongly recommended to measure the oscillation allowance (negative resistance) in the final target system (layout) to determine the optimum parameters for oscillator operation.
The manufacturers of crystals and ceramic resonators offer an oscillator evaluation service. This evaluation checks the crystal/resonator specification limits to ensure a reliable oscillator operation.

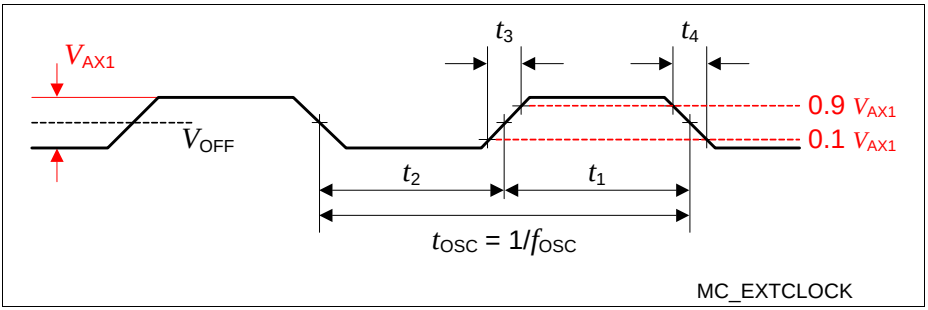

Figure 7 External Clock Drive XTAL1

www.infineon.com

03263AERRA

Published by Infineon Technologies AG