16/32-Bit

Architecture

XC27x8X Derivatives

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

Errata Sheet

V1.2 2013-07

Microcontrollers

Edition 2013-07

Published by
Infineon Technologies AG
81726 Munich, Germany

© 2013 Infineon Technologies AG
All Rights Reserved.

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

XC27x8X Derivatives

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

Errata Sheet

V1.2 2013-07

Microcontrollers

Table of Contents

1 History List / Change Summary

Table 1 History List

Version	Date	Remark1)
1.0	21.02.2011	First Errata Sheet release.
1.1	29.09.2011	Errata No. 02063AERRA, new Marking/Step (AA) added to Errata Sheet.
1.2	09.07.2013	Errata No. 02664AERRA.

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

Trademarks

C166™, TriCore™ and DAVE™ 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 XC27x8X 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.

Some errata of this Errata Sheet do not refer to all of the XC27x8X Derivatives, please look to the overview:

Table 2 for Functional Deviations

Table 3 for Deviations from Electrical and Timing Specification

Table 4 for Application Hints

Table 5 for Documentation Updates

3 Current Documentation

The Infineon XC2000 Family comprises device types from the XC2200 group, the XC2300 group and the XC2700 group. The XC27x8X device types belong to the XC2700 group.

Device	XC27x8X
Marking/Step	EES-AA, ES-AA, AA
Package	PG-LQFP-100, PG-LQFP-144

This Errata Sheet refers to the following documentation:

- XC27x8X Derivatives User’s Manual
- XC2768X Data Sheet
- XC2788X 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/xc2700**.

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	**XC27x8X**	
	EES-AA **ES-AA**	**AA1)**
ADC_AI.002	X	X
ADC_X.001	X	
ADC_X.002	X	X
BROM_TC.006	X	X
ESR_X.002	X	X
ESR_X.004	X	X
GPT12E_X.002	X	X
OCDS_X.003	X	X
PAD_X.001	X	X
RESET_X.003	X	X
RESET_X.004	X	X
SCU_X.012	X	X
StartUp_X.003	X	
USIC_AI.004	X	X
USIC_AI.005	X	X

Table 2 Errata Device Overview: Functional Deviations (cont'd)

Functional Deviation	XC27x8X	
	EES-AA ES-AA	AA[1]
USIC_AI.016	X	X
USIC_AI.018	X	X
WDT_X.002	X	X

[1] From EES/ES-AA step to AA step, 3 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	XC27x8X	
	EES-AA ES-AA	AA[1]
FLASH_X.P001	**X**	**X**
SWD_X.P001	**X**	
SWD_X.P002	**X**	**X**

1) From EES/ES-AA step to AA step, 3 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	XC27x8X	
	EES-AA ES-AA	AA1)
ADC_AI.H002	X	X
ADC_AI.H003	X	X
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
INT_X.H004	X	X
MultiCAN_AI.H005	X	X
MultiCAN_AI.H006	X	X
MultiCAN_AI.H007	X	X
MultiCAN_AI.H008	X	X
MultiCAN_TC.H002	X	X
MultiCAN_TC.H003	X	X
MultiCAN_TC.H004	X	X
OCDS_X.H003	X	X
PVC_X.H001	X	X
RESET_X.H003	X	X
RTC_X.H003	X	X

Table 4 Errata Device Overview: Application Hints (cont'd)

Hint	XC27x8X	
	EES-AA ES-AA	AA[1]
SCU_X.H009	X	X
SWD_X.H001	X	X
USIC_AI.H001	X	X
USIC_AI.H002	X	X
USIC_AI.H003	X	X

1) From EES/ES-AA step to AA step, 3 errata have been fixed.

4.4 Documentation Updates

Table 5 shows the dependencies oft documentation updates in the derivatives.

Table 5 Errata Device Overview: Documentation Updates

Hint	XC27x8X	
	EES-AA ES-AA	AA1)
EBC_X.D001	X	X

1) From EES/ES-AA step to AA step, 3 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 6 shows a short description of the functional deviations.

Table 6 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
ADC_AI.002	Result of Injected Conversion may be wrong	New	21
ADC_X.001	Cross-Current between VAREF and VAGND		21
ADC_X.002	Current Drawn on VAREF Pin can be Unexpected High		22
BROM_TC.006	Baud Rate Detection for CAN Bootstrap Loader		22
ESR_X.002	ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access		23
ESR_X.004	Wrong Value of SCU_RSTCONx Registers after ESRy Application Reset	New	25
GPT12E_X.002	Effects of GPT Module Microarchitecture		26
OCDS_X.003	Peripheral Debug Mode Settings cleared by Reset		27
PAD_X.001	Additional Edges in the Input Signal	New	28
RESET_X.003	P2.[2:0] and P10.[12:0] Switch to Input		33
RESET_X.004	Sticky “Register Access Trap” forces device into power-save mode after reset.	New	33
SCU_X.012	Wake-Up Timer RUNCON Command		34
StartUp_X.003	Debug Interface Configuration from Flash can Fail Upon Power-On		34

Table 6 Functional Deviations (cont'd)

Functional Deviation	**Short Description**	**Chg**	**Pg**
USIC_AI.004	**Receive shifter baudrate limitation**		**35**
USIC_AI.005	**Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time**		**35**
USIC_AI.016	**Transmit parameters are updated during FIFO buffer bypass**	New	**36**
USIC_AI.018	**Clearing PSR.MSLS bit immediately deasserts the SELOx output signal**	New	**37**
WDT_X.002	**Clearing the Internal Flag which Stores Preceding WDT Reset Request**		**37**

5.2 Deviations from Electrical and Timing Specification

Table 7 shows a short description of the electrical and timing deviations from the specification.

Table 7 Deviations from Electrical and Timing Specification

AC/DC/ADC Deviation	Short Description	Chg	Pg
FLASH_X.P001	Test Condition for Flash parameter NER in Data Sheets	New	40
SWD_X.P001	Supply Watchdog Level VSWD_min too Low		40
SWD_X.P002	Supply Watchdog (SWD) Supervision Level in Data Sheet.	New	41

5.3 Application Hints

Table 8 shows a short description of the application hints.

Table 8 Application Hints

Hint	Short Description	Chg	Pg
ADC_AI.H002	Minimizing Power Consumption of an ADC Module		42
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion		42
CAPCOM12_X.H001	Enabling or Disabling Single Event Operation		43
CC6_X.H001	Modifications of Bit MODEN in Register CCU6x_KSCFG		45
ECC_X.H001	ECC Error Indication Permanently Set		45
GPT12E_X.H002	Reading of Concatenated Timers		45
INT_X.H002	Increased Latency for Hardware Traps		47
INT_X.H004	SCU Interrupts Enabled After Reset		47
MultiCAN_AI.H005	TxD Pulse upon short disable request	Update	48
MultiCAN_AI.H006	Time stamp influenced by resynchronization		48
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off		48
MultiCAN_AI.H008	Effect of CANDIS on SUSACK		49
MultiCAN_TC.H002	Double Synchronization of receive input		49
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		50
MultiCAN_TC.H004	Double remote request		50
OCDS_X.H003	Debug Interface Configuration by User Software		51
PVC_X.H001	PVC Threshold Level 2		51

Table 8 Application Hints (cont'd)

Hint	Short Description	Chg	Pg
RESET_X.H003	How to Trigger a PORST after an Internal Failure		52
RTC_X.H003	Changing the RTC Configuration		52
SCU_X.H009	WUCR.TTSTAT can be set after a Power-Up		53
SWD_X.H001	Application Influence on the SWD		53
USIC_AI.H001	FIFO RAM Parity Error Handling		53
USIC_AI.H002	Configuration of USIC Port Pins	New	54
USIC_AI.H003	PSR.RXIDLE Cleared by Software	New	55

5.4 Documentation Updates

Table 9 shows a short description of the documentation updates.

Table 9 Documentation Updates

AC/DC/ADC Deviation	Short Description	Chg	Pg
EBC_X.D001	**Visibility of Internal LXBus Cycles on External Address Bus**	New	**56**

6 Detailed Errata Description

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

6.1 Functional Deviations

ADC_AI.002 Result of Injected Conversion may be wrong

In cancel-inject-repeat mode (`GxARBPR.CSM*` = 1_B), the result of the higher prioritized injected conversion c_H may be wrong if it was requested within a certain time window at the end of a lower prioritized conversion c_L. The width of the critical window depends on the divider factor `DIVA` for the analog internal clock.

In cancel-inject-repeat mode (`RSPR0.CSM*` = 1_B), the result of the higher prioritized injected conversion c_H may be wrong if it was requested within a certain time window at the end of a lower prioritized conversion c_L. The width of the critical window depends on the divider factor `DIVA` for the analog internal clock.

Workaround

Do not use cancel-inject-repeat mode. Instead, use wait-for-start mode (`GxARBPR.CSM*` = 0_B).

Do not use cancel-inject-repeat mode. Instead, use wait-for-start mode (`RSPR0.CSM*` = 0_B).

ADC_X.001 Cross-Current between V_{AREF} and V_{AGND}

The Early Engineering Samples (marked EES) and Engineering Samples (marked ES) draw a cross-current during power-on reset ($\overline{\text{PORST}}$ = V_{SS}) and in standby mode.

Other operating modes are not affected. Later product versions have this problem fixed.

The cross-current depends on the applied reference voltage, see table below.

Table 10 Typical Current Values

V_{AGND} / V	V_{AREF} / V	$I_{AREF\ AGND}$ / mA
0	5.5	17.0
0	5.0	14.8
0	4.5	12.8
0	3.3	7.8
0	3.0	6.6

Workaround

None

ADC_X.002 Current Drawn on V_{AREF} Pin can be Unexpected High

After Power-On with active $\overline{\text{PORST}}$ ($\overline{\text{PORST}}$ = V_{SS}) it can happen that the internal pull-up and/or a pull-down on the V_{AREF} pin are activated randomly. This has no functional impact but leads to a current consumption (<< 1 mA) which is higher than expected during the $\overline{\text{PORST}}$ = V_{SS} period. Once $\overline{\text{PORST}}$ is changed to the high level, the internal pulls at V_{AREF} are disabled. A next enable of the pulls can occur only with the next Power-On.

Workaround

Release $\overline{\text{PORST}}$ for a short time to a high level.

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.

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

1. Make sure that the trigger signals are still active when the associated service routine runs, so the trigger source can be evaluated by software.
2. Disable write access to registers `CLRESREXSTAT1/2` by clearing bit 8 at word address 00'F008$_H$. Use a read-modify-write sequence for this purpose to exclude other bits from this modification.
   To clear the status bits, write access to registers `CLRESREXSTAT1/2` can be enabled by setting bit 8 at word location 00'F008$_H$. Use a read-modify-write sequence for this purpose to exclude other bits from this modification.
   Write access is enabled by default.

ESR_X.003 Some Alternate ESRx Inputs Disabled in Stand-By Mode

The ESR1 and ESR2 inputs can be mapped to alternate input ports. Mappings are controlled by the ESREXCON1 and ESREXCON2 registers.

When in stand-by mode the input port mappings marked DISABLED in the following table are switched off. Consequently these ports can not be used as wakeup trigger inputs.

Table 11 Tables for XC2269I and XC228xI - ESREXCON1

ESREXCON1 selection	ESR1 Port	XC2269I	XC228xI
Input 0	Port 2.4	OK	OK
Input 1	Port 3.0	unused	DISABLED
Input 2	Port 10.0	OK	OK
Input 3	Port 1.0	unused	OK
Input 4	Port 1.2	unused	OK
Input 5	Port 2.1	OK	OK
Input 6	Port 6.1	OK	OK
Input 7	Port 11.0	unused	DISABLED
Input 8	Port 4.1	DISABLED	OK
Input 9	Port 10.4	OK	OK
Input 10	Port 2.5	OK	OK
Input 11	Port 0.0	DISABLED	OK

Table 12 Tables for XC2269I and XC228xI - ESREXCON2

ESREXCON2 selection	ESR2 Port	XC2269I	XC228xI
Input 0	Port 2.3	OK	OK
Input 1	Port 7.0	OK	OK
Input 2	Port 10.14	OK	OK
Input 3	Port 1.1	DISABLED	OK
Input 4	Port 1.3	DISABLED	OK
Input 5	Port 2.2	OK	OK
Input 6	Port 2.6	OK	OK
Input 7	Port 2.7	OK	OK
Input 8	Port 0.4	DISABLED	OK
Input 9	XTAL1	OK	OK

Table 12 Tables for XC2269I and XC228xI - ESREXCON2 (cont'd)

ESREXCON2 selection	ESR2 Port	XC2269I	XC228xI
Input 10	Port 4.5	unused	DISABLED
Input 11	Port 10.8	OK	OK

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 $f_{\text{critical window}}$ is less than 13 cycles.

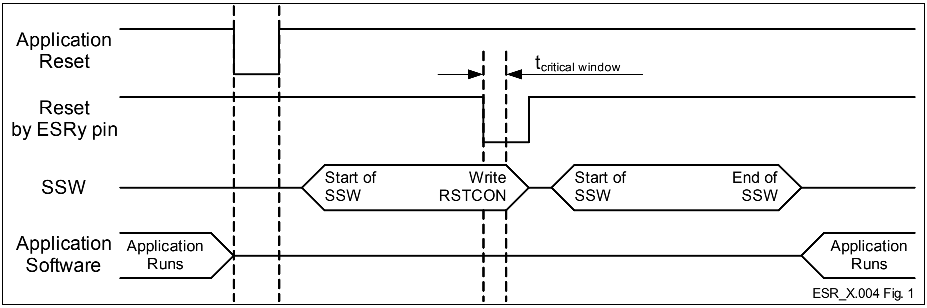

Figure 1 Critical application reset sequence

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.

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.

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.

PAD_X.001 Additional Edges in the Input Signal

The digital input- and I/O-pins are designed using Schmitt trigger input structures with hysteresis. Even with this structure, it is possible that very slow rising edges may generate spikes, resulting in unexpected additional edges at the input signal.

The next picture **Figure 2** is an example for a slow input signal, with spikes shown on the slow rising input signal.

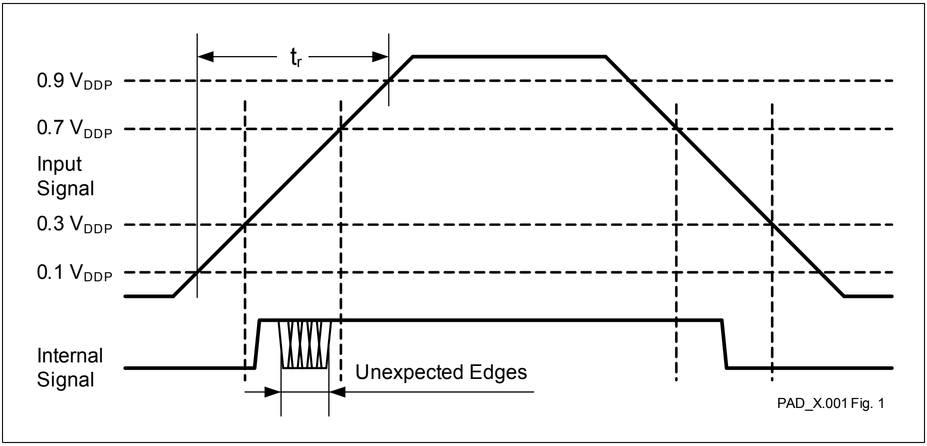

Figure 2 Example for a Slow Input Signal

The first rising edge in **Figure 2** of the internal signal is always valid. The edges which are marked with “Unexpected Edges” must be ignored.

Measurements have shown that a spike can be generated under the following conditions.

Table 13 Conditions for Additional Edges in the Input Signal

Parameter	Symbol	Typ. Value	Unit	Note
Digital supply voltage	V_{DDP}	4.5 to 5.5	V	Upper Voltage Range
Junction Temperature	T_{J}	full range	°C	
System frequency	f_{sys}	all	MHz	
Rising Slope	t_{r}	>1	µs	

The reaction to this spike generation strongly depends on the application (hardware, software, internal and external noise). Although it is not possible to define how the application will react in all cases, it is possible to categorize how applications are typically affected, as shown below.

Applications which can be affected by a spike:

- CCU6, CAPCOM2 and GPT inputs
- Port inputs
- Interrupts input

Applications which should not be affected by a spike due to faster rising slope t_r which is necessary for the application, due the interface protocol or due multiple sampling of the hardware:

- USIC
- CAN
- Others

Workaround for Input Capture Conditions

1. Workaround for all Affected Applications

Use rising edges with faster rising slope t_r than defined in **Table 13**.

2. Workaround for CCU6, CAPCOM2 and GPT Inputs

No generic solution is available for these applications. Add or switch to a software solution.

For example, the software could check whether the measured signal values are in the expected range.

3. Workaround for Port Inputs

1. The captured time interval value should be checked whether it is in a reasonable range.
2. The input pin should be read several times, and a majority decision may be made to decide whether the edge was correct or erratic.

Only if 1) and 2) indicate that the edge was reliable, the captured value should be used for further calculations. Otherwise, a substituted (extrapolated) value might be used.

4. Workaround for Interrupt Inputs

4.1 Falling Edge Detection Approach

1. Measure the time interval since the last interrupt (shown as in $t_{interval}$ in **Figure 3** below) and check that it is in the expected time range.
   In the example, an erratic edge would cause the measured time interval$t_{erratic\ interval}$ to be approximately 50% of the expected value.
2. The state of the input pin that caused the interrupt could be read several times in the interrupt service routine and a majority decision made to check if the input pin really is at a low level to determine whether this is a genuine falling edge interrupt or whether the interrupt was triggered by a spike generated by a slow rising edge.

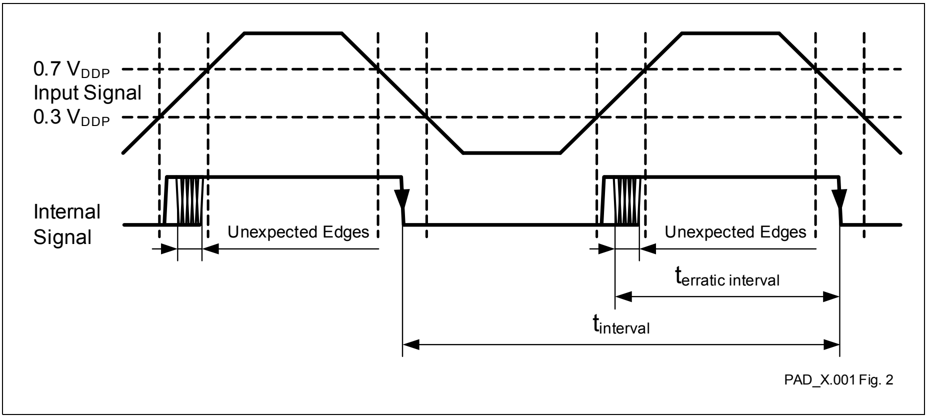

Figure 3 Falling Edge Detection Approach

Only if 1) and/or 2) indicate that the edge was reliable, should the rest of the interrupt service routine be executed.

4.2 Rising Edge Detection Approach

In case of rising edge detection multiple interrupts would be generated when the spike occurs. The time interval since the last interrupt can be measured – if it is very small compared to the expected value, this would indicate a spike and the interrupt should be ignored.

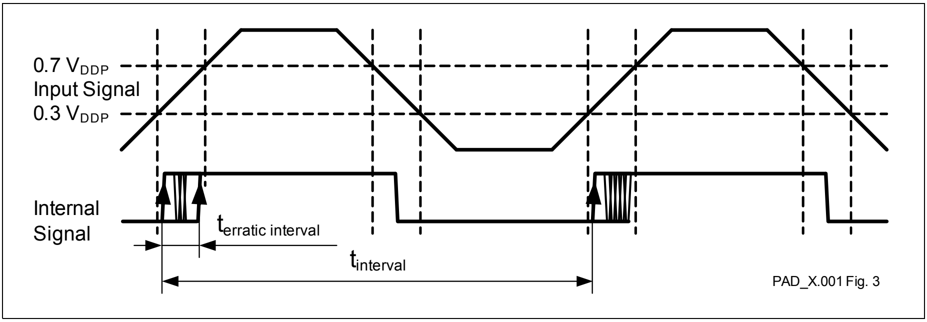

Figure 4 Rising Edge Detection Approach

If the time between the rising signal edge and the rising edge caused by a spike $t_{erratic}$ is less than the ISR service time, a workaround would be to clear the interrupt request (IR) flag before the return from interrupt is done. The clearing of the IR flag will avoid a further erratic interrupt.

The preferred solution for interrupt handling is to use the rising edge detection.

4.3 Rising Edge and Falling Edge Detection Approach

The rising edge detection workaround works also if both edges are used as trigger for interrupt and the following conditions are valid:

- $t_{erratic\ interval} << t_{interval\ r}$ and
- $t_{erratic\ interval} << t_{interval\ f}$

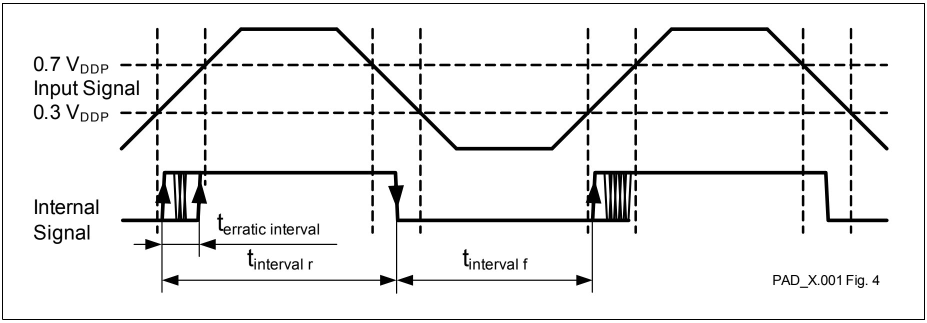

Figure 5 Rising Edge and Falling Edge Detection Approach

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.

SCU_X.012 Wake-Up Timer RUNCON Command

The Wake-Up Timer can be started and stopped by the `WUCR.RUNCON` bit field.

Under the precondition that the Wake-Up Timer is configured to stop when reaching zero (`WUCR.ASP`=1_B) and if two Wake-Up Timer commands are executed successively (e.g. “start” is directly followed by “stop”) then the second command will be ignored and will not change the state of the Wake-Up Timer.

Workaround

After executing the first command wait at least 4 Wake-Up Timer cycles (f_{WUT}) before writing again to the `WUCR.RUNCON` bit field and requesting the second command.

StartUp_X.003 Debug Interface Configuration from Flash can Fail Upon Power-On

XC2000/XE166 devices allow the user to select between a number of debug interface options including type (JTAG/DAP) and pin assignment.

The primary selection is done by configuration pins upon power-on, where one of the supported options is to install the debug interface according to the value taken from dedicated locations in user Flash ($C001F0_H$..$C001F3_H$). This option is selected by configuration pin values (HWCFG) $xxxxx111_B$ (code start from internal Flash) or $x1100000_B$ (code start from external memory). The other configurations directly selecting a debug mode work correctly.

The start-up procedure reads the dedicated locations in Flash too early - before Flash redundancy is installed - which can lead to an unrecoverable read error and terminate the boot process. A limited number of devices is affected – rough estimation is below 1% from the production - and the (mis) behavior is constant. That means any device is either always error free or always failing.

Note, that only the two mentioned modes upon power-on and only the read from dedicated locations during start-up are affected but not in general Flash and debug interface functionality.

Workaround

None.

If these start-up configurations are used during development, a device that does not start-up in the desired debug configuration should be replaced by another device.

Alternatively, select a debug interface not from Flash data but directly using configuration pins - refer to the User’s Manual. With this it is not possible to start from external memory, nor is JTAG position A available.

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.

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

FLASH_X.P001 Test Condition for Flash parameter N_{ER} in Data Sheets

The Flash endurance parameter N_{ER} `Number of erase cycles` for 15000 cycles is documented with a wrong Test Condition.

The Test Condition states today: `$t_{RET}$≥5 years; Valid for up to 64 user selected sectors (data storage)`.

In fact the amount of Flash memory validated for this cycling rate is more limited and the Test Condition must therefore state the following:

- t_{RET}≥5 years; Valid for Flash module 4 (up to 64 kbytes)

Note: The related use case for this parameter is data storage with high cycling rate in general and EEPROM emulation in particular. For these applications concurrent operation of data storage to and program execution from Flash is assumed. Refer also to parameter N_{PP}.

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.

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.

6.3 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.

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

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

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

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

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

   with

   a) N_A = ratio f_{ADC}/f_{ADCI} (N_A = 4 .. 63, as defined in bit field `DIVA`),

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

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

All bit fields mentioned above are located in register `GLOBCTR`.

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

Workaround 1

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

Workaround 2

Select the parameters in register `GLOBCTR` according to the following relation:

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

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 14** 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 14**:

Table 14 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.

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

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

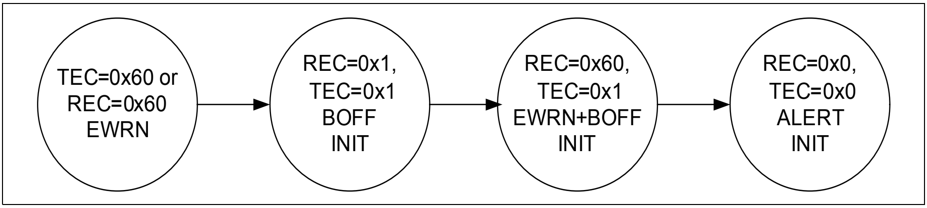

Figure 6 Alert Interrupt Behavior in case of Bus-Off

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.

According to specification CANDIS shall have no influence on SUSACK. 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.

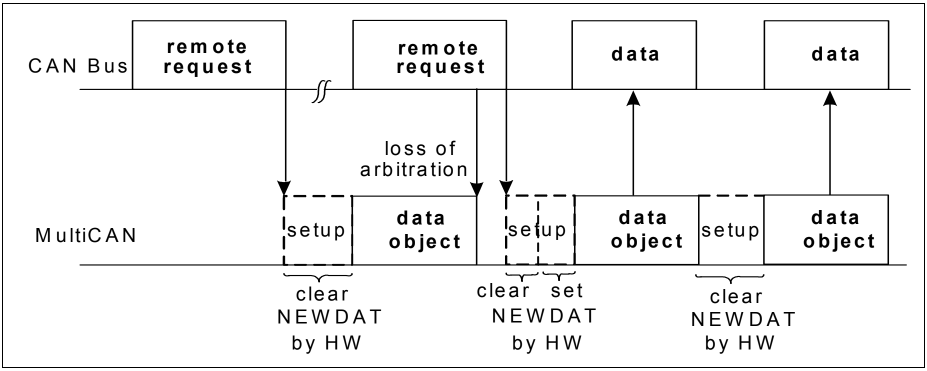

Figure 7 Loss of Arbitration

OCDS_X.H003 Debug Interface Configuration by User Software

If the debug interface must be (re)configured, the sequence of actions to follow is:

1. activate internal test-logic reset by installing `SCU_DBGPRR.TRSTGT`=0
2. disable debug interface by installing `SCU_DBGPRR.DBGEN`=0
3. install desired debug interface configuration in `SCU_DBGPRR[11:0]`
4. activate pull-devices (if internal will be used) by installing `Px_IOCRy` accordingly
5. enable debug interface by installing `SCU_DBGPRR.DBGEN`=1
6. release internal test-logic reset by installing `SCU_DBGPRR.TRSTGT`=1

These steps must be performed as separate, sequential write operations.

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`).

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.

SCU_X.H009 `WUCR.TTSTAT` can be set after a Power-Up

After power-up the wake-up clock f_{WU} is selected for the Wake-Up Timer (WUT). In this case, the trim interrupt trigger cannot be used, because the WUT trim trigger status bit (`WUCR.TTSTAT`) might become set erroneously. This happens sporadically and is, therefore, difficult to find in the development phase of an application. If the trim interrupt trigger is enabled this may lead to unintended SCU interrupts that may also block other interrupt sources (see **INT_X.H004**).

This can be avoided by executing the following sequence:

1. Disable the trim interrupt source `SCU_INTDIS.WUTI`
2. Clear the trim interrupt request flag by writing to `INTCLR.WUTI`
3. Clear the selected SCU request flag (default is `SCU_1IC.IR`)

SWD_X.H001 Application Influence on the SWD

The internal Supply Watchdog (SWD) monitors the external supply voltage of the pad I/O domain V_{DDPB} which is connected to the device. Therefore, adjustable threshold levels are defined over the complete supply voltage range. These limits are also influenced by system environment and may deviate due to external influences slightly from the values given in the Datasheet. Independent of the SWD is the internal start up and operation protected by the PVC, which monitor the core voltage.

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.

6.4 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 the external bus chip select signals ($\overline{CSx}$) are held inactive and control signals (ALE/$\overline{BHE}$/$\overline{RD}$/$\overline{WR}$) are switched off (tri-state) during the internal LXBus access.

www.infineon.com

02664AERRA

Published by Infineon Technologies AG