

# 16/32-Bit

Architecture

## XC2300C Derivatives

16/32-Bit Single-Chip Microcontroller  
with 32-Bit Performance

XC2000 Family / High Line

Errata Sheet

V1.6 2013-03

Microcontrollers

**Edition 2013-03**

**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](http://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

## XC2300C Derivatives

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

Errata Sheet

V1.6 2013-03

## Table of Contents

<b>1</b>	<b>History List / Change Summary</b>	<b>7</b>
<b>2</b>	<b>General</b>	<b>8</b>
<b>3</b>	<b>Current Documentation</b>	<b>10</b>
<b>4</b>	<b>Errata Device Overview</b>	<b>11</b>
4.1	Functional Deviations	11
4.2	Deviations from Electrical and Timing Specification	14
4.3	Application Hints	15
4.4	Documentation Updates	17
<b>5</b>	<b>Short Errata Description</b>	<b>18</b>
5.1	Functional Deviations	18
5.2	Deviations from Electrical and Timing Specification	21
5.3	Application Hints	22
5.4	Documentation Updates	24
<b>6</b>	<b>Detailed Errata Description</b>	<b>25</b>
6.1	Functional Deviations	25
	BROM_TC.006	25
	BSL_CAN_X.001	25
	ECC_X.002	26
	ESR_X.002	27
	ESR_X.004	27
	FlexRay_AI.087	28
	FlexRay_AI.088	29
	FlexRay_AI.089	29
	FlexRay_AI.090	30
	FlexRay_AI.091	31
	FlexRay_AI.092	32
	FlexRay_AI.093	32
	FlexRay_AI.094	33
	FlexRay_AI.095	34
	FlexRay_AI.096	35
	FlexRay_AI.097	35
	FlexRay_AI.098	36
	FlexRay_AI.099	36
	FlexRay_AI.100	37
	FlexRay_AI.101	38
	FlexRay_AI.102	38
	FlexRay_AI.103	39
	FlexRay_X.001	40

	GPT12E_X.002	40
	OCDS_X.003	42
	PAD_X.001	42
	PARITY_X.001	47
	RESET_X.003	47
	RESET_X.004	47
	SCU_X.012	48
	StartUp_X.003	48
	USIC_AI.004	50
	USIC_AI.005	50
	USIC_AI.016	50
	WDT_X.002	51
6.2	Deviations from Electrical- and Timing Specification	53
	FLASH_X.P001	53
	SWD_X.P001	53
	SWD_X.P002	54
6.3	Application Hints	55
	ADC_AI.H002	55
	CAPCOM12_X.H001	55
	CC6_X.H001	57
	ECC_X.H001	57
	FlexRay_AI.H004	57
	FlexRay_AI.H005	58
	FlexRay_AI.H006	58
	FlexRay_AI.H007	58
	FlexRay_AI.H009	59
	GPT12_AI.H001	59
	GPT12E_X.H002	60
	INT_X.H002	61
	INT_X.H004	61
	MultiCAN_AI.H005	62
	MultiCAN_AI.H006	62
	MultiCAN_AI.H007	62
	MultiCAN_AI.H008	63
	MultiCAN_TC.H002	63
	MultiCAN_TC.H003	64
	MultiCAN_TC.H004	64
	OCDS_X.H003	65
	PVC_X.H001	65
	RESET_X.H003	67
	RTC_X.H003	67
	SCU_X.H009	67
	SWD_X.H001	68

	USIC_AI.H001 .....	68
	USIC_AI.H002 .....	68
	USIC_AI.H003 .....	69
6.4	Documentation Updates .....	70
	EBC_X.D001 .....	70
	ECC_X.D002 .....	70
	RESET_X.D001 .....	71

# 1 History List / Change Summary

**Table 1 History List**

Version	Date	Remark <sup>1)</sup>
1.0	30.09.2008	new Errata Sheet
1.1	30.01.2009	
1.2	13.03.2009	new Marking/Step ES-AA added to Errata Sheet, new Errata Sheet layout
1.3	01.03.2010	Errata No. 01715AERRA, new Marking/Step (ES+AA) added to Errata Sheet.
1.4	29.06.2010	Errata No. 01810AERRA, new Marking/Step (ES-AB) added to Errata Sheet.
1.5	23.09.2010	Errata No. 01878AERRA, new Marking/Step (AB) added to Errata Sheet.
1.6	22.03.2013	Errata No. 02533AERRA.

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](mailto:mcdocu.comments@infineon.com)



## 2 General

This Errata Sheet describes the deviations of the XC2300C Derivatives from the current user documentation.

Each erratum identifier follows the pattern **Module\_Arch.TypeNumber**:

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

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

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



Some errata of this Errata Sheet do not refer to all of the XC2300C 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 XC23xxC device types belong to the XC2300 group.

Device	XC23xxC
Marking/Step	EES-AA, ES-AA, ES+AA, ES-AB, AB
Package	PG-LQFP-144

This Errata Sheet refers to the following documentation:

- XC2300C Derivatives User's Manual
- XC238xC 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/xc2300](http://www.infineon.com/xc2300).

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	XC23xxC		
	EES-AA ES-AA	ES+AA <sup>1)</sup> all AB <sup>2)</sup>	
<b>BROM_TC.006</b>	<b>X</b>	<b>X</b>	
<b>BSL_CAN_X.001</b>	<b>X</b>	<b>X</b>	
<b>ECC_X.002</b>	<b>X</b>		
<b>ESR_X.002</b>	<b>X</b>	<b>X</b>	
<b>ESR_X.004</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.087</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.088</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.089</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.090</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.091</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.092</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.093</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.094</b>	<b>X</b>	<b>X</b>	
<b>FlexRay_AI.095</b>	<b>X</b>	<b>X</b>	

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

Functional Deviation	XC23xxC		
	EES-AA ES-AA	ES+AA <sup>1)</sup> all AB <sup>2)</sup>	
FlexRay_AI.096	X	X	
FlexRay_AI.097	X	X	
FlexRay_AI.098	X	X	
FlexRay_AI.099	X	X	
FlexRay_AI.100	X	X	
FlexRay_AI.101	X	X	
FlexRay_AI.102	X	X	
FlexRay_AI.103	X	X	
FlexRay_X.001	X	X	
GPT12E_X.002	X	X	
OCDS_X.003	X	X	
PAD_X.001	X	X	
PARITY_X.001	X	X	
RESET_X.003	X	X	
RESET_X.004	X	X	
SCU_X.012	X	X	
StartUp_X.003	X	X	
USIC_AI.004	X	X	
USIC_AI.005	X	X	
USIC_AI.016	X	X	
WDT_X.002	X	X	

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

- 2) Valid for all AB steps encloses ES-AB and AB step.

## 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	XC23xxC		
	EES-AA	ES-AA <sup>1)</sup> all AB <sup>2)</sup>	
FLASH_X.P001	X	X	
SWD_X.P001	X		
SWD_X.P002	X	X	

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

2) Valid for all AB steps encloses ES-AB and AB step.

## 4.3 Application Hints

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

**Table 4 Errata Device Overview:  
Application Hints**

Hint	XC23xxC		
	EES-AA ES-AA	ES+AA <sup>(1)</sup> all AB <sup>(2)</sup>	
<a href="#">ADC_AI.H002</a>	X	X	
<a href="#">CAPCOM12_X.H001</a>	X	X	
<a href="#">CC6_X.H001</a>	X	X	
<a href="#">ECC_X.H001</a>	X	X	
<a href="#">FlexRay_AI.H004</a>	X	X	
<a href="#">FlexRay_AI.H005</a>	X	X	
<a href="#">FlexRay_AI.H006</a>	X	X	
<a href="#">FlexRay_AI.H007</a>	X	X	
<a href="#">FlexRay_AI.H009</a>	X	X	
<a href="#">GPT12_AI.H001</a>	X	X	
<a href="#">GPT12E_X.H002</a>	X	X	
<a href="#">INT_X.H002</a>	X	X	
<a href="#">INT_X.H004</a>	X	X	
<a href="#">MultiCAN_AI.H005</a>	X	X	
<a href="#">MultiCAN_AI.H006</a>	X	X	
<a href="#">MultiCAN_AI.H007</a>	X	X	
<a href="#">MultiCAN_AI.H008</a>	X	X	
<a href="#">MultiCAN_TC.H002</a>	X	X	
<a href="#">MultiCAN_TC.H003</a>	X	X	

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

Hint	XC23xxC		
	EES-AA	ES-AA <sup>1)</sup> all AB <sup>2)</sup>	
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	
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 ES+AA/ES-AB/AB step, 2 errata have been fixed.

2) Valid for all AB steps encloses ES-AB and AB step.



## 4.4 Documentation Updates

**Table 5** shows the dependencies of documentation updates in the derivatives.

**Table 5 Errata Device Overview:  
Documentation Updates**

Documentation Updates	XC23xxC		
	EES-AA ES-AA	ES+AA <sup>1)</sup> all AB <sup>2)</sup>	
<a href="#">EBC_X.D001</a>	X	X	
<a href="#">ECC_X.D002</a>	X	X	
<a href="#">RESET_X.D001</a>	X	X	

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

2) Valid for all AB steps encloses ES-AB and AB step.

## 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
<b>BROM_TC.006</b>	<b>Baud Rate Detection for CAN Bootstrap Loader</b>	New	<b>25</b>
<b>BSL_CAN_X.001</b>	<b>Quartz Crystal Settling Time after PORST too Long for CAN Bootstrap Loader</b>		<b>25</b>
<b>ECC_X.002</b>	<b>Incorrect ECC Error Indication for DPRAM</b>		<b>26</b>
<b>ESR_X.002</b>	<b>ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access</b>		<b>27</b>
<b>ESR_X.004</b>	<b>Wrong Value of SCU_RSTCONx Registers after ESrY Application Reset</b>	New	<b>27</b>
<b>FlexRay_AI.087</b>	<b>After reception of a valid sync frame followed by a valid non-sync frame in the same static slot the received sync frame may be ignored</b>		<b>28</b>
<b>FlexRay_AI.088</b>	<b>A sequence of received WUS may generate redundant SIR.WUPA/B events</b>		<b>29</b>
<b>FlexRay_AI.089</b>	<b>Rate correction set to zero in case of SyncCalcResult=MISSING_TERM</b>		<b>29</b>
<b>FlexRay_AI.090</b>	<b>Flag SFS.MRCS is set erroneously although at least one valid sync frame pair is received</b>		<b>30</b>

**Short Errata Description**

**Table 6**      Functional Deviations (cont'd)

<b>Functional Deviation</b>	<b>Short Description</b>	<b>Chg</b>	<b>Pg</b>
<b>FlexRay_AI.091</b>	<b>Incorrect rate and/or offset correction value if second Secondary Time Reference Point (STRP) coincides with the action point after detection of a valid frame</b>		<b>31</b>
<b>FlexRay_AI.092</b>	<b>Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00</b>		<b>32</b>
<b>FlexRay_AI.093</b>	<b>Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames</b>		<b>32</b>
<b>FlexRay_AI.094</b>	<b>Sync frame overflow flag EIR.SFO may be set if slot counter is greater than 1024</b>		<b>33</b>
<b>FlexRay_AI.095</b>	<b>Register RCV displays wrong value</b>		<b>34</b>
<b>FlexRay_AI.096</b>	<b>Noise following a dynamic frame that delays idle detection may fail to stop slot</b>		<b>35</b>
<b>FlexRay_AI.097</b>	<b>Loop back mode operates only at 10 MBit/s</b>		<b>35</b>
<b>FlexRay_AI.098</b>	<b>Suspend Mode is not functional</b>		<b>36</b>
<b>FlexRay_AI.099</b>	<b>Erroneous cycle offset during startup after abort of startup or normal operation</b>		<b>36</b>
<b>FlexRay_AI.100</b>	<b>First WUS following received valid WUP may be ignored</b>		<b>37</b>
<b>FlexRay_AI.101</b>	<b>READY command accepted in READY state</b>		<b>38</b>
<b>FlexRay_AI.102</b>	<b>Slot Status vPOC!SlotMode is reset immediately when entering HALT state</b>		<b>38</b>
<b>FlexRay_AI.103</b>	<b>Received messages not stored in Message RAM when in Loop Back Mode</b>	<b>New</b>	<b>39</b>
<b>FlexRay_X.001</b>	<b>Trigger User Reset after Power-On</b>	<b>New</b>	<b>38</b>
<b>GPT12E_X.002</b>	<b>Effects of GPT Module Microarchitecture</b>		<b>40</b>

**Short Errata Description**

**Table 6**      Functional Deviations (cont'd)

<b>Functional Deviation</b>	<b>Short Description</b>	<b>Chg</b>	<b>Pg</b>
<b>OCDS_X.003</b>	<b>Peripheral Debug Mode Settings cleared by Reset</b>		<b>42</b>
<b>PAD_X.001</b>	<b>Additional Edges in the Input Signal</b>	New	<b>42</b>
<b>PARITY_X.001</b>	<b>PMTSR Register Initialization</b>		<b>47</b>
<b>RESET_X.003</b>	<b>P2.[2:0] and P10.[12:0] Switch to Input</b>		<b>47</b>
<b>RESET_X.004</b>	<b>Sticky "Register Access Trap" forces device into power-save mode after reset.</b>	New	<b>47</b>
<b>SCU_X.012</b>	<b>Wake-Up Timer RUNCON Command</b>	New	<b>48</b>
<b>StartUp_X.003</b>	<b>Debug Interface Configuration from Flash can Fail Upon Power-On</b>	New	<b>48</b>
<b>USIC_AI.004</b>	<b>Receive shifter baudrate limitation</b>		<b>50</b>
<b>USIC_AI.005</b>	<b>Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time</b>		<b>50</b>
<b>USIC_AI.016</b>	<b>Transmit parameters are updated during FIFO buffer bypass</b>	New	<b>50</b>
<b>WDT_X.002</b>	<b>Clearing the Internal Flag which Stores Preceding WDT Reset Request</b>		<b>51</b>

## **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**

<b>AC/DC/ADC Deviation</b>	<b>Short Description</b>	<b>Chg</b>	<b>Pg</b>
<b>FLASH_X.P001</b>	<b>Test Condition for Flash parameter NER in Data Sheets</b>	New	<b>53</b>
<b>SWD_X.P001</b>	<b>Supply Watchdog Level VSWD_min too Low</b>		<b>53</b>
<b>SWD_X.P002</b>	<b>Minimizing Power Consumption of an ADC Module</b>	New	<b>54</b>

## 5.3 Application Hints

**Table 8** shows a short description of the application hints.

**Table 8** Application Hints

Hint	Short Description	Chg	Pg
<b>ADC_AI.H002</b>	<b>Minimizing Power Consumption of an ADC Module</b>		<b>55</b>
<b>CAPCOM12_X.H001</b>	<b>Enabling or Disabling Single Event Operation</b>		<b>55</b>
<b>CC6_X.H001</b>	<b>Modifications of Bit MODEN in Register CCU6x_KSCFG</b>		<b>57</b>
<b>ECC_X.H001</b>	<b>ECC Error Indication Permanently Set</b>		<b>57</b>
<b>FlexRay_AI.H004</b>	<b>Only the first message can be received in External Loop Back mode</b>		<b>57</b>
<b>FlexRay_AI.H005</b>	<b>Initialization of internal RAMs requires one eray_bclk cycle more</b>		<b>58</b>
<b>FlexRay_AI.H006</b>	<b>Transmission in ATM/Loopback mode</b>		<b>58</b>
<b>FlexRay_AI.H007</b>	<b>Reporting of coding errors via TEST1.CERA/B</b>		<b>58</b>
<b>FlexRay_AI.H009</b>	<b>Return from test mode operation</b>		<b>59</b>
<b>GPT12_AI.H001</b>	<b>Modification of Block Prescalers BPS1 and BPS2</b>	New	<b>59</b>
<b>GPT12E_X.H002</b>	<b>Reading of Concatenated Timers</b>		<b>60</b>
<b>INT_X.H002</b>	<b>Increased Latency for Hardware Traps</b>		<b>61</b>
<b>INT_X.H004</b>	<b>SCU Interrupts Enabled After Reset</b>		<b>61</b>
<b>MultiCAN_AI.H005</b>	<b>TxD Pulse upon short disable request</b>		<b>62</b>
<b>MultiCAN_AI.H006</b>	<b>Time stamp influenced by resynchronization</b>		<b>62</b>
<b>MultiCAN_AI.H007</b>	<b>Alert Interrupt Behavior in case of Bus-Off</b>	New	<b>62</b>
<b>MultiCAN_AI.H008</b>	<b>Effect of CANDIS on SUSACK</b>	New	<b>63</b>
<b>MultiCAN_TC.H002</b>	<b>Double Synchronization of receive input</b>		<b>63</b>

**Short Errata Description**

**Table 8**      Application Hints (cont'd)

<b>Hint</b>	<b>Short Description</b>	<b>Chg</b>	<b>Pg</b>
<b>MultiCAN_TC.H003</b>	<b>Message may be discarded before transmission in STT mode</b>		<b>64</b>
<b>MultiCAN_TC.H004</b>	<b>Double remote request</b>		<b>64</b>
<b>OCDS_X.H003</b>	<b>Debug Interface Configuration by User Software</b>		<b>65</b>
<b>PVC_X.H001</b>	<b>PVC Threshold Level 2</b>	Update	<b>65</b>
<b>RESET_X.H003</b>	<b>How to Trigger a PORST after an Internal Failure</b>		<b>67</b>
<b>RTC_X.H003</b>	<b>Changing the RTC Configuration</b>		<b>67</b>
<b>SCU_X.H009</b>	<b>WUCR.TTSTAT can be set after a Power-Up</b>		<b>67</b>
<b>SWD_X.H001</b>	<b>Application Influence on the SWD</b>		<b>68</b>
<b>USIC_AI.H001</b>	<b>FIFO RAM Parity Error Handling</b>		<b>68</b>
<b>USIC_AI.H002</b>	<b>Configuration of USIC Port Pins</b>	New	<b>68</b>
<b>USIC_AI.H003</b>	<b>PSR.RXIDLE Cleared by Software</b>	New	<b>69</b>

## 5.4 Documentation Updates

**Table 9** gives a short description of the documentation updates.

**Table 9 Documentation Updates**

Documentation Updates	Short Description	Chg	Pg
<a href="#">EBC_X.D001</a>	<a href="#">Visibility of Internal LxBus Cycles on External Address Bus</a>	New	<a href="#">70</a>
<a href="#">ECC_X.D002</a>	<a href="#">Initialization of the Read-Control Logic</a>		<a href="#">70</a>
<a href="#">RESET_X.D001</a>	<a href="#">Reset Types of Trap Registers</a>		<a href="#">71</a>



## **6 Detailed Errata Description**

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

### **6.1 Functional Deviations**

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

##### **Workaround**

- For low performance CAN 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  $\overline{\text{PORST}}$ . Then switch the system clock to external source and trigger a software reset with CAN bootstrap loader mode selected. Now the

device starts with a CAN bootstrap loader without limitation of the oscillator settling time.

### **ECC X.002 Incorrect ECC Error Indication for DPRAM**

Under certain conditions, the ECC error flag for the dual port memory (DPRAM) may indicate an error when none exists.

Conditions under which ECC error is incorrectly indicated:

- ECC memory protection has been selected for DPRAM (`SCU_MCHKCON.SELDP = 0`).
- The ECC check is enabled for DPRAM (`SCU_ECCCON.DPEN = 1`).
- A concurrent read and write access is made to the same byte or word in the DPRAM (possible due to instruction pipeline).

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

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

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

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

### **Workaround**

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

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

**ESR\_X.002 ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access**

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

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

**Workaround**

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<sub>H</sub>. 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<sub>H</sub>. Use a read-modify-write sequence for this purpose to exclude other bits from this modification. Write access is enabled by default.

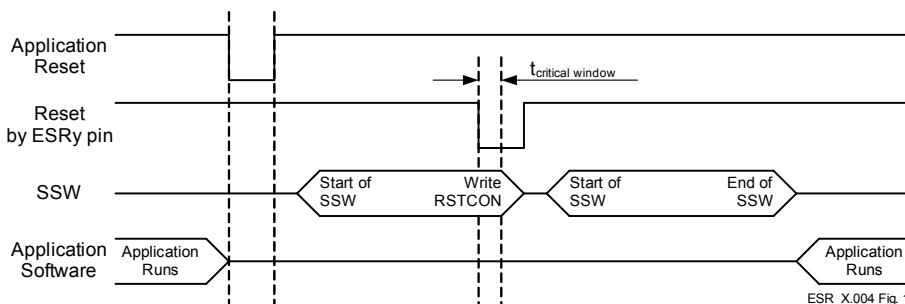
**ESR\_X.004 Wrong Value of SCU\_RSTCONx Registers after ESRy Application Reset**

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

## Detailed Errata Description

at the same time, the ESRy reset request is processed by the reset circuitry. The width of this critical window  $t_{\text{critical window}}$  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.

**FlexRay AI.087** After reception of a valid sync frame followed by a valid non-sync frame in the same static slot the received sync frame may be ignored

### Description:

If in a static slot of an even cycle a valid sync frame followed by a valid non-sync frame is received, and the frame valid detection (`prt_frame_decoded_on_X`) of the DEC process occurs one `sclk` after valid frame detection of FSP process (`fsp_val_syncfr_chx`), the sync frame is not taken into account by the CSP process (`devte_xxs_reg`).

### Scope:

The erratum is limited to the case where more than one valid frame is received in a static slot of an even cycle.

**Effects:**

In the described case the sync frame is not considered by the CSP process. This may lead to a SyncCalcResult of MISSING\_TERM (error flag SFS.MRCS set). As a result the POC state may switch to NORMAL\_PASSIVE or HALT or the Startup procedure is aborted.

**Workaround**

Avoid static slot configurations long enough to receive two valid frames.

**FlexRay\_AI.088 A sequence of received WUS may generate redundant SIR.WUPA/B events****Description:**

If a sequence of wakeup symbols (WUS) is received, all separated by appropriate idle phases, a valid wakeup pattern (WUP) should be detected after every second WUS. The E-Ray detects a valid wakeup pattern after the second WUS and then after each following WUS.

**Scope:**

The erratum is limited to the case where the application program frequently resets the appropriate SIR.WUPA/B bits.

**Effects:**

In the described case there are more SIR.WUPA/B events seen than expected.

**Workaround**

Ignore redundant SIR.WUPA/B events.

**FlexRay\_AI.089 Rate correction set to zero in case of SyncCalcResult=MISSING\_TERM****Description:**

**Detailed Errata Description**

In case a node receives too few sync frames for rate correction calculation and signals a SyncCalcResult of MISSING\_TERM, the rate correction value is set to zero instead to the last calculated value.

**Scope:**

The erratum is limited to the case of receiving too few sync frames for rate correction calculation (SyncCalcResult=MISSING\_TERM in an odd cycle).

**Effects:**

In the described case a rate correction value of zero is applied in NORMAL\_ACTIVE / NORMAL\_PASSIVE state instead of the last rate correction value calculated in NORMAL\_ACTIVE state. This may lead to a desynchronisation of the node although it may stay in NORMAL\_ACTIVE state (depending on gMaxWithoutClockCorrectionPassive) and decreases the probability to re-enter NORMAL\_ACTIVE state if it has switched to NORMAL\_PASSIVE (pAllowHaltDueToClock=false).

**Workaround**

It is recommended to set gMaxWithoutClockCorrectionPassive to 1. If missing sync frames cause the node to enter NORMAL\_PASSIVE state, use higher level application software to leave this state and to initiate a re-integration into the cluster. HALT state can also be used instead of NORMAL\_PASSIVE state by setting pAllowHaltDueToClock to true.

**FlexRay AI.090 Flag `SFS.MRCS` is set erroneously although at least one valid sync frame pair is received****Description:**

If in an odd cycle  $2c+1$  after reception of a sync frame in slot  $n$  the total number of different sync frames per double cycle has exceeded gSyncNodeMax and the node receives in slot  $n+1$  a sync frame that matches with a sync frame received in the even cycle  $2c$ , the sync frame pair is not taken into account by CSP process. This may cause the flags `SFS.MRCS` and `EIR.CCF` to be set erroneously.

**Scope:**

The erratum is limited to the case of a faulty cluster configuration where different sets of sync frames are transmitted in even and odd cycles and the total number of different sync frames is greater than gSyncNodeMax.

**Effects:**

In the described case the error interrupt flag `EIR.CCF` is set and the node may enter either the POC state `NORMAL_PASSIVE` or `HALT`.

**Workaround**

Correct configuration of gSyncNodeMax.

**FlexRay AI.091 Incorrect rate and/or offset correction value if second Secondary Time Reference Point (STRP) coincides with the action point after detection of a valid frame**

**Description:**

If a valid sync frame is received before the action point and additionally noise or a second frame leads to a STRP coinciding with the action point, an incorrect deviation value of zero is used for further calculations of rate and/or offset correction values.

**Scope:**

The erratum is limited to configurations with an action point offset greater than static frame length.

**Effects:**

In the described case a deviation value of zero is used for further calculations of rate and/or offset correction values. This may lead to an incorrect rate and/or offset correction of the node.

## Workaround

Configure action point offset smaller than static frame length.

### **FlexRay AI.092** Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00

Description:

The initial rate correction value as calculated in figure 8-8 of protocol spec v2.1 is zero if parameter pMicroInitialOffsetA,B was configured to be zero.

Scope:

The erratum is limited to the case where pMicroInitialOffsetA,B is configured to zero.

Effects:

Starting with an initial rate correction value of zero leads to an adjustment of the rate correction earliest 3 cycles later (see figure 7-10 of protocol spec v2.1). In a worst case scenario, if the whole cluster is drifting away too fast, the integrating node would not be able to follow and therefore abort integration.

## Workaround

Avoid configurations with pMicroInitialOffsetA,B equal to zero. If the related configuration constraint of the protocol specification results in pMicroInitialOffsetA,B equal to zero, configure it to one instead. This will lead to a correct initial rate correction value, it will delay the startup of the node by only one microtick.

### **FlexRay AI.093** Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames

Description:



**Detailed Errata Description**

If a node receives in an even cycle a startup frame after it has received more than gSyncNodeMax sync frames, this startup frame is added erroneously by process CSP to the number of valid startup frames (zStartupNodes). The faulty number of startup frames is delivered to the process POC. As a consequence this node may integrate erroneously to the running cluster because it assumes that it has received the required number of startup frames.

**Scope:**

The erratum is limited to the case of more than gSyncNodeMax sync frames.

**Effects:**

In the described case a node may erroneously integrate successfully into a running cluster.

**Workaround**

Use frame schedules where all startup frames are placed in the first static slots. gSyncNodeMax should be configured to be greater than or equal to the number of sync frames in the cluster.

**FlexRay AI.094 Sync frame overflow flag EIR.SFO may be set if slot counter is greater than 1024****Description:**

If in the static segment the number of transmitted and received sync frames reaches gSyncNodeMax and the slot counter in the dynamic segment reaches the value  $cStaticSlotIDMax + gSyncNodeMax = 1023 + gSyncNodeMax$ , the sync frame overflow flag EIR.SFO is set erroneously.

**Scope:**

The erratum is limited to configurations where the number of transmitted and received sync frames equals to gSyncNodeMax and the number of static slots plus the number of dynamic slots is greater or equal than  $1023 + gSyncNodeMax$ .

**Effects:**

In the described case the sync frame overflow flag `EIR.SFO` is set erroneously. This has no effect to the POC state.

**Workaround**

Configure `gSyncNodeMax` to number of transmitted and received sync frames plus one or avoid configurations where the total of static and dynamic slots is greater than `cStaticSlotIDMax`.

**FlexRay AI.095 Register RCV displays wrong value****Description:**

If the calculated rate correction value is in the range of `[-pClusterDriftDamping .. +pClusterDriftDamping]`, `vRateCorrection` of the CSP process is set to zero. In this case register `RCV` should be updated with this value. Erroneously `RCV.RCV[11:0]` holds the calculated value in the range `[-pClusterDriftDamping .. +pClusterDriftDamping]` instead of zero.

**Scope:**

The erratum is limited to the case where the calculated rate correction value is in the range of `[-pClusterDriftDamping .. +pClusterDriftDamping]`.

**Effects:**

The displayed rate correction value `RCV.RCV[11:0]` is in the range of `[-pClusterDriftDamping .. +pClusterDriftDamping]` instead of zero. The error of the displayed value is limited to the range of `[-pClusterDriftDamping .. +pClusterDriftDamping]`. For rate correction in the next double cycle always the correct value of zero is used.

**Workaround**

A value of `RCV.RCV[11:0]` in the range of `[-pClusterDriftDamping .. +pClusterDriftDamping]` has to be interpreted as zero.

**FlexRay AI.096 Noise following a dynamic frame that delays idle detection may fail to stop slot****Description:**

If (in case of noise) the time between 'potential idle start on X' and 'CHIRP on X' (see Protocol Spec. v2.1, Figure 5-21) is greater than `gdDynamicSlotIdlePhase`, the E-Ray will not remain for the remainder of the current dynamic segment in the state 'wait for the end of dynamic slot rx'. Instead, the E-Ray continues slot counting. This may enable the node to further transmissions in the current dynamic segment.

**Scope:**

The erratum is limited to noise that is seen only locally and that is detected in the time window between the end of a dynamic frame's DTS and idle detection ('CHIRP on X').

**Effects:**

In the described case the faulty node may not stop slot counting and may continue to transmit dynamic frames. This may lead to a frame collision in the current dynamic segment.

**Workaround**

None.

**FlexRay AI.097 Loop back mode operates only at 10 MBit/s****Description:**

The looped back data is falsified at the two lower baud rates of 5 and 2.5 MBit/s.

**Scope:**

The erratum is limited to test cases where loop back is used with the baud rate prescaler (`PRTC1.BRP[1:0]`) configured to 5 or 2.5 MBit/s.

Effects:

The loop back self test is only possible at the highest baud rate.

### **Workaround**

Run loop back tests with 10 MBit/s (`PRTC1.BRP[1:0] = 00B`).

### **FlexRay AI.098 Suspend Mode is not functional**

Description:

The applied kernel mode (`KSCFG.SUMCFG=2`, Stop Mode 0) in suspend state will not be performed. The module proceeds its communication and the module clock is not switched off.

### **Workaround**

To keep a smooth FlexRay communication ongoing, the FlexRay controller will go to halt state at the end of the communication cycle. If the FlexRay controller is in “NORMAL\_ACTIVE” or “NORMAL\_PASSIVE” state, the controller goes to POC HALT state, by setting `SUCC1.CMD=0110B`. In any other state, the command will not be accepted (`SUCC1.CMD=0000B` “COMMAND\_NOT\_ACCEPTED”).

### **FlexRay AI.099 Erroneous cycle offset during startup after abort of start-up or normal operation**

Description:

An abort of startup or normal operation by a READY command near the macotick border may lead to the effect that the state INITIALIZE\_SCHEDULE is one macrotick too short during the first following integration attempt. This leads to an early cycle start in state INTEGRATION\_COLDSTART\_CHECK or INTEGRATION\_CONSISTENCY\_CHECK.

As a result the integrating node calculates a cycle offset of one macrotick at the end of the first even/odd cycle pair in the states

**Detailed Errata Description**

INTEGRATION\_COLDSTART\_CHECK or  
INTEGRATION\_CONSISTENCY\_CHECK and tries to correct this offset.

If the node is able to correct the offset of one macrotick ( $\text{pOffsetCorrectionOut} >> \text{gdMacrotick}$ ), the node enters NORMAL\_ACTIVE with the first startup attempt.

If the node is not able to correct the offset error because  $\text{pOffsetCorrectionOut}$  is too small ( $\text{pOffsetCorrectionOut} \leq \text{gdMacrotick}$ ), the node enters ABORT\_STARTUP and is ready to try startup again. The next (second) startup attempt is not effected by this erratum.

**Scope:**

The erratum is limited to applications where READY command is used to leave STARTUP, NORMAL\_ACTIVE, or NORMAL\_PASSIVE state.

**Effects:**

In the described case the integrating node tries to correct an erroneous cycle offset of one macrotick during startup.

**Workaround**

With a configuration of  $\text{pOffsetCorrectionOut} >> \text{gdMacrotick} \cdot (1 + \text{cClockDeviationMax})$  the node will be able to correct the offset and therefore also be able to successfully integrate.

**FlexRay AI.100 First WUS following received valid WUP may be ignored****Description:**

When the protocol engine is in state WAKEUP\_LISTEN and receives a valid wakeup pattern (WUP), it transfers into state READY and updates the wakeup status vector  $\text{CCSV.WSV}[2:0]$  as well as the status interrupt flags  $\text{SIR.WST}$  and  $\text{SIR.WUPA/B}$ . If the received wakeup pattern continues, the protocol engine may ignore the first wakeup symbol (WUS) following the state transition and signals the next  $\text{SIR.WUPA/B}$  at the third instead of the second WUS.

**Scope:**

The erratum is limited to the reception of redundant wakeup patterns.

**Effects:**

Delayed setting of status interrupt flags `SIR.WUPA/B` for redundant wakeup patterns.

**Workaround**

None.

**FlexRay AI.101 READY command accepted in READY state**

**Description:**

The E-Ray module does not ignore a READY command while in READY state.

**Scope:**

The erratum is limited to the READY state.

**Effects:**

Flag `CCSV.CSI` is set. Cold starting needs to be enabled by POC command `ALLOW_COLDSTART (SUCC1.CMD = 1001B)`.

**Workaround**

None.

**FlexRay AI.102 Slot Status vPOC!SlotMode is reset immediately when entering HALT state**

**Description:**

When the protocol engine is in the states `NORMAL_ACTIVE` or `NORMAL_PASSIVE`, a HALT or FREEZE command issued by the Host resets

**Detailed Errata Description**

vPOC!SlotMode immediately to SINGLE slot mode ( $CCSV.SLM[1:0] = 00_B$ ). According to the FlexRay protocol specification, the slot mode should not be reset to SINGLE slot mode before the following state transition from HALT to DEFAULT\_CONFIG state.

Scope:

The erratum is limited to the HALT state.

Effects:

The slot status vPOC!SlotMode is reset to SINGLE when entering HALT state.

**Workaround**

None.

**FlexRay AI.103 Received messages not stored in Message RAM when in Loop Back Mode**

After a FREEZE or HALT command has been asserted in NORMAL\_ACTIVE state, and if state LOOP\_BACK is then entered by transition from HALT state via DEF\_CONFIG and CONFIG, it may happen that acceptance filtering for received messages is not started, and therefore these messages are not stored in the respective receive buffer in the Message RAM.

Scope:

The erratum is limited to the case where Loop Back Mode is entered after NORMAL\_ACTIVE state was left by FREEZE or HALT command.

Effects:

Received messages are not stored in Message RAM because acceptance filtering is not started.

**Workaround**

Leave HALT state by hardware reset.

**FlexRay\_X.001 Trigger User Reset after Power-On**

After a Power-On the FlexRay Interrupt Request Flags (e.g. `FR_0IC.IR`) are sometime set erroneous. After interrupt enabling (e.g. `FR_0IC.IE`) and if the corresponding Interrupt Request Flag is set the interrupt routine will be called.

**Workaround**

Trigger a Functional / User Reset (i.e. Debug Reset, Internal Application Reset or Application Reset) by software and after that the FlexRay module is correct installed.

**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 = 01B`) and processes `T6` before `T5`. The GPT1 state machine has 8 states (4 states when `BPS1 = 01B`) 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`:



## Detailed Errata Description

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

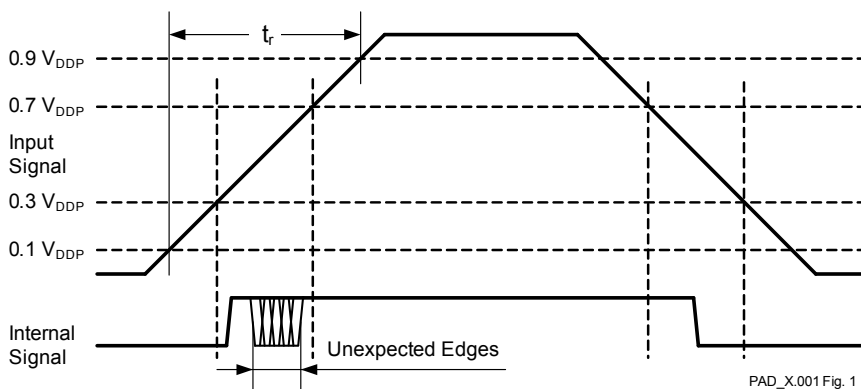
**Workaround**

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 10 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 10](#).

### **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**

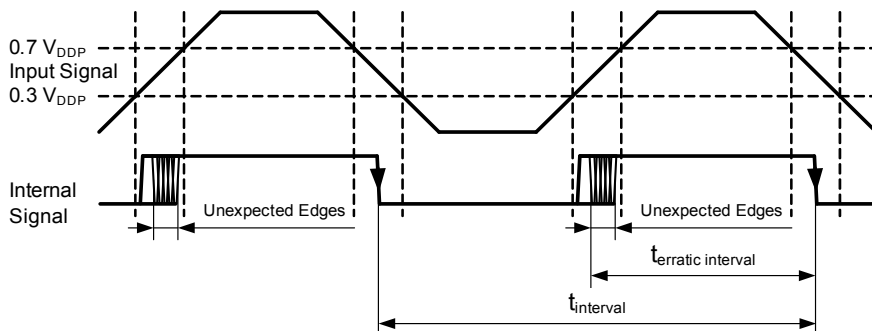
1. The captured time interval value should be checked whether it is in a reasonable range.
2. The input pin should be read several times, and a majority decision may be made to decide whether the edge was correct or erratic.

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  $t_{\text{interval}}$  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/  
 $t_{\text{erratic interval}}$  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.



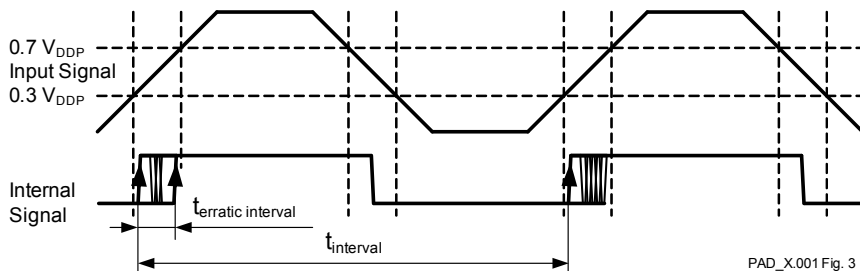
PAD\_X.001 Fig. 2

**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**

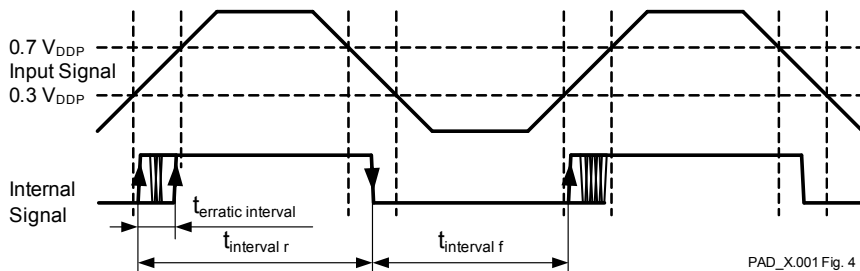
If the time between the rising signal edge and the rising edge caused by a spike  $t_{erratic}$  is less than the ISR service time, a workaround would be to clear the interrupt request (IR) flag before the return from interrupt is done. The clearing of the IR flag will avoid a further erratic interrupt.

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:

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



**Figure 5 Rising Edge and Falling Edge Detection Approach**

**PARITY X.001 PMTSR Register Initialization**

The PMTSR register content after start-up is 0100<sub>H</sub>, meaning the parity logic for SBRAM is not in standard mode of operation.

**Workaround**

If parity will be used as Memory Control mechanism for SBRAM, it must be enabled by initializing the PMTSR register with 8000<sub>H</sub>.

**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

**Detailed Errata Description**

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<sub>B</sub>) 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**

This erratum only affects devices with a Date Code before G1203 (i.e. digit value < 1203).



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<sub>H</sub>..C001F3<sub>H</sub>). This option is selected by configuration pin values (HWCFG) xxxxx111<sub>B</sub> (code start from internal Flash) or x1100000<sub>B</sub> (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 if the block from C001F0<sub>H</sub> to C001FF<sub>H</sub> is programmed by the user. A limited number of devices are affected – a 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 if no programming of the block from C001F0<sub>H</sub> to C001FF<sub>H</sub> is done after the last power-on.

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**

1. Do not program the page from C00180<sub>H</sub> to C001FF<sub>H</sub>. This provides an erased, error free flash read during start-up (without installed flash redundancy) of DBGPRR register value which allows start-up from internal/external memory and JTAG position A.
2. 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.
3. 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  $f_{\text{sys}}/2$  instead of  $f_{\text{sys}}$ .

**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  $t_{\text{HDEL}} = 4 \times 1/f_{\text{SYS}} + \text{delay}$  (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)

**Detailed Errata Description**

- 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.

**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<sub>B</sub> 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.

**Detailed Errata Description**

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<sub>B</sub> 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: ` $t_{RET} \geq 5$  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:

- $t_{RET} \geq 5$  years; Valid for Flash module 6 (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  $SWDCON.LEV_{xV} = 1001_B$ , or  $< 3.0$  V for  $SWDCON.LEV_{xV} = 0001_B$ ).

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  $V_{\text{SWD}}$  tolerance boundaries for 5.5 V are changed from  $\pm 0.15 \text{ V}$  to  $\pm 0.30 \text{ V}$ .

## 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 ( $f_{ADC1}$ ) 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<sub>B</sub> (event for channel CC<sub>y</sub> has not yet occurred)
- Meanwhile, event for CC<sub>y</sub> occurs, and bit `SEEy` is cleared to 0<sub>B</sub> by hardware
- In the Write-Back stage, software writes `CCn_SEE` with bit `SEEx` = 1<sub>B</sub> (intended event for CC<sub>x</sub> enabled via OR instruction) **and** bit `SEEy` = 1<sub>B</sub>
- or, as inverse procedure, software writes `CCn_SEE` with bit `SEEx` = 0<sub>B</sub> (intended event for CC<sub>x</sub> disabled via AND instruction) **and** bit `SEEy` = 1<sub>B</sub>

In these cases, another unintended event for channel CC<sub>y</sub> 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<sub>B</sub>), 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.

**FlexRay\_AI.H004 Only the first message can be received in External Loop Back mode**

If the loop back (TXD to RXD) will be performed via external physical transceiver, there will be a large delay between TXD and RXD.

A delay of two sample clock periods can be tolerated from TXD to RXD due to a majority voting filter operation on the sampled RXD.

Only the first message can be received, due to this delay.

To avoid that only the first message can be received, a start condition of another message (idle and sampling '0' -> low pulse) must be performed.

The following procedure can be applied at one or both channels:

- wait for no activity (`TEST1.AOx=0` -> bus idle)
- set Test Multiplexer Control to I/O Test Mode (`TEST1.TMC=2`), simultaneously `TXDx=TXENx=0`
- wait for activity (`TEST1.AOx=1` -> bus not idle)
- set Test Multiplexer Control back to Normal signal path (`TEST1.TMC=0`)
- wait for no activity (`TEST1.AOx=0` -> bus idle)

Now the next transmission can be requested.

### **FlexRay AI.H005 Initialization of internal RAMs requires one eray\_bclk cycle more**

The initialization of the E-Ray internal RAMs as started after hardware reset or by CHI command `CLEAR_RAM` (`SUCC1.CMD[3:0] = 1100B`) takes 2049 eray\_bclk cycles instead of 2048 eray\_bclk cycles as described in the E-Ray Specification.

Signalling of the end of the RAM initialization sequence by transition of `MHDS.CRAM` from `1B` to `0B` is correct.

### **FlexRay AI.H006 Transmission in ATM/Loopback mode**

When operating the E-Ray in ATM/Loopback mode there should be only one transmission active at the same time. Requesting two or more transmissions in parallel is not allowed.

To avoid problems, a new transmission request should only be issued when the previously requested transmission has finished. This can be done by checking registers `TXRQ1/2/3/4` for pending transmission requests.

### **FlexRay AI.H007 Reporting of coding errors via `TEST1.CERA/B`**

When the protocol engine receives a frame that contains a frame CRC error as well as an FES decoding error, it will report the FES decoding error instead of the CRC error, which should have precedence according to the non-clocked SDL description.

This behaviour does not violate the FlexRay protocol conformance. It has to be considered only when `TEST1.CERA/B` is evaluated by a bus analysis tool.

### **FlexRay AI.H009 Return from test mode operation**

The E-Ray FlexRay IP-module offers several test mode options

- Asynchronous Transmit Mode
- Loop Back Mode
- RAM Test Mode
- I/O Test Mode

To return from test mode operation to regular FlexRay operation we strongly recommend to apply a hardware reset via input `eray_reset` to reset all E-Ray internal state machines to their initial state.

*Note: The E-Ray test modes are mainly intended to support device testing or FlexRay bus analyzing. Switching between test modes and regular operation is not recommended.*

### **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 (`00B`), it must be initialized first before any mode involving external trigger signals is configured for the associated GPT<sub>x</sub> 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 GPT<sub>x</sub> 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 11](#) 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 11](#):

**Table 11      Equivalent Number of System Clock Cycles Required to Wait for Two Basic Clock Cycles**

<b>Setting of BPS1</b>	<b>BPS1 = 01</b>	<b>BPS1 = 00</b>	<b>BPS1 = 11</b>	<b>BPS1 = 10</b>
Required Number of System Clocks	8	16	32	64
<b>Setting of BPS2</b>	<b>BPS2 = 01</b>	<b>BPS2 = 00</b>	<b>BPS2 = 11</b>	<b>BPS2 = 10</b>
Required Number of System Clocks	4	8	16	32

**Detailed Errata Description**

In case the required timer resolution can be achieved with different combinations of the Block Prescaler  $BPS1/BPS2$  and the Individual Prescalers  $T_{xI}$ , 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

**Detailed Errata Description**

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**

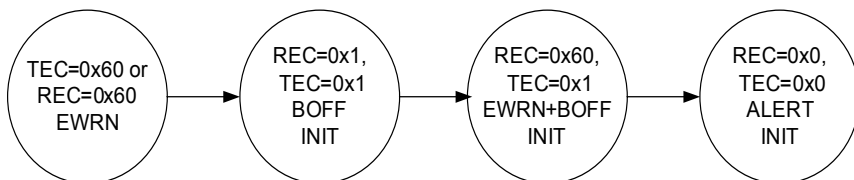
The time stamp measurement feature is not based on an absolute time measurement, but on actual CAN bit times which are subject to the CAN resynchronization during CAN bus operation. The time stamp value merely indicates the number of elapsed actual bit times. Those actual bit times can be shorter or 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<sub>B</sub>, 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<sub>B</sub>, the node waits for the bus idle state and then sets bit NSR.SUSACK = 1<sub>B</sub>.

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

**Detailed Errata Description**

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.

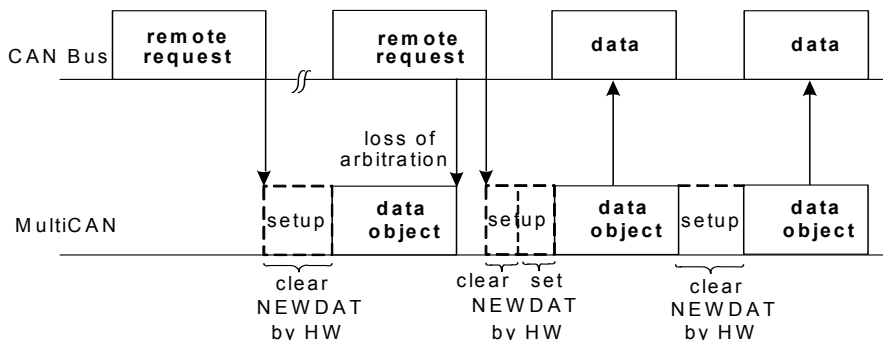
**MultiCAN\_TC.H004 Double remote request**

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 (PVC, 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.

**Detailed Errata Description**

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.PVCM12` and `SCU_INTDIS.PVC1I2`.
3. Clear the PVC interrupt request flag source `SCU_DMPMITCLR.PVCM12` and `SCU_DMPMITCLR.PVC1I2`.
4. Clear the PVC interrupt request flag by writing to `SCU_INTCLR.PVCM12` and `SCU_INTCLR.PVC1I2`.
5. Clear the selected SCU request flag (default is `SCU_1IC.IR`).

The Power Validation Circuits (PVC) 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.PVCM12`.
3. Clear the PVC interrupt request flag source `SCU_DMPMITCLR.PVCM12`.
4. Clear the PVC interrupt request flag by writing to `SCU_INTCLR.PVCM12`.
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  $f_{\text{RTC}}$  for the Real Time Clock module (RTC) can be selected via bit field  $\text{RTCCLKSEL}$  in register  $\text{RTCCLKCON}$ . Whenever the system clock is less than 4 times faster than the RTC count input clock ( $f_{\text{SYS}} < f_{\text{RTC}} \times 4$ ), Asynchronous Mode must be selected (bit  $\text{RTCCM} = 1_{\text{B}}$  in register  $\text{RTCCLKCON}$ ).

To assure data consistency in the count registers  $\text{T14}$ ,  $\text{RTCL}$ ,  $\text{RTCH}$ , the RTC module must be temporarily switched off by setting bit  $\text{MODEN} = 0_{\text{B}}$  in register  $\text{RTC\_KSCCFG}$  before register  $\text{RTCCLKCON}$  is modified, i.e. whenever

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

### **SCU\_X.H009 $\text{WUCR.TTSTAT}$ can be set after a Power-Up**

After power-up the wake-up clock  $f_{\text{WU}}$  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 ( $\text{WUCR.TTSTAT}$ ) 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  $\text{SCU\_INTDIS.WUTI}$
2. Clear the trim interrupt request flag by writing to  $\text{INTCLR.WUTI}$
3. Clear the selected SCU request flag (default is  $\text{SCU\_1IC.IR}$ )

**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 = 0001B, 0010B, 0011B or 0100B`) might lead to unintended spikes on these port pins. To avoid

**Detailed Errata Description**

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 resetting 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<sub>B</sub> 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 external bus control signals are held inactive during the internal LXBus access.*

### **ECC\_X.D002 Initialization of the Read-Control Logic**

In chapter ‘ECC Error Handling’ (8.14.3) of the User's Manual version 1.1 the following note should be added.

*Note: The state of the read-control logic must get cleared after initialization of a RAM with ECC enabled. This is achieved with a read operation from one*

**Detailed Errata Description**

*location in each of the RAMs which was initialized before. This read operation must get executed before enabling the corresponding ECC trap.*

**RESET\_X.D001 Reset Types of Trap Registers**

The reset type of SCU registers TRAPDIS, TRAPSET, TRAPNP and TRAPNP1 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.

[www.infineon.com](http://www.infineon.com)