# 16-Bit

Architecture

# XE166U Derivatives

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

## Errata Sheet

V1.2 2-13-09

Microcontrollers

**Edition 2-13-09**

**Published by**
**Infineon Technologies AG**
**81726 Munich, Germany**



### Legal Disclaimer

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

### Information

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

### Warnings

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

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

# 16-Bit

Architecture

# XE166U Derivatives

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

## Errata Sheet

V1.2 2-13-09

Microcontrollers

# Table of Contents

# 1 History List / Change Summary

**Table 1 History List**

| Version | Date | Remark[1)] |
|---|---|---|
| 1.0 | 26.01.2011 | First Errata Sheet release. |
| 1.1 | 10.08.2011 | Errata No. 02059AERRA, new Marking/Step (AA) added to Errata Sheet. |
| 1.2 | 30.09.2013 | Errata No. 02802AERRA. |

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 XE166U 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 XE166U 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

# 3 Current Documentation

The Infineon XE166 Family comprises device types from the XE160x Series and the XE161x Series.

| | |
|---|---|
| Device | XE16xxU |
| Marking/Step | EES-AA, ES-AA, AA |
| Package | PG-TSSOP-38, PG-VQFN-48 |

This Errata Sheet refers to the following documentation:

- XE166U Derivatives User's Manual
- XE160FU Data Sheet
- XE161FU Data Sheet
- Documentation Addendum (if applicable)

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

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

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

# 4 Errata Device Overview

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

## 4.1 Functional Deviations

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

**Table 2 Errata Device Overview: Functional Deviations**

| Functional Deviation | XE16xxU | | |
|---|---|---|---|
| | EES-AA ES-AA | AA[1)] | |
| ADC_AI.002 | X | X | |
| ADC_X.001 | X | | |
| ADC_X.002 | X | X | |
| ESR_X.002 | X | X | |
| ESR_X.004 | X | X | |
| GPT12E_X.002 | X | X | |
| OCDS_X.003 | X | X | |
| RESET_X.004 | X | X | |
| SCU_X.012 | X | X | |
| StartUp_X.004 | X | X | |
| USIC_AI.004 | X | X | |
| USIC_AI.005 | X | X | |
| USIC_AI.016 | X | X | |
| USIC_AI.018 | X | X | |

1) From EES/ES-AA step to AA step, 1 erratum has 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 | XE16xxU | | |
|---|---|---|---|
| | EES-AA ES-AA | AA | |
| FLASH_X.P001 | X | X | |
| StartUp_X.P001 | X | X | |
| SWD_X.P002 | X | X | |

## 4.3 Application Hints

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

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

| Hint | XE16xxU | | |
|---|---|---|---|
| | EES-AA ES-AA | AA | |
| ADC_AI.H002 | X | X | |
| ADC_AI.H003 | X | X | |
| CAPCOM12_X.H001 | X | X | |
| CC6_X.H001 | X | X | |
| GPT12_AI.H001 | X | X | |
| GPT12E_X.H002 | X | X | |
| INT_X.H002 | X | X | |
| INT_X.H004 | X | X | |
| OCDS_X.H003 | X | X | |
| PVC_X.H001 | 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 | |

# 5 Short Errata Description

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

## 5.1 Functional Deviations

Table 5 shows a short description of the functional deviations.

**Table 5 Functional Deviations**

| Functional Deviation | Short Description | Chg | Pg |
|---|---|---|---|
| ADC_AI.002 | Result of Injected Conversion may be wrong | New | 17 |
| ADC_X.001 | Cross-Current between VAREF and VAGND | | 17 |
| ADC_X.002 | Current Drawn on VAREF Pin can be Unexpected High | | 18 |
| ESR_X.002 | ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access | | 18 |
| ESR_X.004 | Wrong Value of SCU_RSTCONx Registers after ESRy Application Reset | New | 19 |
| GPT12E_X.002 | Effects of GPT Module Microarchitecture | | 20 |
| OCDS_X.003 | Peripheral Debug Mode Settings cleared by Reset | | 21 |
| RESET_X.004 | Sticky “Register Access Trap” forces device into power-save mode after reset. | New | 22 |
| SCU_X.012 | Wake-Up Timer RUNCON Command | | 23 |
| StartUp_X.004 | PSRAM Initialization | | 23 |
| USIC_AI.004 | Receive shifter baudrate limitation | | 24 |
| USIC_AI.005 | Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time | | 24 |
| USIC_AI.016 | Transmit parameters are updated during FIFO buffer bypass | New | 25 |

**Table 5 Functional Deviations** (cont'd)

| Functional Deviation | Short Description | Chg | Pg |
|---|---|---|---|
| USIC_AI.018 | Clearing PSR.MSLS bit immediately deasserts the SELOx output signal | New | 26 |

## 5.2 Deviations from Electrical and Timing Specification

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

**Table 6 Deviations from Electrical and Timing Specification**

| AC/DC/ADC Deviation | Short Description | Chg | Pg |
|---|---|---|---|
| FLASH_X.P001 | Test Condition for Flash parameter NER in Data Sheets | New | 27 |
| StartUp_X.P001 | Supply Voltage Restrictions wrong or missing | New | 27 |
| SWD_X.P002 | Supply Watchdog (SWD) Supervision Level in Data Sheet. | New | 27 |

## 5.3 Application Hints

Table 7 shows a short description of the application hints.

**Table 7 Application Hints**

| Hint | Short Description | Chg | Pg |
|---|---|---|---|
| ADC_AI.H002 | Minimizing Power Consumption of an ADC Module | | 28 |
| ADC_AI.H003 | Injected conversion may be performed with sample time of aborted conversion | | 28 |
| CAPCOM12_X.H001 | Enabling or Disabling Single Event Operation | | 29 |
| CC6_X.H001 | Modifications of Bit MODEN in Register CCU6x_KSCFG | | 31 |
| GPT12_AI.H001 | Modification of Block Prescalers BPS1 and BPS2 | New | 31 |
| GPT12E_X.H002 | Reading of Concatenated Timers | | 32 |
| INT_X.H002 | Increased Latency for Hardware Traps | | 33 |
| INT_X.H004 | SCU Interrupts Enabled After Reset | | 33 |
| OCDS_X.H003 | Debug Interface Configuration by User Software | | 34 |
| PVC_X.H001 | PVC Threshold Level 2 | | 34 |
| RTC_X.H003 | Changing the RTC Configuration | | 35 |
| SCU_X.H009 | WUCR.TTSTAT can be set after a Power-Up | | 35 |
| SWD_X.H001 | Application Influence on the SWD | | 36 |
| USIC_AI.H001 | FIFO RAM Parity Error Handling | | 36 |
| USIC_AI.H002 | Configuration of USIC Port Pins | New | 36 |
| USIC_AI.H003 | PSR.RXIDLE Cleared by Software | New | 37 |

# 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 (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 (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 ($\overline{PORST}$ = $V_{SS}$).

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 8 Typical Current Values**

| $V_{AGND}$ / V | $V_{AREF}$ / V | $I_{AREF\ AGND}$ / mA |
|---|---|---|
| 0 | 5.5 | 8.5 |
| 0 | 5.0 | 7.4 |
| 0 | 4.5 | 6.4 |

**Table 8 Typical Current Values** (cont'd)

| $V_{AGND}$ / V | $V_{AREF}$ / V | $I_{AREF\ AGND}$ / mA |
|---|---|---|
| 0 | 3.3 | 3.9 |
| 0 | 3.0 | 3.3 |

**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.

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

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

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

**Workaround**

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

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

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

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

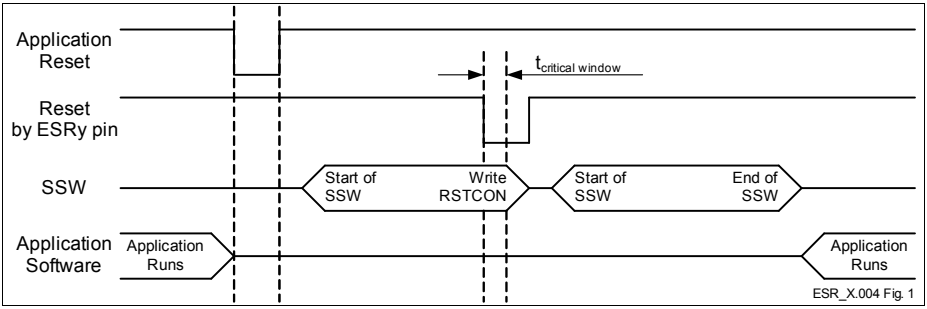


**Figure 1 Critical application reset sequence**

### Workaround

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

## 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.

**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.004 PSRAM Initialization

As the User’s Manual states, any RAM (PSRAM, DSRAM and DPRAM) that uses parity as Memory Content Protection mechanism needs to be initialized before the parity is activated.

Because the built-in initialization does not work properly for PSRAM, the user software must perform following steps at its very beginning if parity in PSRAM is needed:

1. Check if the last start-up event has been a power-on - after such event the RAMs contain random data and must be initialized,
   - if SCU_STMEM0.[4] <> $1_B$ - no power-on, no initialization needed (it has already been performed) - exit this sequence;
   - if SCU_STMEM0.[4] = $1_B$ - initialization needed, continue with step 2.
2. Optional step,
   if the application and the system allow a clock-frequency above 10 MHz

(system frequency after power-on) - clock reconfiguration can be done here to use the increased speed for a faster RAM initialization;

3. Activate parity in PSRAM by installing the bits as follows:
   - disable parity traps by setting SCU_TRAPDIS.PET = $1_B$
   - enable trap requests by setting SCU_PEEN.PEENPS = $1_B$
   - enable parity error sensitivity by setting SCU_PMTSR.PESEN = $1_B$
4. Perform a write access to each PSRAM location
   The exact content written doesn't matter for parity; the user can decide either to fill the memories with all zeroes or something else.
5. Read one (arbitrary) PSRAM location to assure correct initial state of the read-control logic
6. Assure error-flag is reset for PSRAM - clear SCU_PECON.PEFPS by writing one to it

After this sequence, PSRAM is ready to be used and parity is active.

It is a further decision of the user either to enable parity trap (by resetting SCU_TRAPDIS.PET) for error-handling.

### 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 x $1/f_{SYS}$ + 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 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 bit 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.

## 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 1 (up to 32 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 $N_{PP}$.*

**StartUp_X.P001 Supply Voltage Restrictions wrong or missing**

The following restriction:

“During power-on sequences, the supply voltages may only change with a maximum speed of dV/dt < 5 V/µs, i.e. the target supply voltage may be reached earliest after approx. 1 µs.”

Is wrongly given in Section “4.2 DC Parameters” of the Data Sheet.

Is missing in Section “4.3 DC Parameters” of the Data Sheet.

Please adhere to the above requirement in your Application.

**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.

### GPT12_AI.H001 Modification of Block Prescalers BPS1 and BPS2

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

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

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

## GPT12E_X.H002 Reading of Concatenated Timers

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

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

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

- read Timer_high_temp = T2
- read Timer_low = T3
- wait two basic clock cycles (to allow increment/decrement of auxiliary timer in case of core timer overflow/underflow) - see **Table 9** 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 9**:

**Table 9 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.

### 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) compare the supply voltage of the respective domain (DMP_M) with programmable levels (LEV1V and LEV2V in register SCU_PVCMCON0).

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

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

LEV2V can be disabled by executing the following sequence:

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

### 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.

w w w . i n f i n e o n . c o m

02802AERRA

Published by Infineon Technologies AG