16/32-Bit

Architecture

XC27x2X Derivatives

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

Errata Sheet

V1.0 2011-01

Microcontrollers

Edition 2011-01

Published by
Infineon Technologies AG
81726 Munich, Germany

Legal Disclaimer

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

Information

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

Warnings

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

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

16/32-Bit

Architecture

XC27x2X Derivatives

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

Errata Sheet

V1.0 2011-01

Microcontrollers

Table of Contents

1 History List / Change Summary . . . 5

2 General . . . 6

3 Current Documentation . . . 7

4 Short Errata Description . . . 8
4.1 Functional Deviations . . . 8
4.2 Deviations from Electrical and Timing Specification . . . 9
4.3 Application Hints . . . 10

5 Detailed Errata Description . . . 11
5.1 **Functional Deviations** . . . 11
ESR_X.002 . . . 11
GPT12E_X.002 . . . 11
OCDS_X.003 . . . 13
StartUp_X.004 . . . 14
USIC_AI.004 . . . 15
USIC_AI.005 . . . 15
5.2 **Deviations from Electrical and Timing Specification** . . . 16
5.3 **Application Hints** . . . 17
ADC_AI.H002 . . . 17
CAPCOM12_X.H001 . . . 17
CC6_X.H001 . . . 19
GPT12E_X.H002 . . . 19
INT_X.H002 . . . 20
INT_X.H004 . . . 21
OCDS_X.H003 . . . 21
PVC_X.H001 . . . 22
RTC_X.H003 . . . 22
SCU_X.H009 . . . 23
SWD_X.H001 . . . 23
USIC_AI.H001 . . . 23

1 History List / Change Summary

Table 1 History List

Version	Date	Remark[1)]
1.0	26.01.2011	First Errata Sheet release.

1) Errata changes to the previous Errata Sheet are marked in **Chapter 4** "**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 XC27x2X Derivatives from the current user documentation.

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

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

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

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

3 Current Documentation

The Infineon XC2000 Family comprises device types from the XC2200 group, the XC2300 group and the XC2700 group. The XC27x2X device types belong to the XC2700 group.

Device	XC27x2X
Marking/Step	EES-AA, ES-AA
Package	PG-TSSOP-38, PG-VQFN-48

This Errata Sheet refers to the following documentation:

- XC27x2X Derivatives User's Manual
- XC2712X Data Sheet
- XC2722X Data Sheet
- Documentation Addendum (if applicable)

Make sure you always use the corresponding documentation for this device available in category 'Documents' at **www.infineon.com/xc2700**.

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

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

4 Short Errata Description

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

4.1 Functional Deviations

Table 2 shows a short description of the functional deviations.

Table 2 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
ESR_X.002	ESREXSTAT1 and ESREXSTAT2 Status Bits can be Cleared after a Write Access	New	11
GPT12E_X.002	Effects of GPT Module Microarchitecture	New	11
OCDS_X.003	Peripheral Debug Mode Settings cleared by Reset	New	13
StartUp_X.004	PSRAM Initialization	New	14
USIC_AI.004	Receive shifter baudrate limitation	New	15
USIC_AI.005	Only 7 data bits are generated in IIC mode when TBUF is loaded in SDA hold time	New	15

4.2 Deviations from Electrical and Timing Specification

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

Table 3 Deviations from Electrical and Timing Specification

AC/DC/ADC Deviation	Short Description	Chg	Pg
- none -			

4.3 Application Hints

Table 4 shows a short description of the application hints.

Table 4 Application Hints

Hint	Short Description	Chg	Pg
ADC_AI.H002	Minimizing Power Consumption of an ADC Module	New	17
CAPCOM12_X.H001	Enabling or Disabling Single Event Operation	New	17
CC6_X.H001	Modifications of Bit MODEN in Register CCU6x_KSCFG	New	19
GPT12E_X.H002	Reading of Concatenated Timers	New	19
INT_X.H002	Increased Latency for Hardware Traps	New	20
INT_X.H004	SCU Interrupts Enabled After Reset	New	21
OCDS_X.H003	Debug Interface Configuration by User Software	New	21
PVC_X.H001	PVC Threshold Level 2	New	22
RTC_X.H003	Changing the RTC Configuration	New	22
SCU_X.H009	WUCR.TTSTAT can be set after a Power-Up	New	23
SWD_X.H001	Application Influence on the SWD	New	23
USIC_AI.H001	FIFO RAM Parity Error Handling	New	23

5 Detailed Errata Description

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

5.1 Functional Deviations

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

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

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

Workaround

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

GPT12E_X.002 Effects of GPT Module Microarchitecture

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

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

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

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

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

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

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

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

Workaround:

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

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

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

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

Workaround 1:

To compensate the delay and achieve correct timing,

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

Workaround 2:

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

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.

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 $t_{HDEL} = 4 \times 1/f_{SYS}$ + delay (approximately 60ns @ 80MHz).

Workaround

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

5.2 Deviations from Electrical and Timing Specification

- none -

5.3 Application Hints

ADC_AI.H002 Minimizing Power Consumption of an ADC Module

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

Recommendation for Minimum Power Consumption:

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

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

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

CAPCOM12_X.H001 Enabling or Disabling Single Event Operation

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

One Channel in Single Event Operation

As the Single Event Enable registers CC1_SEE, CC2_SEE are not located in the bit-addressable SFR address range, they can only be modified by instructions

operating on data type WORD. This is no problem when only one channel of a CAPCOM unit is used in single event mode.

Two or more Channels in Single Event Operation

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

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

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

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

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

CC6_X.H001 Modifications of Bit MODEN in Register CCU6x_KSCFG

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

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

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 5** 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 5**:

Table 5 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.

www.infineon.com

Published by Infineon Technologies AG