

XE166M Derivatives

16-Bit Single-Chip

Real Time Signal Controller

16bit

Microcontrollers



Never stop thinking

Edition 30.01.2009

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

**© 2009 Infineon Technologies AG
All Rights Reserved.**

Legal Disclaimer

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

Information

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

Warnings

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

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

XE166M Derivatives

16-Bit Single-Chip

Real Time Signal Controller

Microcontrollers



Never stop thinking

Table of Contents

1	History List / Change Summary	6
2	General	7
3	Current Documentation	8
4	Errata Device Overview	9
4.1	Functional Deviations	9
4.2	Deviations from Electrical and Timing Specification	10
4.3	Application Hints	11
4.4	Documentation Updates	12
5	Short Errata Description	13
5.1	Errata removed in this Errata Sheet	13
5.2	Functional Deviations	14
5.3	Deviations from Electrical and Timing Specification	15
5.4	Application Hints	16
5.5	Documentation Updates	17
6	Detailed Errata Description	18
6.1	Functional Deviations	18
	ECC_X.002	18
	ESR_X.001	19
	GPT12E_X.002	19
	HARR_X.001	20
	OCDS_X.003	21
	PARITY_X.001	22
	RESET_X.003	22
	StartUp_X.001	22
	USIC_AI.004	23
	WDT_X.002	24
6.2	Deviations from Electrical and Timing Specification	26
6.3	Application Hints	27
	CAPCOM12_X.H001	27
	CC6_X.H001	28
	ECC_X.H001	28
	GPT12E_X.H002	29
	INT_X.H002	30
	LXBUS_X.H001	31
	MultiCAN_AI.H005	31
	MultiCAN_AI.H006	31
	MultiCAN_TC.H002	32
	MultiCAN_TC.H003	32

	MultiCAN_TC.H004	32
	RESET_X.H003	33
	RTC_X.H003	33
	USIC_AI.H001	34
6.4	Documentation Updates	35
	ECC_X.D001	35
	StartUp_X.D001	35

1 History List / Change Summary

Table 1 History List

Version	Date	Remark ¹⁾
1.0	26.06.2008	new Errata Sheet
1.1	05.09.2008	
1.2	30.01.2009	Errata Sheet name is changed from “XE166xM Derivatives” to “XE166M Derivatives”; Documentation Updates are added into the Errata Sheet

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

2 General

This Errata Sheet describes the deviations of the XE166M 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 a workaround which is possibly supported by the compiler tool vendor. Some corresponding compiler switches need possibly to be set. Please see the respective documentation of your compiler.

Some errata of this Errata Sheet do not refer to all of the XE166M 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 XE166 Family comprises device types from the XE162x Series, the XE164x Series and the XE167x Series.

Device	XE16xxM
Marking/Step	EES-AA, ES-AA, AA
Package	PG-LQFP-64, PG-LQFP-100, PG-LQFP-144

This Errata Sheet refers to the following documentation:

- XE166M Derivatives User's Manual
- XE162xM Data Sheet
- XE164xM Data Sheet
- XE167xM 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	XE16xxM	
	EES-AA ES-AA	AA ¹⁾
ECC_X.002	X	X
ESR_X.001	X	X
GPT12E_X.002	X	X
HARR_X.001	X	
OCDS_X.003	X	X
PARITY_X.001	X	X
RESET_X.003	X	X
StartUp_X.001	X	
USIC_AI.004	X	X
WDT_X.002	X	X

1) Fix of the Start-Up sequence is done.

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	XE16xxM	
	EES-AA ES-AA	AA ¹⁾

- none -

1) Fix of the Start-Up sequence is done.

4.3 Application Hints

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

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

Hint	XE16xxM	
	EES-AA ES-AA	AA ¹⁾
CAPCOM12_X.H001	X	X
CC6_X.H001	X	X
ECC_X.H001	X	X
GPT12E_X.H002	X	X
INT_X.H002	X	X
LXBUS_X.H001	X	X
MultiCAN_AI.H005	X	X
MultiCAN_AI.H006	X	X
MultiCAN_TC.H002	X	X
MultiCAN_TC.H003	X	X
MultiCAN_TC.H004	X	X
RESET_X.H003	X	X
RTC_X.H003	X	X
USIC_AI.H001	X	X

1) Fix of the Start-Up sequence is done.

4.4 Documentation Updates

The **Table 5** show the dependencies of documentation updates in the derivatives.

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

Documentation Updates	XE16xxM	
	EES-AA ES-AA	AA ¹⁾
ECC_X.D001	X	X
StartUp_X.D001	X	X

1) Fix of the Start-Up sequence is done.

5 Short Errata Description

This chapter gives an overview over the description and shows changes to the last Errata Sheet.

5.1 Errata removed in this Errata Sheet

Table 6 shows an overview of removed errata in this Errata Sheet. In column **Change** the reason for removal is given.

Table 6 Errata removed in this Errata Sheet

Errata	Short Description	Change
OCDS_AI.001	DAP restart lost when DAP0 inactive	Not valid for this product. ¹⁾

1) Erroneously put in Errata Sheet Rel. 1.1.

5.2 Functional Deviations

Table 7 shows a short description of the functional deviations.

Table 7 Functional Deviations

Functional Deviation	Short Description	Chg	Pg
ECC_X.002	Incorrect ECC Error Indication for DPRAM	New	18
ESR_X.001	ESREXSTAT1 and ESREXSTAT2 Status Bits are Cleared by Hardware		19
GPT12E_X.002	Effects of GPT Module Microarchitecture	New	19
HARR_X.001	Not Requested HARR after User Reset		20
OCDS_X.003	Peripheral Debug Mode Settings cleared by Reset		21
PARITY_X.001	PMTSR Register Initialization	New	22
RESET_X.003	P2.[2:0] and P10.[12:0] Switch to Input		22
StartUp_X.001	RAM Initialization before ECC/Parity Activation		22
USIC_AI.004	Receive shifter baudrate limitation		23
WDT_X.002	Clearing the Internal Flag which Stores Preceding WDT Reset Request	New	24

5.3 Deviations from Electrical and Timing Specification

Table 8 shows a short description of the electrical- and timing deviations from the specification.

Table 8 Deviations from Electrical and Timing Specification

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

5.4 Application Hints

The [Table 9](#) shows a short description of the application hints.

Table 9 Application Hints

Hint	Short Description	Chg	Pg
CAPCOM12_X.H001	Enabling or Disabling Single Event Operation	Up-date	27
CC6_X.H001	Modifications of Bit MODEN in Register CCU6x_KSCFG		28
ECC_X.H001	ECC Error Indication Permanently Set		28
GPT12E_X.H002	Reading of Concatenated Timers	New	29
INT_X.H002	Increased Latency for Hardware Traps		30
LXBUS_X.H001	Do Not Access Reserved Locations on the LXBus	New	31
MultiCAN_AI.H005	Txd Pulse upon short disable request	Up-date	31
MultiCAN_AI.H006	Time stamp influenced by resynchronization		31
MultiCAN_TC.H002	Double Synchronization of receive input		32
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		32
MultiCAN_TC.H004	Double remote request		32
RESET_X.H003	How to Trigger a PORST after an Internal Failure		33
RTC_X.H003	Changing the RTC Configuration		33
USIC_AI.H001	FIFO RAM Parity Error Handling		34

5.5 Documentation Updates

The [Table 10](#) gives a short description of the documentation updates.

Table 10 Documentation Updates

Documentation Updates	Short Description	Chg	Pg
ECC_X.D001	ECC Software Testing Support	New	35
StartUp_X.D001	Table "Configuration Data for Bootstrap Loader Modes"	New	35

6 Detailed Errata Description

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

6.1 Functional Deviations

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.001 ESREXSTAT1 and ESREXSTAT2 Status Bits are Cleared by Hardware

The status bits of register `ESREXSTAT1` and `ESREXSTAT2` are set as the trigger signal was active. The status bits are not stored and are only set as long as the external trigger inputs are active.

Workaround

None.

GPT12E_X.002 Effects of GPT Module Microarchitecture

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

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

The GPT2 state machine has 4 states (2 states when `BPS2 = 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:

- when T3 is counting **up**, `0000H` or `0001H` may be read from T3 directly after an overflow, although the reload value in T2/T4 is higher (`0001H` may be read in particular if `BPS1 = 01B` and `T3I = 000B`),

Detailed Errata Description

- 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.

HARR_X.001 Not Requested HARR after User Reset

Halt after Reset (HAR) request is configured after any power-on reset (\overline{PORST}). If no debugger is connected upon \overline{PORST} to the device no halt happens and

HAR request remains active. In case a debugger is attached later on (so-called hot-attach) and the debugger does not clear the HAR request, the device will be halted after the next User Reset.

Workaround

If working with a debugger and no HARR is desired, clear the HAR request before the next User Reset.

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. ADC0, ADC1, CC60...CC63, USIC0...USIC2, 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.

PARITY X.001 PMTSR Register Initialization

The PMTSR register content after start-up is 0100_H, 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_H.

RESET X.003 P2 . [2:0] and P10 . [12:0] Switch to Input

During a User Reset the port pins P2 . [2:0] and P10 . [12:0] are switched to input. The user has to check his application to this behaviour.

Workaround

None.

StartUp X.001 RAM Initialization before ECC/Parity Activation

As the User's Manual states, any RAM (PSRAM, DSRAM, DPRAM and SBRAM) that uses a kind of Content Protection (ECC/parity), needs to be initialized before the memory-control mechanism is activated. Because the built-in initialization does not work properly upon start-up with a Power Reset for V_{DDIM} and V_{DDI1} supply domains, the user software must perform following steps at its very beginning, if a memory-control is needed:

1. Check if the last start-up event has been a power-on in DMP_1 supply domain - after such event the RAMs contain random data and must be initialized,
 - if SCU_RSTSTAT1.ST1 \neq 11_B - no power-on in DMP_1, no initialization needed (it has already been performed) - exit this sequence;
 - if SCU_RSTSTAT1.ST1 = 11_B - initialization needed, continue with step 2.
2. Optional step,
 - if the application and the system allow a clock-frequency faster than 10 MHz

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

3. Select and activate the desired control-mechanism by installing the bits corresponding to target memories as follows:
 - if parity will be used - disable parity traps by setting `SCU_TRAPDIS.PET = 1B`, set `SCU_MCHKCON.SELxx = 1B` and `SCU_PEEN.PEENxx = 1B`;
 - if ECC will be used - set `SCU_ECCCON.xxEN = 1B` (ECC traps are disabled and ECC is selected as default control-mechanism after power-on).
4. Perform a write access to any location of the target memories

Note: The exact content written doesn't matter for the control-mechanism; the user can decide either to fill the memories with all zeroes or something else.

Note: When using the autoincrement mode to read the SBRAM and having ECC/parity enabled, it is necessary not only to initialize the used addresses but also the following one. This problem does not exist if the whole SBRAM is initialized.

After this sequence, the memories are ready to be used and the selected control-mechanism is active. It is a further decision of the user either to enable the respective trap (by resetting `SCU_TRAPDIS.ECCT/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 $f_{\text{sys}}/2$ instead of f_{sys} .

Workaround

None.

WDT_X.002 Clearing the Internal Flag which Stores Preceding WDT Reset Request

The information that the WDT has already been exceeded once is stored in an internal flag. In contrary to the documentation, that this flag can be cleared by writing a 1_B to bit `WDTCS.CLRIRF` at any time, clearing of the internal flag is only possible, when the WDT is in Prewarning Mode.

Workaround 1

Applications following the proposal of Application Note **AP16103** (section 'Using ESR pins to trigger a PORST reset') to trigger a Power-on Reset upon a WDT event will find the internal flag cleared upon the Power-on Reset and thus will have no issue with this limitation.

Workaround 2

In case the WDT triggers a User Reset upon a WDT overflow, the internal flag will not be cleared by the reset itself. Any further overflow of the WDT will lead to a permanent reset of the device.

Applications which intentionally let the WDT exceed once, e.g. in conjunction with an initial self test, might want to have the internal flag cleared to prevent a permanent reset upon a real WDT overflow.

If the internal flag shall be cleared by software, this must be done as a reaction on a WDT overflow in the time frame the WDT is in Prewarning Mode before the permanent User Reset will be triggered. The CPU is notified upon the WDT entering Prewarning Mode by issuing an interrupt request. The application can react on this request and clear the internal flag now by writing a 1_B to bit `WDTCS.CLRIRF` e.g. within an ISR.

Workaround 3

Some applications may not want to use or rely on the interrupt logic in conjunction with a WDT overflow event. The proposed remedy in this case is, to initiate a Power Reset to clear the internal flag by changing the settings of the active Supply Watchdog (SWD) as follows:

1. Disable SFR protection.

Detailed Errata Description

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

- none -

6.3 Application Hints

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 SEE_y for a channel CC_y 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 $SEE_y = 1_B$ (event for channel CC_y has not yet occurred)
- Meanwhile, event for CC_y occurs, and bit SEE_y is cleared to 0_B by hardware
- In the Write-Back stage, software writes CCn_SEE with bit $SEE_x = 1_B$ (intended event for CC_x enabled via OR instruction) **and** bit $SEE_y = 1_B$
- or, as inverse procedure, software writes CCn_SEE with bit $SEE_x = 0_B$ (intended event for CC_x disabled via AND instruction) **and** bit $SEE_y = 1_B$

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

To avoid this effect, one of the following solutions - depending on the characteristics of the application - is recommended to enable or disable further

compare events for CAPCOM channels concurrently operating in single event mode:

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

CC6_X.H001 Modifications of Bit MODEN in Register CCU6x_KSCFG

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

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

ECC_X.H001 ECC Error Indication Permanently Set

The ECC error flag of the `ECCSTAT` register for the DPRAM, DSRAM, PSRAM and SBRAM can not be cleared, if a memory location with an ECC error is selected and the ECC is enabled. The memory can be selected by an active or by the latest read or write access.

Workaround

Select a memory location without ECC error in the respective memory (e.g. make a read to another address) and then clear the ECC error flag. Be aware that the new selected address may also have an ECC error.

GPT12E_X.H002 Reading of Concatenated Timers

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

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

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

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

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 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.

LXBUS_X.H001 Do Not Access Reserved Locations on the LXBus

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

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

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

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

MultiCAN_AI.H005 Txd Pulse upon short disable request

If a CAN disable request is set and then canceled in a very short time (one bit time or less) then a dominant transmit pulse may be generated by MultiCAN module, even if the CAN bus is in the idle state.

Example for setup of the CAN disable request:

MCAN_KSCCFG.MODEN = 0 and then MCAN_KSCCFG.MODEN = 1

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_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 frequencies and high CAN baudrate, this delay may become significant and has to be taken into account when calculating the overall physical delay on the CAN bus (transceiver delay...).

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.

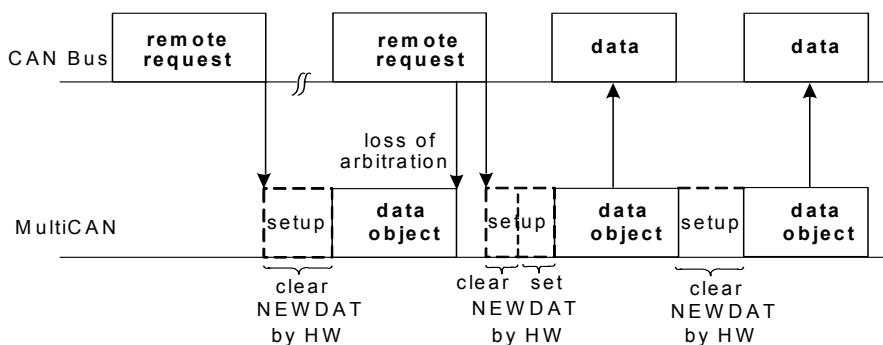


Figure 1 Loss of Arbitration

RESET_X.H003 How to Trigger a $\overline{\text{PORST}}$ after an Internal Failure

There is no internal User Reset that restores the complete device including the power system like a Power-On Reset. In some applications it is possible to connect $\overline{\text{ESR1}}$ or $\overline{\text{ESR2}}$ with the $\overline{\text{PORST}}$ pin and set the used $\overline{\text{ESR}}$ pin as Reset output. With this a WDT or Software Reset can trigger a Power-On Reset.

A detailed description is in the Application Note **AP16103**.

RTC_X.H003 Changing the RTC Configuration

The count input clock f_{RTC} for the Real Time Clock module (RTC) can be selected via bit field `RTCCLKSEL` in register `RTCCLKCON`. Whenever the system clock is less than 4 times faster than the RTC count input clock ($f_{\text{SYS}} <$

$f_{\text{RTC}} \times 4$), Asynchronous Mode must be selected (bit $\text{RTCCM} = 1_{\text{B}}$ 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 $\text{MODEN} = 0_{\text{B}}$ in register RTC_KSCCFG before register RTCCLKCON is modified, i.e. whenever

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

In case power domain DMP_1 is switched off, it is not required to switch the RTC to Asynchronous Mode, since it will receive a reset in any case.

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 $0x0$.

6.4 Documentation Updates

ECC_X.D001 ECC Software Testing Support

In the “ECC Error Handling” section of chapter 8.14.3 of the user’s manual version 1.2 the following update is done:

The ECC error detection can be triggered on purpose to test the detection itself and the associated trap routine. This test option is available for RAMs that can operate with ECC and with parity protection, i.e. the standard RAM areas (DPRAM, DSRAM, PSRAM, SBRAM).

The software based ECC test uses both ECC and parity checking. It can generate a single-bit ECC error by executing the following sequence:

- Select parity mode for the respective RAM ($\text{MCHKCON.SEL}_X = 1$)
- Read back MCHKCON to allow the parity logic to activate
- Write 2000_H to a location within this RAM (while parity is selected)
- Select ECC mode for the respective RAM ($\text{MCHKCON.SEL}_X = 0$)
- Read back MCHKCON to allow the ECC logic to activate
- Read from the chosen RAM location (while ECC is selected)

This will generate a single bit ECC error for databit 9:

If ECC operation is enabled ($\text{ECCCON.XEN} = 1$) the corrected data value (2000_H) is read and an ECC error is indicated.

If ECC operation is disabled ($\text{ECCCON.XEN} = 0$) the uncorrected data value (2200_H) is read and no error is indicated.

Note: Due to the structure of the PSRAM this test modifies a complete 8-Byte memory line (...000_B - ...111_B). A non-destructive test, therefore, must save and restore all 4 words of the respective memory line.

StartUp_X.D001 Table "Configuration Data for Bootstrap Loader Modes"

In the “Summary of Bootstrap Loader Modes” section of chapter 12.6.5 of the user’s manual version 1.2 the following table was updated (changes are **red**).

Table 12 Configuration Data for Bootstrap Loader Modes

Bootstrap Loader Mode	Configuration on P10.[7:0] ¹⁾	Receive Line from Host	Transmit Line to Host	Transferred Data	Supported Host Speed
Standard UART	xxxx x110 _B ²⁾	RxD=P7.4	TxD=P7.3	32 bytes	2.4 - 19.2 kbit/s
	xxxx x010 _B	RxD=P2.4	TxD=P2.3		
Enhanced UART	x001 0000 _B	RxD=P2.4	TxD=P2.3	l bytes ³⁾	2.4 - 19.2 kbit/s at start, then changeable by Header
Sync. Serial	xxxx 1001 _B	MRST=P2.4	MTSR=P2.3 SCLK=P2.5 SLS=P2.6	m bytes ⁴⁾	--- (controlled by device)
MultiCAN	xxxx 1101 _B	RxDC0=P2.6	TxDC0=P2.5	8 x n bytes ⁵⁾	125 - 500 kbit/s

- 1) x means that the level on the corresponding pin is irrelevant.
- 2) Not available in 64-pin package.
- 3) l = Code_Length sent by the host, the values allowed are 1...(PSRAM_size-256).
- 4) m = data size read from EEPROM, the values allowed are 1...(PSRAM_size-256).
- 5) n = DMSGC, Data Message Count sent by the host with Initialization frame, the values allowed are 1... (PSRAM_size-256)/8.

www.infineon.com

Published by Infineon Technologies AG