Errata Sheet

Rel. 1.0, 2018-12-20

Device	**TC37xEXT**
Marking/Step	**(E)ES-AA**
Package	**see Data Sheet**

This Errata Sheet describes the deviations from the current user documentation.

Table 1 Current Documentation[1)]

AURIX™ TC3xx User's Manual	V1.0.0	2018-08
AURIX™ TC37xEXT Appendix to User's Manual	V1.0.0	2018-08
TC37x AA-Step Target Data Sheet	V0.4	2018-01
TriCore TC1.6.2 Core Architecture Manual: - Core Architecture (Vol. 1) - Instruction Set (Vol. 2)	 V1.0 V1.0	 2017-01-11 2017-01-11
AURIX™ TC3xx ED Target Specification[2)]	V2.5.1	2018-04

1) Newer versions replace older versions, unless specifically noted otherwise.

2) Distribution under NDA, only relevant for tool development not for application development.

Make sure you always use the corresponding documentation for this device (User's Manual, Data Sheet, Documentation Addendum (if applicable), TriCore Architecture Manual, Errata Sheet) available in category 'Documents' at **www.infineon.com/AURIX** and **www.myInfineon.com**.

Conventions used in this document

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
- **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.

Notes

1. This Errata Sheet applies to all temperature and frequency versions and to all memory size variants, unless explicitly noted otherwise. For a synopsis of the available variants, see the latest Data Sheet/User’s Manual and the addendum “AURIX™ TC3x Variants” of the corresponding TC3x device.
   This Errata Sheet covers several device versions. If an issue is related to a particular module, and this module is not specified for a specific device version, this issue does not apply to this device version.
   E.g. issues with identifier “EBU” do not apply to devices where no EBU is specified, and issues with identifier “RIF” only apply to “ADAS” devices.
2. 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.
   The specific test conditions for EES and ES are documented in a separate Status Sheet.
3. 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.

1 History List / Change Summary

Table 2 History List

Version	Date	Remark
1.0	2018-12-20	This is the first version for TC37xEXT step AA

Table 3 Functional Deviations

Functional Deviation	Short Description	Change	Page
BROM_TC.013	CAN BSL does not send error message if no valid baudrate is detected		14
BROM_TC.014	Lockstep Comparator Alarm for CPU0 after Warm PORST, System or Application Reset if Lockstep is disabled		14
BROM_TC.016	Uncorrectable ECC error in Boot Mode Headers		15
DAP_TC.005	DAP client_read: dirty bit feature of Cerberus’ Triggered Transfer Mode		15
DAP_TC.007	Incomplete client_blockread telegram in DXCM mode when using the “read CRCup” option		16
DAP_TC.010	Performance when accessing EMEM in UWM and WM modes		16
DMA_TC.059	ACCEN Protection not implemented for ERRINTRr		16
FlexRay_AI.087	After reception of a valid sync frame followed by a valid non-sync frame in the same static slot the received sync frame may be ignored		17

Table 3 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
FlexRay_AI.088	A sequence of received WUS may generate redundant SIR.WUPA/B events		18
FlexRay_AI.089	Rate correction set to zero in case of SyncCalcResult=MISSING_TERM		18
FlexRay_AI.090	Flag SFS.MRCS is set erroneously although at least one valid sync frame pair is received		19
FlexRay_AI.091	Incorrect rate and/or offset correction value if second Secondary Time Reference Point (STRP) coincides with the action point after detection of a valid frame		20
FlexRay_AI.092	Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00		20
FlexRay_AI.093	Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames		21
FlexRay_AI.094	Sync frame overflow flag EIR.SFO may be set if slot counter is greater than 1024		22
FlexRay_AI.095	Register RCV displays wrong value		23
FlexRay_AI.096	Noise following a dynamic frame that delays idle detection may fail to stop slot		23
FlexRay_AI.097	Loop back mode operates only at 10 MBit/s		24
FlexRay_AI.099	Erroneous cycle offset during startup after abort of startup or normal operation		25
FlexRay_AI.100	First WUS following received valid WUP may be ignored		26
FlexRay_AI.101	READY command accepted in READY state		26

Table 3 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
FlexRay_AI.102	Slot Status vPOC!SlotMode is reset immediately when entering HALT state		27
FlexRay_AI.103	Received messages not stored in Message RAM when in Loop Back Mode		27
FlexRay_AI.104	Missing startup frame in cycle 0 at coldstart after FREEZE or READY command		28
FlexRay_AI.105	RAM select signals of IBF1/IBF2 and OBF1/OBF2 in RAM test mode		29
FlexRay_AI.106	Data transfer overrun for message transfers Message RAM to Output Buffer (OBF) or from Input Buffer (IBF) to Message RAM		30
GETH_AI.001	Packets with Destination Address (DA) mismatch are delayed until EOP is received in threshold (cut-through) mode		33
GETH_AI.002	Incorrect Weighted Round Robin Arbitration between Tx and Rx DMA Channels to Access the common Host Bus		34
GETH_AI.003	Header-Payload Split Function Does Not Support IPv6 Packets Received With Zero TCP Payload		36
GETH_AI.005	Application Error Along With Start-of-Packet Can Corrupt the Ongoing Transmission of MAC Generated Packets		37
GETH_AI.006	Incorrect IP Header or Payload Checksum Status Given After MTL TX FIFO Flush		38
GETH_AI.007	IEEE 1588 Timestamp Interrupt Status Bits are Incorrectly Cleared on Write Access to the CSR Register with Similar Offset Address		39

Table 3 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
GETH_AI.008	Application Error Along with Start-of-Packet Can Corrupt the FCS Field of the Previous Frame in the MAC Pipeline		40
GETH_AI.009	Corrupted Rx Descriptor Write Data		41
GETH_AI.010	Fatal Bus Error Interrupt Might Be Generated for Incorrect DMA Channel		42
GETH_AI.011	Receive Queue Overflow at End of Frame Along with SPRAM Read-Write Conflict Can Cause Data Loss		43
GETH_AI.012	Incorrect Flexible PPS Output Interval When Fine Time Correction Method is Used		43
GETH_AI.013	False Dribble and CRC Error Reported in RMII PHY 10Mbps Mode		45
GTM_AI.254	TIM TDU: TDU_STOP=b101 not functional		46
GTM_AI.262	DPLL: PSSC behavior in mode change (DPLL_CTRL_0.RMO = 0 ->1)		46
GTM_AI.263	DPLL: DPLL_STATUS.LOCK1 flag (0 ->1) delayed after direction change when DPLL operating in DPLL_CTRL_0.RMO = 1		47
GTM_AI.304	MCS: Scheduling modes Single Prioritization and Multiple Prioritization are not functional		48
GTM_AI.305	TIM Signal Generation with serial shift mode TSSM: If TSSM_OUT is used in channel x and channel x+1 uses edges of FOUT_PREV, these edges show an unexpected delay which lead to a delayed operation of channel measurement or TDU functionality of channel x+1		49

Table 3 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
GTM_AI.306	DPLL: DPLL_NUTC.syn_t_old, DPLL_NUSC.syn_s_old not updated according specification		50
GTM_AI.307	IRQ: AEI_IM_ADDR is not set in GTM_IRQ_NOTIFY register if cluster 0 is disabled		51
GTM_AI.308	TIM, ARU: Limitation that back-to-back TIM data transfers at full ARU clock rate cannot be transferred correctly with ARU dynamic routing feature		52
GTM_AI.309	TIM Signal Generation with serial shift mode TSSM in channel x: Generated TSSM_OUT signal used in lookup table of inputsrc module of channel x has unpredictable delay		52
GTM_AI.318	MCS: NARD(I) instruction terminates unexpectedly		53
GTM_AI.319	(A)TOM: Unexpected (A)TOM_CCU1TCx_IRQ in up/down counter mode		54
GTM_AI.320	ATOM: Unexpected restart of a SOMS oneshot cycle while ATOM[i]_CH[x]_CM0 is zero		54
GTM_AI.322	DPLL: PSTC, PSSC not updated correctly after fast pulse correction completed (DPLL_CTRL1.PCM1/2 = 0)		55
GTM_AI.323	DPLL: Registers DPLL_NUTC.SYN_T and DPLL_NUSC.SYN_S are updated by the profile (ADT_T.NT/ADT_S.NS) before the DPLL is synchronized (DPLL_STATUS.SYT/S=0)		56

Table 3 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
GTM_AI.325	TIM: Bits ACB[2:1] lost on interface to ARU (always zero)		57
GTM_AI.326	TIM: ARU bit ACB[0] (signal level) incorrect in case a second ARU request occurs while the actual request is just acknowledged		59
GTM_AI.329	Interference of MCS to AEI/ADC and CPU to AEI traffic within the same cluster could result in incorrect MCS program execution		59
GTM_AI.331	GTM_TIM[i]_AUX_IN_SRC and GTM_EXT_CAP_EN_[i] register: wrong status 2 by AEI write access if cluster 0 is disabled		62
GTM_AI.332	Access to registers GTM_TIM[i]_AUX_IN_SRC and GTM_EXT_CAP_EN_[i] via legacy address space: read data always 0 for AEI read access while cluster 0 is disabled		63
GTM_AI.333	MCS bus master interface: a not word aligned address access to DPLL ram region can cause incorrect execution of MCS channel code		63
GTM_AI.334	DPLL RAM content of single address can be corrupted after leaving debug mode		64
GTM_TC.018	DPLLRAM trace data can be wrong		65
GTM_TC.019	ARU can not be traced if GTM cluster 5 is disabled		66
GTM_TC.020	Debug/Normal read access control via bit field ODA.DRAC		66
HSCT_TC.012	HSCT sleep mode not supported		67

Table 3 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
HSCT_TC.013	Internal Loopback Mode not reliable		67
MCDS_TC.052	TriCore wrap around write access causes redundant MCDS message		68
MCDS_TC.064	ACCEN0 register write not supervisor protected		68
MCDS_TC.065	Selection of SRI trace sources		68
MCMCAN_AI.015	Edge filtering causes mis-synchronization when falling edge at Rx input pin coincides with end of integration phase		69
MCMCAN_AI.017	Retransmission in DAR mode due to lost arbitration at the first two identifier bits		71
MCMCAN_AI.018	Tx FIFO Message Sequence Inversion		72
MCMCAN_AI.019	Unexpected High Priority Message (HPM) interrupt		74
MTU_TC.012	Security of CPU Cache Memories During Runtime is Limited		76
MTU_TC.017	Unexpected alarms after application reset		77
MTU_TC.018	Gated SRAM alarms		77
PER_PLL_TC.001	Peripheral PLL weakness for 25 MHz input clock when using Divider Bypass		79
PMS_TC.005	Voltage rise at P33 and P34 up to 2.5V during start-up and power-down		82
PMS_TC.006	PORST not released during Cold Power-on Reset until VDDM is available		82
PMS_TC.007	VDDP3 or VDD Overvoltage during start-up may not be detected by PBIST		83
QSPI_TC.006	Baud rate error detection in slave mode (error indication in current frame)		84
QSPI_TC.009	USR Events for PT1=2 (SOF: Start of Frame)		84

Table 3 **Functional Deviations** (cont'd)

Functional Deviation	Short Description	Change	Page
QSPI_TC.010	Move Counter Mode - USR Events for PT1=4 (RBF: Receive Buffer Filled)		85
QSPI_TC.013	Slave: No RxFIFO write after transmission upon change of BACON.MSB		85
QSPI_TC.014	Slave: Incorrect parity bit upon TxFIFO underflow		86
SCR_TC.015	Bit SCU_PMCON1.WCAN_DIS does not disable WCAN PCLK input		86
SCR_TC.016	DUT response to first telegram has incorrect C_START value		86
SCR_TC.018	SSC Receive FIFO not working		87
SCR_TC.019	Accessing the XRAM while SCR is in reset state		87
SCR_TC.020	Stored address in mon_RETH may be wrong after a break event		88

Table 4 **Deviations from Electrical- and Timing Specification**

AC/DC/ADC Deviation	Short Description	Change	Page
ADC_TC.P009	Increased TUE for G10 when using Alternate Reference		89

Table 5 Application Hints

Hint	Short Description	Change	Page
ADC_TC.H026	Additional Waiting Phase in Slow Standby Mode		90
AGBT_TC.H004	Configuration of registers PYCR2 and PACR2		90
ASCLIN_TC.H001	Bit field FRAMECON.IDLE in LIN slave mode		90
BROM_TC.H008	CAN BSL does not support DLC = 9 and DLC = 11		91
BROM_TC.H009	Re-Enabling Lockstep via BMHD		91
BROM_TC.H011	Assertion of ALM7[14] after Cold/Warm PORST		91
BROM_TC.H012	Availability of V_{DDSB} during start-up		92
BROM_TC.H014	SSW behavior in case of wrong state or uncorrectable error in UCBs - Documentation Update		93
CCU_TC.H012	Configuration of the Oscillator- Documentation Update		93
CPU_TC.H016	List of OS and I/O Privileged Instructions - Documentation Update		94
FlexRay_AI.H004	Only the first message can be received in External Loop Back mode		94
FlexRay_AI.H005	Initialization of internal RAMs requires one eray_bclk cycle more		95
FlexRay_AI.H006	Transmission in ATM/Loopback mode		95
FlexRay_AI.H007	Reporting of coding errors via TEST1.CERA/B		95
FlexRay_AI.H009	Return from test mode operation		96
FlexRay_TC.H003	Initialization of E-Ray RAMs		96
GETH_AI.H001	Preparation for Software Reset		97

Table 5 Application Hints (cont'd)

Hint	Short Description	Change	Page
GTM_TC.H010	Trigger Selection for EVADC and EDSADC		98
GTM_TC.H019	Register GTM_RST - Documentation Update		98
I2C_TC.H008	Handling of RX FIFO Overflow in Slave Mode		99
MCDS_TC.H007	Program trace of CPUx (x > 0) program start not correct		99
MCMCAN_TC.H006	Unintended Behavior of Receive Timeout Interrupt		100
MTU_TC.H013	First Write Access to SSH registers of SCR_RAMINT		100
MTU_TC.H015	ALM7[0] may be triggered after cold PORST		101
OCDS_TC.H014	Avoiding failure of key exchange command due to overwrite of COMDATA by firmware		101
OCDS_TC.H015	System or Application Reset while OCDS and lockstep monitoring are enabled		102
OCDS_TC.H016	Release of application reset via OJCONF may fail		103
SCR_TC.H009	RAM ECC Alarms in Standby Mode		104
SCR_TC.H010	HRESET command erroneously sets RRF flag		104
SCR_TC.H011	Hang-up when warm PORST is activated during Debug Monitor Mode		104
SCR_TC.H012	Reaction in case of XRAM ECC Error		105
SDMMC_TC.H001	Idle State of SDMMC0_CLK		106
SMU_TC.H010	Clearing individual SMU flags: use only 32-bit writes		106

Table 5 Application Hints (cont'd)

Hint	Short Description	Cha nge	Pa ge
SMU_TC.H012	Handling of SMU alarms ALM7[1] and ALM7[0]		107
SRI_TC.H001	Using LDMST and SWAPMSK.W instructions on SRI mapped Peripheral Registers (range 0xF800 0000-0xFFFF FFFF)		107

2 Functional Deviations

BROM_TC.013 CAN BSL does not send error message if no valid baudrate is detected

If the CAN Bootstrap loader (BSL) is unable to determine the baudrate from the initialization message sent by the host, it does not send the error message as defined in table “Error message (No baudrate detected)” in chapter “AURIX™ TC3xx Platform Firmware”, but enters an endless loop with no activity on external pins.

Workaround

If the external host does not receive Acknowledgment Message 1 from the CAN BSL within the expected time (~5 ms), it should check the integrity of the connection, and then may reset the TC3xx to restart the boot procedure.

BROM_TC.014 Lockstep Comparator Alarm for CPU0 after Warm PORST, System or Application Reset if Lockstep is disabled

Lockstep monitoring may be disabled in the Boot Mode Header structure (BMHD) for each CPUx with lockstep functionality (including CPU0). The startup software (SSW) will initially re-enable lockstep upon the next reset trigger.

If lockstep is disabled for CPU0, and the next reset is a warm PORST, System or Application reset, a lockstep comparator alarm will be raised for CPU0.

Note: This effect does not occur for CPUx, x>0.

Workaround

Do not disable lockstep for CPU0, always keep lockstep on CPU0 enabled.

Non-safety applications may ignore the lockstep comparator alarm for CPU0.

BROM_TC.016 Uncorrectable ECC error in Boot Mode Headers

If one or more boot mode headers UCB_BMHDx_ORIG or UCB_BMHDx_COPY contain an uncorrectable ECC error (4-bit error) in the BMI, BMHDID, STAD, CRCBMHD or CRCBMHD_N fields, firmware will end up in an irrecoverable state resulting in a device not being able to boot anymore.

This may happen in the following scenarios:

- Power-loss during BMHD reprogramming or erase
- Over-programming of complete BMHD contents.

Workaround

- Ensure continuous power-supply during BMHD reprogramming and erase using power monitoring including appropriate configuration.
- Avoid over-programming of BMHD contents.
- Ensure that also in any BMHDx_ORIG or _COPY unused in the application, the above fields are in a defined ECC-error free state (e.g. clear them to 0).

DAP_TC.005 DAP client_read: dirty bit feature of Cerberus' Triggered Transfer Mode

Note: This problem is only relevant for tool development, not for application development.

The DAP telegram client_read reads a certain number of bits from an IOclient (e.g. Cerberus). The parameter k can be selected to be zero, which is supposed to activate reading of 32 bits plus dirty bit.

However, in the current implementation, the dirty bit feature does not work correctly.

It is recommended not to use this dirty bit feature, meaning the number k should not evaluate to “0”.

DAP_TC.007 Incomplete client_blockread telegram in DXCM mode when using the “read CRCup” option

In DXCM (DAP over CAN Messages) mode, the last parcel containing the CRC32 might be skipped in a client_blockread telegram using the “read CRCup” option.

Workaround

Do not use CRCup option with client_blockread telegrams in DXCM mode.

Instead the CRCup can be read by a dedicated getCRCup telegram.

DAP_TC.010 Performance when accessing EMEM in UWM and WM modes

Note: This problem is only relevant for development tools and their device connection.

The read bandwidth of DAP for accessing EMEM is about 25 Mbyte/s in Unidirectional Wide Mode (UWM) and Wide Mode (WM) if the DAP is clocked at 160 MHz. This is lower than the target value of 30 Mbyte/s. The bandwidth is limited by the long path from DAP to EMEM.

If the DAP frequency is below 125 MHz, this effect is hidden behind other delays. Therefore below this frequency the bandwidth will be proportional to the DAP frequency. At 125 MHz a bandwidth of 23.5 Mbyte/s was measured in simulation.

The details of WM and UWM are described in the section “DAP Modes and Options” in the OCDS chapter of the device documentation.

Note: The bandwidth measurement is conducted with the BBB frequency of 150 MHz.

DMA_TC.059 ACCEN Protection not implemented for ERRINTRr

In the current documentation, the debug feature Error Interrupt Set Register ERRINTRr for Resource Partition r (r = 0..3) is specified as access enable protected (symbol “Pr” in column Access Mode/Write) in table “Register Overview” of the DMA chapter.

However, in this design step, register ERRINTRr (r = 0..3) is not implemented as access enable protected.

Workaround

None.

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

Description:

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

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.088 A sequence of received WUS may generate redundant `SIR.WUPA/B` events

Description:

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

Scope:

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

Effects:

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

Workaround

Ignore redundant `SIR.WUPA/B` events.

FlexRay_AI.089 Rate correction set to zero in case of SyncCalcResult=MISSING_TERM

Description:

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

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.090 Flag `SFS.MRCS` is set erroneously although at least one valid sync frame pair is received

Description:

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

Scope:

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

Effects:

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

Workaround

Correct configuration of gSyncNodeMax.

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

Description:

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

Scope:

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

Effects:

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

Workaround

Configure action point offset smaller than static frame length.

FlexRay_AI.092 Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00

Description:

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

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.093 Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames

Description:

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

Scope:

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

Effects:

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

Workaround

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

<u>FlexRay_AI.094</u> Sync frame overflow flag `EIR.SFO` may be set if slot counter is greater than 1024

Description:

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

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.095 Register RCV displays wrong value

Description:

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

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.096 Noise following a dynamic frame that delays idle detection may fail to stop slot

Description:

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

Scope:

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

Effects:

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

Workaround

None.

FlexRay_AI.097 Loop back mode operates only at 10 MBit/s

Description:

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

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.099 Erroneous cycle offset during startup after abort of startup or normal operation

Description:

An abort of startup or normal operation by a READY command near the macotick border may lead to the effect that the state INITIALIZE_SCHEDULE is one macrotick too short during the first following integration attempt. This leads to an early cycle start in state INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK.

As a result the integrating node calculates a cycle offset of one macrotick at the end of the first even/odd cycle pair in the states INTEGRATION_COLDSTART_CHECK or INTEGRATION_CONSISTENCY_CHECK and tries to correct this offset.

If the node is able to correct the offset of one macrotick (pOffsetCorrectionOut >> gdMacrotick), the node enters NORMAL_ACTIVE with the first startup attempt.

If the node is not able to correct the offset error because pOffsetCorrectionOut is too small (pOffsetCorrectionOut ≤ gdMacrotick), the node enters ABORT_STARTUP and is ready to try startup again. The next (second) startup attempt is not effected by this erratum.

Scope:

The erratum is limited to applications where READY command is used to leave STARTUP, NORMAL_ACTIVE, or NORMAL_PASSIVE state.

Effects:

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

Workaround

With a configuration of pOffsetCorrectionOut >> gdMacrotick • (1+cClockDeviationMax) the node will be able to correct the offset and therefore also be able to successfully integrate.

FlexRay_AI.100 First WUS following received valid WUP may be ignored

Description:

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

Scope:

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

Effects:

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

Workaround

None.

FlexRay_AI.101 READY command accepted in READY state

Description:

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

Scope:

The erratum is limited to the READY state.

Effects:

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

Workaround

None.

FlexRay_AI.102 Slot Status vPOC!SlotMode is reset immediately when entering HALT state

Description:

When the protocol engine is in the states NORMAL_ACTIVE or NORMAL_PASSIVE, a HALT or FREEZE command issued by the Host resets vPOC!SlotMode immediately to SINGLE slot mode (`CCSV.SLM[1:0]` = 00_B). According to the FlexRay protocol specification, the slot mode should not be reset to SINGLE slot mode before the following state transition from HALT to DEFAULT_CONFIG state.

Scope:

The erratum is limited to the HALT state.

Effects:

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

Workaround

None.

FlexRay_AI.103 Received messages not stored in Message RAM when in Loop Back Mode

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

Scope:

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

Effects:

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

Workaround

Leave HALT state by hardware reset.

<u>FlexRay_AI.104</u> Missing startup frame in cycle 0 at coldstart after FREEZE or READY command

When the E-Ray is restarted as leading coldstarter after it has been stopped by FREEZE or READY command, it may happen, depending on the internal state of the module, that the E-Ray does not transmit its startup frame in cycle 0. Only E-Ray configurations with startup frames configured for slots 1 to 7 are affected by this behaviour.

Scope:

The erratum is limited to the case when a coldstart is initialized after the E-Ray has been stopped by FREEZE or READY command. Coldstart after hardware reset is not affected.

Effects:

During coldstart it may happen that no startup frame is sent in cycle 0 after entering COLDSTART_COLLISION_RESOLUTION state from COLDSTART_LISTEN state.

Severity:

Low, as the next coldstart attempt is no longer affected. Coldstart sequence is lengthened but coldstart of FlexRay system is not prohibited by this behaviour.

Workaround

Use a static slot greater or equal 8 for the startup / sync message.

FlexRay_AI.105 RAM select signals of IBF1/IBF2 and OBF1/OBF2 in RAM test mode

When accessing Input Buffer RAM 1,2 (IBF1,2) or Output Buffer RAM 1,2 (OBF1,2) in RAM test mode, the following behaviour can be observed when entering RAM test mode after hardware reset.

- Read or write access to IBF2:
  - In this case also IBF1 RAM select **eray_ibf1_cen** is activated initiating a read access of the addressed IBF1 RAM word. The data read from IBF1 is evaluated by the respective parity checker.
- Read or write access to OBF1:
  - In this case also OBF2 RAM select **eray_obf2_cen** is activated initiating a read access of the addressed OBF2 RAM word. The data read from OBF2 is evaluated by the respective parity checker.

If the parity logic of the erroneously selected IBF1 resp. OBF2 detects a parity error, bit **MHDS.PIBF** resp. **MHDS.POBF** in the E-Ray Message Handler Status register is set although the addressed IBF2 resp. OBF1 had not error. The logic for setting **MHDS.PIBF** / **MHDS.POBF** does not distinguish between set conditions from IBF1 or IBF2 resp. OBF1 or OBF2.

Due to the IBF / OBF swap mechanism as described in section 5.11.2 in the E-Ray Specification, the inverted behaviour with respect to IBF1,2 and OBF1,2 can be observed depending on the IBF / OBF access history.

Scope:

The erratum is limited to the case when IBF1,2 or OBF1,2 are accessed in RAM test mode. The problem does not occur when the E-Ray is in normal operation mode.

Effects:

When reading or writing IBF1,2 / OBF1,2 in RAM test mode, it may happen, that the parity logic of IBF1,2 / OBF1,2 signals a parity error.

Severity:

Low, workaround available.

Workaround

For RAM testing after hardware reset, the Input / Output Buffer RAMs have to be first written and then read in the following order: IBF1 before IBF2 and OBF2 before OBF1

FlexRay_AI.106 Data transfer overrun for message transfers Message RAM to Output Buffer (OBF) or from Input Buffer (IBF) to Message RAM

The problem occurs under the following conditions:

1) A received message is transferred from the Transient Buffer RAM (TBF) to the message buffer that has its data pointer pointing to the first word of the Message RAM's Data Partition located directly after the last header word of the Header Partition of the Last Configured Buffer as defined by **MRC.LCB**.

2) The Host triggers a transfer from / to the Last Configured Buffer in the Message RAM with a specific time relation to the start of the TBF transfer described under 1).

Under these conditions the following transfers triggered by the Host may be affected:

a) Message buffer transfer from Message RAM to OBF

When the message buffer has its payload configured to maximum length (**PLC** = 127), the OBF word on address 00h (payload data bytes 0 to 3) is overwritten with unexpected data at the end of the transfer.

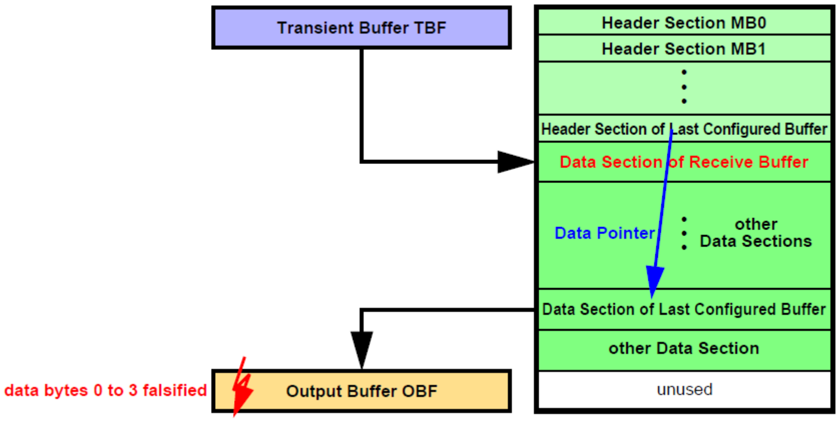
Transient Buffer TBF
Header Section MB0
Header Section MB1
Header Section of Last Configured Buffer
Data Section of Receive Buffer
Data Pointer
other Data Sections
Data Section of Last Configured Buffer
other Data Section
unused
data bytes 0 to 3 falsified
Output Buffer OBF

Figure 1 Message buffer transfer from Message RAM to OBF

b) Message buffer transfer from IBF to Message RAM

After the Data Section of the selected message buffer in the Message RAM has been written, one additional write access overwrites the following word in the Message RAM which might be the first word of the next Data Section.

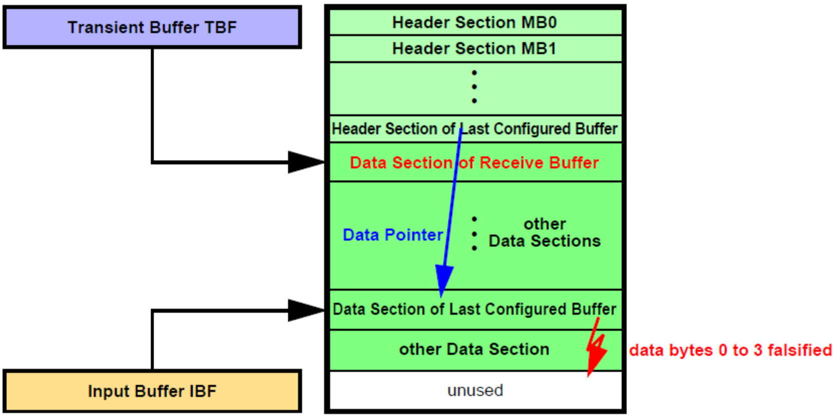
Transient Buffer TBF
Header Section MB0
Header Section MB1
Header Section of Last Configured Buffer
Data Section of Receive Buffer
Data Pointer
other Data Sections
Data Section of Last Configured Buffer
other Data Section
data bytes 0 to 3 falsified
Input Buffer IBF
unused

Figure 2 Message buffer transfer from IBF to Message RAM

Scope:

The erratum is limited to the case when (see **Figure 3** “Bad Case”):

1) The first Data Section in the Data Partition is assigned to a receive buffer (incl. FIFO buffers)

AND

2) The Data Partition in the Message RAM starts directly after the Header Partition (no unused Message RAM word in between)

Effects:

a) When a message is transferred from the Last Configured Buffer in the Message RAM to the OBF and **PLC** = 127 it may happen, that at the end of the transfer the OBF word on address 00h (payload data bytes 0 to 3) is overwritten with unexpected data (see **Figure 1**).

b) When a message is transferred from IBF to the Last Configured Buffer in the Message RAM, it may happen, that at the end of the transfer of the Data Section one additional write access overwrites the following word, which may be the first word of another message's Data Section in the Message RAM (see **Figure 2**).

Severity:

Medium, workaround available, check of configuration necessary.

Workaround

1) Leave at least one unused word in the Message RAM between Header Section and Data Section.

OR

2) Ensure that the Data Section directly following the Header Partition is assigned to a transmit buffer.

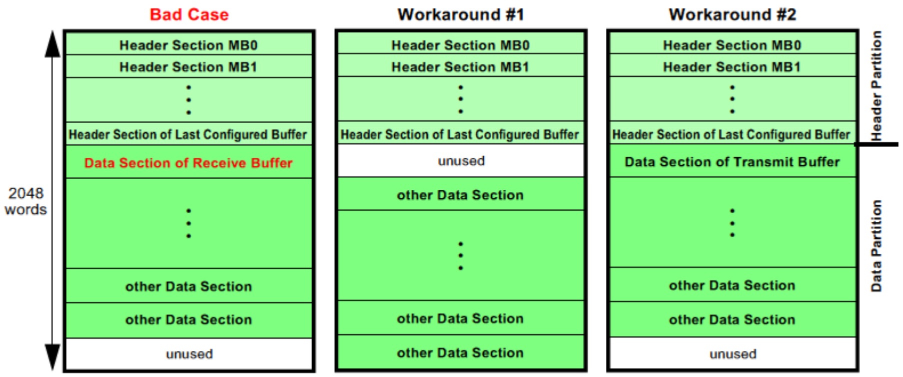

Figure 3 Message RAM Configurations

<u>GETH_AI.001</u> Packets with Destination Address (DA) mismatch are delayed until EOP is received in threshold (cut-through) mode

For each received packet, Header status is created by the MAC receiver based on the parsing of the Ethernet/VLAN/IP Header fields and forwarded to the DMA. This header status includes information about the size of the L2/L3/L4 header data (SPLIT_HDR_EN configurations) and/or the DMA Channel (NUM_DMA_RX_CH>1 configuration) which will forward the packet to host memory. The DA match result would provide DMA channel information based on the DCS field in the corresponding MAC Address Register that matched the DA field.

Due to this defect, instead of waiting for the DA match operation to complete, the design was waiting for a successful DA match to happen. If a DA match did not happen, the Header Status was being generated at the time of receiving the End of Packet (EoP).

The MTL Rx Queue controller waits for the Header status, stores it before it forwards the packet to target Rx DMA. Since the packets without a successful DA match were not getting the header status until the EoP, MTL Rx Queue controller forwards the packet only after the EoP is received in cut through mode.

Impacted Use Cases:

The DA mismatch packets will be forwarded only when Receive All or Promiscuous mode is set. In other use-cases, packets with DA mismatch will get dropped by the MTL Rx Queue controller and never reach the RxDMA.

Consequence:

Additional/un-necessary latency is introduced in the transfer of received packets with DA mismatch in the MTL Rx Queue operating in threshold (cut-through) mode. Effectively, it operates in store and forward mode for such packets.

Method of Reproducing:

1. Enable Receive All or Promiscuous mode for the receiver by programming MAC_Packet_Filter register.
2. Enable Threshold (Cut-through) mode and program the threshold value by writing to RSF and RTC fields of MTL_RxQ<n>_Operation_Mode.
3. When a packet with a packet length greater than threshold value is received, and a DA match does not happen, the packet will be read out of MTL Rx FIFO only after the EoP is received, while the expected behavior would have been to read the packet after the threshold is crossed.

Workaround:

None.

<u>GETH_AI.002</u> Incorrect Weighted Round Robin Arbitration between Tx and Rx DMA Channels to Access the common Host Bus

The DWC_EQOS has independent Transmit (Tx) and Receive (Rx) DMA engines. The transaction requests from the Tx and Rx DMA engines are arbitrated to allow access to the common AHB or DMA master interface. The following two types of arbitrations are supported by programming Bit[1] of the DMA_Mode register

- Weighted Round-Robin Arbitration
- Fixed-Priority Arbitration

The Bits[14:12] control the ratio of the weights between the Tx DMA and the Rx DMA in the Weighted Round Robin scheme. Table 11-3 in the DesignWare Cores Ethernet Quality-of-Service Databook, Version 4.21a explains the expected behavior.

However, due to this defect, programmed Priority Ratio (Bit[14:12] - PR) in Weighted-Round Robin scheme is not adhered to, when Rx DMA is given higher priority over Tx DMA or vice-versa. This is due to clock-cycle gaps present between the completion of one transfer initiated by Rx (or Tx) DMA and the next request for a transfer from Rx (or Tx) DMA. In this gap, the arbiter allocates the bus to the request from Tx (or Rx) DMA. Therefore, the arbiter operates in a 1:1 (round-robin) scheme even though higher ratio is given to the requests from Rx (or Tx) DMA.

In the Rx DMA engine, the gaps are introduced due to latency involved during arbitration across multiple Rx Queues (if selected), arbitration across multiple Rx DMA Channels (if selected), and inherent latency inside the Rx DMA controller. Similarly, in the Tx DMA engine, the gaps are introduced due to latency involved during arbitration across multiple Tx DMA Channels (if selected).

The arbiter module is fixed to delay the arbitration to absorb the various latencies. This allows the possibility of successful consecutive transfers from Tx or Rx DMA engines as per the programmed Priority Ratio. Also, the delay introduced on the Rx path due to arbitration across Rx Queues must be eliminated by programming bit[3] (RXQ_FRM_ARBIT) of MTL_RxQx_Control register to 1.

Impacted Use Cases:

When all the following conditions are met in the use case:

1. "Weighted Round Robin" arbitration scheme is selected by programming Bit[1] of the DMA_Mode register to 0
2. Programming different weights in the TXPR and PR fields of DMA_Mode register.
3. Only One Queue and One DMA is enabled.
4. Both Tx and Rx DMAs are simultaneously requesting for access.

Consequence:

The expected QoS (Quality of Service) requirement between Tx and Rx DMA Channels for host bus bandwidth allocation might not get adhered to. This defect might have an impact only if the host bus bandwidth is limited and less than or a little more than the total Ethernetline rate traffic. The impact can be in terms of Buffer Underflow (in TX when it is cut-through mode) or Buffer overflows (in RX). If the host side bandwidth is much more than the Ethernet line rate traffic, then this bandwidth allocation of WRR scheme is of no consequence.

Workaround:

Operate in Fixed Priority arbitration mode (DA=1) with Rx DMA having higher priority over Tx (TXPR=0). Operate the Tx buffers in Store-and-Forward mode to avoid any buffer Underflows/Overflows.

GETH_AI.003 Header-Payload Split Function Does Not Support IPv6 Packets Received With Zero TCP Payload

The header-payload split function identifies the boundary between the TCP/IP header bytes and the payload of TCP/UDP and stores the header and payload data in separate buffers in the host memory.

However, when TCP/IPv6 packets are received with a IPv6 Payload Length that is equal to IPv6 header length (including extensions) plus TCP header size, the DWC_EQOS does not separate the boundary and forwards the complete packet to the Header buffer. This is because, the header-payload split function is triggered only when the IPv6 payload length field is greater than the IPv6 header length with extension fields, plus TCP header size (with at the least 1 byte of TCP payload).

Impacted Use Cases:

Received TCP/IPv6 packets that have only TCP header (and no TCP payload).

Consequence:

The dummy TCP payload and the Ethernet FCS bytes are stored in the Header buffer instead of getting stored in a separate payload buffer. Therefore, if the header buffer is mapped to a cache/faster memory, unnecessary dummy payload (if present) is stored in the cache.

There is no functional impact because the TCP stack does not forward the null-payload to the upper layer.

Workaround:

None required as there is no functional impact. TCP stack does not forward null payload to the upper layer.

GETH_AI.005 Application Error Along With Start-of-Packet Can Corrupt the Ongoing Transmission of MAC Generated Packets

On the MAC Transmit interface (MTI), if an error indication (mti_err_i) is asserted along with the Start of Packet (mti_sof_i) of a new packet while the MAC is transmitting an internally generated packet (ARP, PTO, Pause), the error indication aborts the ongoing transmission prematurely. This abort corrupts the MAC generated packet being transmitted. This defect manifests because the mti_err_i is inadvertently passed to the MAC transmitter logic directly when sampled along with mti_sof_i.

The scenarios that cause mti_sof_i and mti_err_i to be asserted together in non-Core configurations are:

1. DMA Configurations: Bus error on the first beat of frame data read from the application.
2. MTL Configurations: ati_error_i asserted along with ati_sof_i on the ATI interface.

Impacted Configurations:

Bus Error / ATI Error / MTI Error received from the system along with the first beat of packet data, manifesting as mti_sof_i and mti_err_i getting asserted together on MTI interface of MAC core, when a MAC generated packet is in transmission.

Consequence:

The MAC generated packet is sent on the line as a runt frame with corrupted FCS. The aborted packet is not retransmitted and can cause

a) Failure of intended flow control in case of PAUSE/PFC packet corruption

b) Delay in ARP Handshake from ARP Offload Engine; The ARP stack recovers because it sends ARP requests periodically

c) Delay in PTP Response/SYNC packets generated by PTP Offload Engine; The PTP stack recovers because it sends request packets periodically.

The probability of occurrence of an application error on the first beat of data and coinciding with a MAC generated packet transmission is very low.

Workaround:

No software workaround possible.

GETH_AI.006 Incorrect IP Header or Payload Checksum Status Given After MTL TX FIFO Flush

When Transmit Checksum Engine is enabled, it generates the IP Header error (IHE) and Payload checksum error (PCE) bits in the status given back to application after the packet is transmitted. These fields are written into a small FIFO when a packet is transferred from MTL to MAC. These bits are read from the FIFO and combined with the Tx status received from the MAC and given back to the application (ATI or DMA descriptor).

When a MTL Tx FIFO Queue flush is initiated for a different queue than the one from which the current packet is being transmitted, a dummy Tx status with flush indication is sent from MTL for the flushed packets. This can happen out of order with respect to the MAC status of the ongoing transmitted packet, when the queues are different.

However, due to this defect, the Checksum Engine status bits are incorrectly read out from the small FIFO along with the transfer of the dummy Tx status of the flushed queue and forwarded to the application. Later, when the MAC Tx status of the ongoing packet arrives, as the FIFO has already been read out, it returns zeros for checksum status fields instead of the actual values.

Impacted Use Cases:

When multiple transmit queues with Checksum offload engines are enabled, Drop Tx Status is not enabled (DTXSTS = 0 in MTL_Operation_Mode register) and a Flush TxQueue is given to a queue (FTQ = 1 in MTL_TxQ(#i)_Operation_Mode register) other than the one from which the ongoing packet transmission is taking place.

Consequence:

Incorrect checksum engine error status might be given for the packet under transmission (and/or the next one) at the time of Flush event in another queue.

Note: The checksum error status bits are normally used for debug purpose by the software driver. Therefore, there is no impact to normal operation.

Workaround:

None. CRC offload engine signals wrong debug status.

GETH_AI.007 IEEE 1588 Timestamp Interrupt Status Bits are Incorrectly Cleared on Write Access to the CSR Register with Similar Offset Address

When RCWE bit of MAC_CSR_SW_Ctrl register is set to 1, all interrupt status bits (events) are cleared only when the specific status bits are written with 1'b1.

However, due to the defect, the Status bits[9:0] of MAC_Timestamp_Status register at address 0x0b20 are unintentionally cleared when 1'b1 is written to the corresponding bit positions in any CSR register with address offset [7:0] = 8'h20.

The Status bits[9:0] correspond to the following events:

- Target time interrupt (TSTARGT[n] where n= 0, 1, 2, 3)
- Timestamp Seconds Register Overflow interrupt (TSSOVF)
- Target time programming error interrupt (TSTRGTERR[n] where n= 0, 1, 2, 3)

This defect is because, address bits[11:8] are not used in decoding the select signal that is used to clear the status bits, when 1'b1 is written to that bit.

Impacted Use Cases:

Software enables the write 1 to clear interrupt status bits, by setting RCWE = 1 in MAC_CSR_SW_Ctrl register.

Consequence:

When any of the Target Time Interrupts or Timestamp Seconds overflow events occur, the software might inadvertently clear the corresponding status bits (and the interrupt gets de-asserted), if it first writes to any CSR register at the shadow address (0x0_xx20 or 0x1_xx20). Consequently, the Interrupt Service Routine might not identify the source of these interrupt events, as the corresponding status bits are already cleared.

Note: The Timestamp Seconds Register Overflow event is extremely rare (once in ~137 years) and the Target Time Error interrupt can be avoided by appropriate programming. The frequency of Target Time reached interrupt events depends on the application usage.

Workaround:

When RCWE = 1 and Timestamp event interrupts are enabled, process and clear the MAC Timestamp Interrupt events first in the Interrupt Service Routine software, so that write operations to other shadow CSR registers are avoided.

GETH_AI.008 Application Error Along with Start-of-Packet Can Corrupt the FCS Field of the Previous Frame in the MAC Pipeline

On the MAC Transmit Interface (MTI) if an application error indication is asserted along with the Start of Packet of a new packet while the MAC is transmitting a packet, the error indication can corrupt the FCS field of the packet being transmitted. This defect manifests because the error indication is inadvertently passed to the MAC transmitter logic directly when sampled along with the Start of Packet indication.

The scenario that causes the problem is:

- Bus error on the first beat of frame data read from the application.

Impacted Use Cases:

This issue occurs when Bus Error is received from the system along with the first beat of new packet data, manifesting as error indication and Start of Packet indication asserted simultaneously during an ongoing packet transmission.

Consequence:

The packet in transmission is sent with corrupted FCS and therefore the remote end discards it.

Workaround:

Discard pending data on bus error and re-init the GETH.

GETH_AI.009 Corrupted Rx Descriptor Write Data

Packets received by DWC_ether_qos are transferred to the system memory address space as specified in the receive descriptor prepared by the software. After transferring the packet to the system memory, DWC_ether_qos updates the descriptor with the packet status.

However, due to a defect in the design, the Rx packet status gets corrupted when the MTL Rx FIFO status becomes empty during the packet status read. This can happen only when the MTL Rx FIFO is in Threshold (cut through) mode and Frame based arbitration is enabled on the receive.

Impacted Use Cases:

The defect is applicable when the Rx FIFO is in Threshold (Cut-through) mode and Frame based arbitration is enabled in the RxFIFO.

MTL Rx FIFO working in cut-through mode (bit[5], RSF in MTL_RxQ[n]_Operation_Mode register is set to 0, the default value) and

MTL Rx FIFO is enabled to work in Frame Based Arbitration (bit[3], RXQ_FRM_ARBIT in MTL_RxQn_Control register is set to 1.

Consequence:

The Rx packet status written into the descriptor for the affected packet is corrupt. All subsequent frames are processed as expected.

Workaround:

Do not use cut through OR/AND do not use RX arbitration.

GETH_AI.010 Fatal Bus Error Interrupt Might Be Generated for Incorrect DMA Channel

When a bus error occurs, the status reflects the associated RX DMA channel number.

When the current burst or packet transfer is about to end, the MTL arbiter might grant access to another Rx DMA channel for the next burst or packet transfer (with ari_chnum signal indicating the channel number of Rx DMA that is granted latest access).

However due this defect, when bus error occurs towards end of current burst, the DMA might associate it with Rx DMA channel of next burst (based on the ari_chnum) and provide the incorrect Rx DMA channel number in the status register.

Impacted Use Cases:

Cases where the MTL arbiter has already granted access to another Rx DMA channel for next burst transfer and bus error occurs for current burst.

Consequence:

A wrong Rx DMA channel number is reported for the Fatal Bus Error interrupt.

Workaround:

Discard pending data on bus error and re-init the GETH. Debugger can not rely on DMA Status register after bus error of a RX Burst.

GETH_AI.011 Receive Queue Overflow at End of Frame Along with SPRAM Read-Write Conflict Can Cause Data Loss

Read and write operations can conflict, based on the address being read and written. During a conflict in the MTL Receive FIFO, the read operation gets priority and the write operation is retried in the subsequent cycle.

When End of Frame (EoF) is received, the MTL Receiver computes FIFO overflow condition based on the anticipated space needed to write End of Frame (EoF) and RxStatus. When EoF is received on MRI interface and a read-write conflict occurs in the SPRAM for the EoF write along with a FIFO overflow computation, it causes the MTL Receive FSM to malfunction.

Impacted Use Cases:

This issue occurs when the MTL Receive FIFO has a read-write conflict and the Rx FIFO computes an overflow condition upon receiving EoF in the MRI interface.

Consequence:

The packet that causes MTL FIFO overflow is handled correctly. However due to the malfunctioning of MTL receive FSM, the subsequent packet loses a part of the data at the beginning of the frame.

Workaround:

Discard pending data on bus error and re-init the GETH.

GETH_AI.012 Incorrect Flexible PPS Output Interval When Fine Time Correction Method is Used

The MAC provides programmable option, fine and coarse, for correcting the IEEE 1588 internal time reference.

When coarse correction method is used, the correction is applied in one shot and does not affect the flexible PPS output.

When fine correction method is used, the correction is applied uniformly and continuously to the IEEE 1588 internal time reference as well as to the flexible PPS output.

However, due to this defect, when fine correction method is used and the drift in the frequency of the clock that drives the IEEE 1588 internal time reference is large (when compared with the grandmaster source clock), the flexible PPS output interval is incorrect. This does not impact the IEEE 1588 internal time reference updates.

The internal PPS counter used for generating the PPS interval is incorrectly reset earlier than expected, resulting in the next PPS cycle starting incorrectly, earlier than expected.

Impacted Use Cases:

The Flexible PPS Output feature is used in Pulse Train mode and the Fine Correction method is used for correcting the IEEE 1588 internal time reference due to drift in the frequency of the clock that drives it.

Consequence:

The incorrect Flexible PPS Output Interval from the MAC can cause the external devices, that are synchronized with flexible PPS trigger outputs, to go out of synchronization.

Workaround:

The application can use coarse method for correcting the IEEE 1588 internal time reference. Because, in the coarse correction method, as the time correction is applied in a single shot, timestamp captured for at the most one packet is impacted. This might be the case when current cycle of time-synchronization related packet-exchanges coincides with the coarse time correction of previous cycle. This discrepancy is corrected in the next time-synchronization correction cycle.

GETH_AI.013 False Dribble and CRC Error Reported in RMII PHY 10Mbps Mode

The MAC receiver clock is derived synchronously from RMII REF_CLK, the frequency is 2.5MHz in 10Mbps speed mode and 25MHz in 100Mbps speed mode. In 10 Mbps mode, the 2-bit RMII data is captured every 10 cycles of RMII REF_CLK, combined and provided as 4-bit data on the MAC receiver clock. As per RMII protocol, the RMII CRS_DV is asserted asynchronously with RMII REF_CLK, which also implies that it is asynchronous to the MAC receiver clock. The MAC correctly captures the received packet irrespective of the phase relation between RMII CRS_DV assertion and MAC receiver clock.

However due to this defect, in the 10Mbps speed mode, when the RMII CRS_DV is asserted two RMII REF_CLK rising edges ahead of MAC receiver clock, the MAC reports false dribble and CRC error in the Receive status. The dribble error is reported when MAC receives odd number of nibbles (4-bit words) and CRC error is additionally reported. In this case the additional nibble captured is a repetition of the last valid nibble. The MAC forwards only the data received on byte boundaries to the software and ignores the extra nibble. Therefore, there is no data loss or corruption of packet forwarded to the software. However, if error-packet drop is enabled (FEP bit in MTL_RxQ(#i)_Operation_Mode register is set to 0), MAC drops the packets, causing packet loss and impacts the performance.

Impacted Use Cases:

The RMII PHY interface is enabled for 10Mbps operation and RMII CRS_DV is asserted two RMII REF_CLK rising edges ahead of MAC receiver clock.

Consequence:

The MAC reports false dribble and CRC error in Receive status. If error-packet drop is enabled, (FEP bit in MTL_RxQ(#i)_Operation_Mode register is set to 0), MAC drops the packets causing packet loss and impacts the performance. If the error-packet drop is disabled (FEP bit in MTL_RxQ(#i)_Operation_Mode register is set to 1), MAC forwards the packet to the software, up to the byte boundary, and there is no data loss or corruption.

Workaround:

If error-packet drop is enabled (FEP bit in MTL_RxQ(#i)_Operation_Mode register is set to 0), software can disable it and take the dropping decision based on the Rx status. If the dropping of error packets is disabled (FEP bit in MTL_RxQ(#i)_Operation_Mode register is set to 1), software can ignore the dribble and CRC error and accept packets that have both these errors together. The occurrence of real dribble error is rare and happens when there are synchronization issues due to faulty clock recovery.

GTM_AI.254 TIM TDU: TDU_STOP=b101 not functional

Stop counting of register TO_CNT on an tdu_word_event or stop counting of TO_CNT1 on a tdu_frame_event is not possible.

Scope

TIM

Effects

TO_CNT1, TO_CNT can not be stopped counting.

Workaround

No workaround available.

GTM_AI.262 DPLL: PSSC behavior in mode change (DPLL_CTRL_0.RMO = 0 ->1)

When changing from normal mode to emergency mode (DPLL_CTRL_0.RMO = 0 ->1) the RAM1b.PSSC value is not calculated like described in the specification.

The PSSC value is not updated at the following trigger slope but at the following STATE slope with the value PSSC=PSTC.

Scope

DPLL

Effects

When changing from normal mode to emergency mode (DPLL_CTRL_0.RMO = 0- >1) the RAM1b.PSSC value is not calculated like described in the specification.

The PSSC value is not updated at the following trigger slope but at the following STATE slope with the value PSSC=PSTC.

The specification showed that the sentence “For changing from normal mode to emergency mode at the following TRIGGER slope” is not helpful because one must expect when changing the RMO bit that there are problems with the next trigger slope so that implementing a modification like described in the specification seems not the right thing to do.

Specification will be modified towards observed behavior.

Workaround

If possible leave PSSC as is.

If a different value for PSSC is necessary the value could be written by CPU interface. Starting with version 3.1.5 the modification could be done by MCS0 as well.

GTM_AI.263 DPLL: DPLL_STATUS.LOCK1 flag (0 ->1) delayed after direction change when DPLL operating in DPLL_CTRL_0.RMO = 1

The DPLL_STATUS.LOCK1 flag does not behave like requested in the specification:

When the DPLL is operating in emergency mode (DPLL_CTRL_0.RMO = 1) and a direction change happens the lock1 flag is reset to “0” as described in the specification.

The problem is, that the LOCK1 bit is set to “1” again after 4 detected gaps (missing state irq, ms -flag) and not as requested after 2 subsequent gaps.

Scope

DPLL

Effects

When the DPLL is operating in emergency mode (DPLL_CTRL_0.RMO = 1) and a direction change happens the LOCK1 flag is reset to “0” as described in the specification.

The problem is, that the LOCK1 bit is set to “1” again after 4 detected gaps (missing state irq, ms -flag) and not as requested after 2 subsequent gaps.

Workaround

If you need to use this information one could observe the missing state interrupt to check for the correct point in time when the LOCK1 flag should be set again.

If the use of the LOCK1 flag is not time critical it could be used as is with the latency described above.

GTM_AI.304 MCS: Scheduling modes Single Prioritization and Multiple Prioritization are not functional

If an MCS instance is configured with the Single or Multiple Prioritization Scheduling mode and the last non-suspended and prioritized MCS channel (CLP) is entering its suspended state (which means that the MCS starts scheduling the remaining non-prioritized channels with accelerated scheduling scheme) and if the suspended state of CLP is resumed five clock cycles after it was entering the suspended state the MCS channel CLP is not executing the instruction that is following the suspending instruction.

Scope

MCS

Effects

The program execution of a prioritized MCS channel can skip an instruction that is directly following a suspending instruction.

Workaround

Add an additional NOP instruction after all suspending instructions (WURM, WURMX, WURCX, WUCE, ARD, ARDI, NARD, NARDI, AWR, AWRI, BRD, BRDI, BWR, and BWRI) in a prioritized MCS program.

<u>GTM_AI.305</u> TIM Signal Generation with serial shift mode TSSM: If TSSM_OUT is used in channel x and channel x+1 uses edges of FOUT_PREV, these edges show an unexpected delay which lead to a delayed operation of channel measurement or TDU functionality of channel x+1

If channel x operates in TSSM mode and signal generation is active with (CNTS[21:20]!=b00) and channel x+1 uses the signal edges from FOUT_PREV(x+1)

- a) for channel measurements with TIM[i]_CH[x+1]_ECTRL.USE_PREV_CH_IN=1
  or
- b) inside the timeout detection unit with TIM[i]_CH[x+1]_ECTRL.USE_PREV_TDU_IN=1

The actions in TDU or channel measurement triggered by edges on FOUT_PREV(x+1) will occur with unexpected delay. (Delay correlates to shift clock in channel x).

Scope

TIM TSSM mode

Effects

Channel measurements or TDU functionality in channel x+1 triggered with unexpected delay.

Workaround

Using the LUT in the filter unit in channel x+1 ensures that the signal edge information of FOUT_PREV(x+1) is reconstructed properly on F_OUT[x+1].

Use the following settings:

TIM[i]_CH[x+1]_ECTRL.USE_LUT=b10;

TIM[i]_CH[x+1]_TDUC.TO_CNT2=0xF0;

TIM[i]_CH[x+1]_ECTRL.USE_PREV_CH_IN=0;

TIM[i]_CH[x+1]_ECTRL.USE_PREV_TDU_IN=0.

GTM_AI.306 DPLL: DPLL_NUTC.syn_t_old, DPLL_NUSC.syn_s_old not updated according specification

The DPLL specification defines for DPLL_NUTC.WSYN=1 that an update of register DPLL_NUTC allows writing of the bits DPLL_NUTC.syn_t while DPLL_NUTC.syn_t_old inherits the previous value of DPLL_NUTC_syn_t.

Differing from the specified behavior the actual hardware does not update the value of DPLL_NUTC.syn_t_old with the previous value of DPLL_NUTC.syn_t but instead updates DPLL_NUTC.syn_t_old according to the corresponding bits of the write operation executed by the CPU.

The DPLL specification defines for DPLL_NUTC.WSYN=1 that an update of register DPLL_NUSC allows writing of the bits DPLL_NUSC.syn_s while DPLL_NUSC.syn_s_old inherits the previous value of DPLL_NUSC_syn_s.

Differing from the specified behavior the actual hardware does not update the value of DPLL_NUSC.syn_s_old with the previous value of DPLL_NUSC.syn_s but instead updates DPLL_NUSC.syn_s_old according to the corresponding bits of the write operation executed by the CPU.

Scope

DPLL

Effects

The registers bits DPLL_NUTC.syn_t_old are not updated with the previous value of DPLL_NUTC_syn_t but by the bits of the input data word.

The registers bits DPLL_NUSC.syn_s_old are not updated with the previous value of DPLL_NUSC_syn_s but by the bits of the input data word.

Workaround

If the update of syn_t/s_old shall be done like described in the specification the register DPLL_NU(T/S)C.syn_t/s must be read first, then the DPLL_NU(T/S)C.syn_(t/s) can be used to modify the bits which are written to DPLL_NU(T/S)C.syn_(t/s)_old.

As the current behavior of DPLL_NUT/SC.syn_s/t_old is in use by and can be advantageous for certain applications, there is no intend to change the current hardware behavior at this point in time. Instead a specification update to align the specification with the current hardware behavior is planned for future GTM generations.

GTM_AI.307 IRQ: AEI_IM_ADDR is not set in GTM_IRQ_NOTIFY register if cluster 0 is disabled

Bit 2 - AEI_IM_ADDR - in register GTM_IRQ_NOTIFY is not set and the related error interrupt (if enabled) does not occur if cluster 0 is disabled and

- an FPI read or write access to a cluster 0 register
  OR
- an illegal FPI write access to any enabled cluster is done.

In case BRIDGE_MODE.MSK_WR_RSP = 0x0 (not recommended) an FPI bus error is issued for all write accesses.

Scope

IRQ

Effects

See description above.

Workaround

- a) Do not disable cluster 0.
- b) If cluster 0 is disabled: after each WRITE access check register GTM_AEI_STA_XPT; if the read value is >0, it signs an access error.
- c) If cluster 0 is disabled: do not read from cluster 0 register or any other disabled cluster register.

GTM_AI.308 TIM, ARU: Limitation that back-to-back TIM data transfers at full ARU clock rate cannot be transferred correctly with ARU dynamic routing feature

If TIM input signals with signal changes faster or equal than ARU clock rate are processed with the TIM and the results are routed via ARU in dynamic routing mode, it is likely that there is a data loss and only each second data can be transferred.

Scope

ARU Routing, DEBUG signal interface

Effects

- a) If the ARU CADDR is kept stable and data is transferred back-to-back for 2 or more consecutive aru clock cycles while operating in ARU dynamic routing mode, then every second data provided by the TIM module gets lost.
- b) Debugging of an ARU data transfer not completely correct. Every second GTM_DBG_ARU_DATAi_val signal missing.

Workaround

Do not use the dynamic routing feature of ARU in the manner that the same ARU caddr is served for multiple cycles with back-to-back data transfers.

Ensure that every ARU clock cycle the CADDR address will change.

GTM_AI.309 TIM Signal Generation with serial shift mode TSSM in channel x: Generated TSSM_OUT signal used in lookup table of inputsrc module of channel x has unpredictable delay

If channel x operates in TSSM mode and signal generation is active with (CNTS[21:20]!=b00) and channel x uses the signal TSSM_OUT(x) in the lookup table with USE_LUT(x)=0b11.

Results of lookuptable function will behave unexpected due to delayed input of TSSM_OUT(x). (Delay correlates to shift clock in channel x)

Scope

TIM TSSM mode

Effects

Lookup table in TIM channel inputsrc module shows unexpected results.

Workaround

Use lookup table of inputsrc module channel x+1. The TSSM_OUT signal of channel x which is routed via FOUT_NEXT(x) to channel x+1 can be used with USE_LUT(x+1)=0b10.

GTM_AI.318 MCS: NARD(I) instruction terminates unexpectedly

If the bit field CFG_CLK_RATE of register CCM[i]_HW_CONF is set and an MCS runs on a cluster i while bit field CLSc_CLK_DIV of register GTM_CLS_CLK_CFG is set to 1, the execution of a NARD or NARDI instruction might signalize an unsuccessful data transfer (bit field SAT of internal MCS register STA is cleared) although the data source has data available for sending. The unexpected behavior depends on the current state of the internal ARU counter.

Scope

MCS

Effects

The available data from the ARU source is not sent via ARU.

Workaround

None. However, typical applications that are polling data sources with the NARD or NARDI instruction do not have to concern about this errata since the polling loop just requires more iterations.

GTM_AI.319 (A)TOM: Unexpected (A)TOM_CCU1TCx_IRQ in up/down counter mode

If the up-down counter mode is activated (bit field UDMODE of register (A)TOM[i]_CH[x]_CTRL is set to a value greater than zero) and the interrupt (A)TOM_CCU1TCx_IRQ is enabled (bit field CCU1TC_IRQ_EN of register (A)TOM[i]_CH[x]_IRQ_EN is set), the interrupt signal (A)TOM_CCU1TCx_IRQ will be set unexpectedly directly after the interrupt (A)TOM_CCU0TCx_IRQ was set and indicates that the counter (A)TOM[i]_CH[x]_CN0 reaches (A)TOM[i]_CH[x]_CM0.

Scope

TOM/ATOM

Effects

Interrupt signal (A)TOM_CCU1TCx_IRQ is set unexpectedly.

Workaround

If the interrupt (A)TOM_CCU1TCx_IRQ is needed, it can be disabled by the first occurrence of itself and enabled again with the interrupt (A)TOM_CCU0TCx_IRQ. With the following occurrence of the interrupt (A)TOM_CCU1TCx_IRQ it will be disabled again and so on.

GTM_AI.320 ATOM: Unexpected restart of a SOMS oneshot cycle while ATOM[i]_CH[x]_CM0 is zero

If ATOM is set to SOMS oneshot mode (bit field MODE of ATOM[i]_CH[x]_CTRL is set to 0b11 and bit field OSM in register ATOM[i]_CH[x]_CTRL is set) a oneshot cycle is started immediately by writing a value unequal to zero to ATOM[i]_CH[x]_SR0 register while the value of ATOM[i]_CH[x]_CM0 register is zero.

Scope

ATOM

Effects

Restarting of a oneshot cycle starts immediately while ATOM[i]_CH[x]_CM0 is zero and a write access to ATOM[i]_CH[x]_SR0 is executed with a value unequal to zero.

Workaround

Avoid value 0 in ATOM[i]_CH[x]_CM0 register if SOMS oneshot mode is enabled (bit field OSM in register ATOM[i]_CH[x]_CTRL).

GTM_AI.322 DPLL: PSTC, PSSC not updated correctly after fast pulse correction completed (DPLL_CTRL1.PCM1/2 = 0)

When additional pulses are requested using DPLL_CTRL_11.PCMF1/2=1 AND PCMF1/2_INCCNT_B=0 the PSTC/PSSC parameters as well as NMB_T/S_TAR are not updated correctly, because either the amount of additional pulses (MPVAL1/2) are not incremented or NMB_T/S_TAR is set to a wrong value.

Scope

DPLL

Effects

After the pulse correction is performed the fields NMB_T/S_TAR are set to wrong values such that after a new input event the parameters PSTC/PSSC are not updated correctly.

Incorrect PSTC/PSSC values are ending up in wrong NA[i] parameters. These wrong NA[i] values are leading to incorrect PMT calculations.

The pulse generation itself (register DPLL_INC_CNT1/2 and the status of the angle clocks TBU_TS1/2) is correct and not affected by this issue.

Workaround

Implement the workaround in the TISI interrupt, i.e. start the workaround at the arrival of inactive edge. This ensures that swon_t/swon_s is stable and the incorrect nmb_t_tar/nmb_s_tar has already been generated. This is to ensure the following:

- Start the workaround after the incorrect nmb_t_tar/nmb_s_tar has been generated and swon_t/swon_s is not toggling anymore
- Workaround should be finished before the arrival of next active edge.

Workaround steps are as follows:

1. Check swon_t/swon_s,
   - If swon_t/swon_s = 1, save & use nmb_t_tar/nmb_s_tar for further corrections
   - Else save & use nmb_t_tar_old/nmb_s_tar_old for further corrections.
2. Set PCM1=1 to trigger the fast pulse correction with PCMF1 already set to 1
3. Wait for PCM1 to reset to 0
4. Overwrite the nmb_t_tar/nmb_s_tar or nmb_t_tar_old/nmb_s_tar_old with the correct value based on swon_t/swon_s, similarly based on the choice in step 1.

After the next active edge, the PSTC/PSSC values are corrected.

GTM_AI.323 DPLL: Registers DPLL_NUTC.SYN_T and DPLL_NUSC.SYN_S are updated by the profile (ADT_T.NT/ADT_S.NS) before the DPLL is synchronized (DPLL_STATUS.SYT/S=0)

The registers DPLL_NUTC.SYN_T and DPLL_NUSC.SYN_S as well as the corresponding *_OLD registers are updated unexpectedly by the profile (ADT_T.NT/ADT_S.NS) before the DPLL is synchronized (DPLL_STATUS.SYT/S=0).

This is not a problem for the calculation of the number of pulses (nmb_t/s,..), due to the fact that the correct value of SYN_T/S for the internal use is determined by the signal DPLL_STATUS.SYT/S. The microtick generation of the DPLL is not affected by this bug.

This problem is only relevant if the SYN_T/S values are read from other consumers than the DPLL.

Scope

DPLL

Effects

When the DPLL is enabled and before the DPLL is synchronized (by writing to the relevant pointers (DPLL_APT_2c/DPLL_APS_1C3) the DPLL_NUTC.SYN_T/DPLL_NUSC.SYN_S registers are unexpectedly updated by the profile.

Because the SYN_T_OLD and SYN_S_OLD registers are updated by SYN_T, SYN_S they are affected as well.

The DPLL internal processes of calculation of the number of microticks for the next increment is not affected by that bug.

Workaround

When DPLL_NUTC.SYN_T/_OLD, DPLL_NUSC_SYN_S/_OLD values are needed outside the DPLL it must be checked that the DPLL is already synchronized (DPLL_STATUS.SYT/SYS). When the relevant DPLL channel (TRIGGER/STATE) is not synchronized yet the SYN_T/S values should be taken into account as "1".

GTM_AI.325 TIM: Bits ACB[2:1] lost on interface to ARU (always zero)

In case of CFG_CLOCK_RATE=1 (see register CCM[i]_HW_CONF) some cluster can be configured to run with fast clock frequency (200 MHz) while the ARU always runs with slow clock frequency (100 MHz).

For this configuration of a cluster running with fast clock frequency (200 MHz) also the TIM module in this cluster is running with fast clock frequency (200 MHz).

In this case and if a TIM channel is configured to send data to ARU (ARU_EN bit in TIM[i]_CH[x]_CTRL register), the bits ACB[2:1] will not be transferred with any ARU transfer request, they are always 0.

Due to this any master receiving ARU data is not able to use the ACB bit information received from a fast running TIM.

- ACB[1]: additional ARU request event received while actual request not finished
- ACB[2]:Timeout event occurred

Scope

TIM, ARU transfers

Effects

ARU bits ACB[2:1] sent by TIM are always zero.

Workaround 1

For applications running within a single cluster, use AEI communication (MCS-TIM) instead of ARU communication.

Workaround 2

Use half clock rate for the cluster that contains the TIM module read out via ARU.

Workaround 3

If the data from TIM channel should be routed over the ARU to the MCS module, then the MCS can read the information (TIM[i]_CH[x]_IRQ_NOTIFY[4:3]) directly over the AEI bus master interface instead of routing it through the ARU.

Hint: For this workaround the TIM channel and the MCS channel have to be in the same cluster.

Workaround 4

If the data from TIM channel should be routed over the ARU to the FIFO or BRC module, then the MCS can read the information (TIM[i]_CH[x]_IRQ_NOTIFY[4:3]) directly over the AEI bus master interface and forward the data via its ARU interface to FIFO or BRC.

Hint: For this workaround the TIM channel and the MCS channel have to be in the same cluster.

Workaround 5

Depending on the application it might be possible by comparing the actual ARU data with the previous received ARU data to “reconstruct“ the ACB bits [2:1].

GTM_AI.326 TIM: ARU bit ACB[0] (signal level) incorrect in case a second ARU request occurs while the actual request is just acknowledged

An issued ARU request will be served at least after the ARU round trip time.

If one GTM clock cycle before the ARU request is acknowledged a new capture event occurs (overflow condition due to e.g. input change) the bit ACB[0] will not show the new value.

The overflow bit ACB[1] and the ARU data words selected by (E)GPRy_SEL) will show the correct behavior, only the ACB[0] will show the previous state.

Scope

TIM, ARU transfers

Effects

ARU bit ACB[0] not consistent with data transferred in ARU data words.

Workaround 1

Ensure that events which trigger a ARU request occur with a greater timely distance than the ARU round trip time.

Workaround 2

Use the signal level information embedded in the ARU data words (selectable by ECNT/TIM_INP_VAL).

This data will show the correct signal level.

GTM_AI.329 Interference of MCS to AEI/ADC and CPU to AEI traffic within the same cluster could result in incorrect MCS program execution

Operations of the MCS via its AEI master port on the AEI bus can be categorized into 3 different types of operations based on the response time required by an addressed resource to complete the operation on the bus. As operations from MCS to ADC are also handled via the MCS AEI master port, ADC operations are also relevant regarding the bus traffic scenarios.

The vast majority of register accesses via AEI as well as ADC reads complete with zero wait states (N=0) on the AEI bus and fall into the first category. The second category is defined by register operations to a small set of special registers that require 1 wait cycle (N=1) on the AEI interface to complete while the third category covers AEI accesses to memories (e.g. DPLL memory, MCS memory or FIFO memory) as well as 2 special registers in MCS that require multiple wait cycles (N>1) on the AEI interface to complete.

Certain interferences between accesses from MCS to the AEI/ADC interface and AEI accesses from CPU within the same cluster can result in bus traffic situations that impact the correct program execution of MCS channels. These rare but critical traffic conditions must be avoided to ensure the correct execution of MCS code.

Further the dynamic usage of GTM_MCS_AEM_DIS to temporarily disable a MCS AEI master port (AEI and ADC communication path) must be avoided. This switch can only be used for the permanent disablement of the MCS AEI master port.

MCS AEI master port usage scenarios proven to avoid the problematic traffic conditions under all circumstances include the usage scenarios described in the workaround section of this errata. Usage of MCS to AEI or ADC communication not covered by tested scenarios must be avoided. Communication from MCS to any GTM resource via ARU is not impacted and has no influence on the problems and scenarios described within this errata.

- GTM resources with 1 AEI wait cycle (N=1):
  - GTM_RST GTM_CLS_CLK_CFG
  - BRC_RST
  - TIM[i]_RST TOM[i]_TGC0_GLB_CTRL
  - TOM[i]_TGC1_GLB_CTRL
  - DPLL_CTRL_1
  - ATOM[i]_AGC_GLB_CTRL
- GTM resources with more than 1 AEI wait cycle (N>1):
  - AFD[i]_[0-7]_BUFF_ACC
  - FIF0[i]_MEMORY RAM address space*
  - DPLL_RAM1A RAM address space
  - DPLL_RAM1B RAM address space
  - DPLL_RAM1C RAM address space

- DPLL_RAM2 RAM address space
- MCS[i]_MEMORY RAM address space*
- MCS[i]_CTRG MCS[i]_STRG

* Not accessible from MCS

Scope

Usage of MCS AEI master port (AEI and ADC communication from MCS); MCS channel code execution; Dynamic usage of GTM_MCS_AEM_DIS.

Effects

Incorrect MCS channel code execution (skipping execution of instructions or repetitive execution of instructions) or processing of incorrect read data from AEI or ADC interface by MCS channel code.

Workaround

To ensure that a correct execution of MCS channel code is not influenced by certain traffic scenarios on the MCS AEI/ADC bus master interface, only proven usage scenarios are allowed for MCS to AEI/ADC communication. The most common usage scenarios tested to be safe include:

Option 1:

Limit the usage of the MCS AEI master port (ADC and AEI communication) to one MCS channel per MCS at a time. ARU communication is available for all MCS channels and there are no limitations for the CPU access path in this usage model.

In case multiple MCS channels want to use the AEI master port for AEI or ADC communication, establish a mechanism that ensures that only one channel uses the AEI master at a time (e.g. exchange a token between channels or use trigger registers to hand over the AEI master port ownership between MCS channels).

Option 2:

Limit the usage of the MCS AEI master port to ADC communication only. The usage of the MCS AEI master port for AEI communication must be avoided for all channels. ARU communication is available for all MCS channels and there are no limitations for the CPU access path in this usage model.

Option 3:

Limit the usage of the MCS AEI master port to ADC as well as AEI communication with zero wait cycles (N=0) only. AEI communication from MCS to resources with N>0 must be avoided.

Further the access from CPU to this cluster has to be limited to accesses with zero or one wait cycle (N=0 and N=1) only. Memories or registers with N>1 within the given clusters can not be accessed by the CPU in this usage model.

If the CPU has to access these resources in this cluster, the number of MCS threads using the MCS AEI master port to access AEI or ADC temporarily has to be limited to one thread while all other MCS threads accessing AEI or ADC have to be suspended while the CPU accesses N>1 resources.

GTM_AI.331 GTM_TIM[i]_AUX_IN_SRC and GTM_EXT_CAP_EN_[i] register: wrong status 2 by AEI write access if cluster 0 is disabled

AEI write access to the register GTM_TIM[i]_AUX_IN_SRC and GTM_EXT_CAP_EN_[i] via the legacy address space responds with status 2 even though the write operation is correctly executed and the register contains the correct new value.

Scope

Register access via legacy address.

Effects

If status 2 is responded an interrupt bit in GTM_IRQ_NOTIFY is set and the write address will be caught in register GTM_AEI_STA_XPT.

Any further AEI access with responded status >0 will not be caught in GTM_AEI_STA_XPT until GTM_AEI_STA_XPT is reset by AEI read to GTM_AEI_STA_XPT.

Workaround

Do not use the legacy addresses of the listed registers while cluster 0 is disabled.

GTM_AI.332 Access to registers GTM_TIM[i]_AUX_IN_SRC and GTM_EXT_CAP_EN_[i] via legacy address space: read data always 0 for AEI read access while cluster 0 is disabled

AEI read access to registers via legacy address space which are not in any cluster will respond always with read value 0 if cluster 0 is disabled.

- Impacted registers are:
  - GTM_TIM[i]_AUX_IN_SRC
  - GTM_EXT_CAP_EN_[i]

Scope

Register access via legacy address.

Effects

If cluster 0 is disabled, register data unequal to 0 would not be read from any register which is not part of a cluster (see list of impacted register sections) via its legacy address.

Workaround

Do not access the impacted registers via their legacy address while cluster 0 is disabled.

GTM_AI.333 MCS bus master interface: a not word aligned address access to DPLL ram region can cause incorrect execution of MCS channel code

MCS accesses to the DPLL ram regions with not correctly aligned address while concurrently CPU accesses to the same cluster occur could result in incorrect execution of MCS channel code.

Scope

MCS bus interface; MCS program execution.

Effects

MCS channel program execution incorrect. Instructions might be executed multiple times or might be skipped. MCS BRD* instruction reads wrong data.

Workaround

Ensure that address used in BWR* /BRD* instructions is correctly aligned.

Note: If the bus master addresses as provided in table "MCS Master Interface Address Map" are used along with BWR /BRD* then this issue will not occur.*

GTM_AI.334 DPLL RAM content of single address can be corrupted after leaving debug mode

Assume a MCS RAM write access to DPLL RAM address x in RAM1a, RAM1bc or RAM2 is executed at the point in time when the GTM is switched to debug mode (gtm_halt_req=1). Any following write access to DPLL address space while in debug mode will corrupt the data in memory location x when the restore operation which is executed while leaving debug mode (gtm_halt_req=0) is processed.

Read operations to DPLL address space while in debug mode will not corrupt the DPLL memory content.

Scope

GTM Debug

Effects

Data in RAM might be corrupted

Workaround

If only READ accesses to DPLL address space are performed while in debug mode the described effect will never occur.

When write accesses to DPLL address space are performed while in debug mode the following workaround has to be considered:

1. Determine with the debugger whether a BWR instruction to DPLL RAM was executed just before the HALT occurred.
2. The active address and the data of this instruction has to be written again with a debug access directly before leaving debug mode.

<u>GTM_TC.018</u> DPLLRAM trace data can be wrong

Note: This problem only has an effect during debugging.

The OCDS Trigger Bus Interface supports routing of various trigger sets to MCDS on OTGBM0 and OTGBM1 (see table “Trigger Set Mapping Options” in the GTM chapter).

Tracing of addresses or data of a DPLL RAM on one part of the OTGBM interface (OTGBM0 or OTGBM1, respectively) can create wrong DPLLRAM trace data when any other source (including data or addresses of a different DPLL RAM) is configured for the other part of the OTBGM interface (OTGBM1 or OTGBM0, respectively).

Workaround

- If DPLLRAM addresses are configured for OTGBM0 (bit field OTSS.OTGBM0 = 2 or 3 or 4):
  – only DPLLRAM data of the same DPLL RAM may be selected for OTGBM1 (i.e. OTSS.OTGBM1 must be equal to OTSS.OTGBM0);
  – otherwise “No Trigger Set” must be selected for OTGBM1 (OTSS.OTGBM1 = 0).
- If DPLLRAM data are configured for OTGBM1 (bit field OTSS.OTGBM1 = 2 or 3 or 4):
  – only DPLLRAM addresses of the same DPLL RAM may be selected for OTGBM0 (i.e. OTSS.OTGBM1 must be equal to OTSS.OTGBM0);

- otherwise “No Trigger Set” must be selected for OTGBM0 (OTSS.OTGBM0 = 0).

GTM_TC.019 ARU can not be traced if GTM cluster 5 is disabled

Note: This problem only has an effect during debugging.

Tracing of the ARU is controlled by the GTM cluster 5 clock. If cluster 5 is disabled, the ARU can not be traced anymore.

Workaround

Do not disable GTM cluster 5 if ARU shall be traced.

GTM_TC.020 Debug/Normal read access control via bit field ODA.DRAC

A few GTM registers have a different read behavior when accessing them with debug read accesses (see section “GTM Software Debugger Support” in the GTM chapter of the User’s Manual for further details).

Depending on the reading master and the configuration of bit field DRAC in register GTM_ODA (OCDS Debug Access Register), the read can be performed in a specific way for debug related read operation.

According to the User’s Manual the read is performed as a debug read operation

- for all masters when ODA.DRAC = 10_B or 11_B,
- for the Cerberus (OCDS) FPI master when ODA.DRAC = 00_B

Problem Description

In the current implementation the read is performed as debug read operation

- for all masters when ODA.DRAC = 10 or 11_B,
- for the CPU2 FPI master when ODA.DRAC = 00_B

Workaround

The problem described above has 2 aspects:

1. For CPU2 Access to GTM

When the CPU2 FPI master is used to perform a normal read of the GTM registers mentioned above, setting ODA.DRAC = 01_B is required to avoid an unintended debug read access that would be caused by this issue.

2. For Cerberus (OCDS) Access to GTM

When ODA.DRAC = 00_B, due to this problem any read access of the Cerberus (OCDS) FPI master to the registers that by default have a different behavior between normal and debug read will cause the normal read behavior. To get the intended debug read behavior, ODA.DRAC needs to be set to 10_B or 11_B before each access of the Cerberus and set back to 00_B afterwards to not affect the access of other FPI masters on the registers mentioned above.

HSCT_TC.012 HSCT sleep mode not supported

Due to unreliability of the wake-up functionality, sleep mode for the HSCT is no longer supported and shall not be used.

Do not set bit SLEEPCTRL.SLPEN = 1_B.

HSCT_TC.013 Internal Loopback Mode not reliable

The internal loopback mode used for looping the HSCT TX data back internally to HSCT RX is not reliable to work under all operating conditions.

Therefore, do not use the internal loopback mode (i.e. do not set bit TESTCTRL. LLOPTXRX = 1_B).

Workaround

Use external loopback by configuring another external device (slave) by sending an interface control command with payload value = 0xFF (Turn on payload loopback) from the master interface.

MCDS_TC.052 TriCore wrap around write access causes redundant MCDS message

When the TriCore performs a wrap around write access, MCDS generates a redundant message.

Example:

TriCore writes 0x87654321 to address 0xAF01F90E, but due to wrap around it first writes 0x4321 to 0xAF01F90E and then writes 0x8765 to 0xAF01F900.MCDS outputs the following messages:

- DTU_<AorB>_TCX.DTW 0x8765 0xAF01F900
  - i.e. writing HWORD(8765) to AF01F900
- DTU_<AorB>_TCX.DTW 0x87654321 0xAF01F90E
  - i.e. writing WORD(87654321) to AF01F90E

Workaround

No workaround possible.

MCDS_TC.064 ACCEN0 register write not supervisor protected

The write access terms for register ACCEN0 are defined as “SV, SE” (access only when Supervisor Mode/Safety Endinit is active) in table “Registers Overview” in the MCDS and MCDSLight chapter of the TC3XX_ED documentation.

Actually, the implementation of register ACCEN0 in the MCDS and MCDSLight modules does not have Supervisor Mode protection (“SV”), it only has Safety Endinit protection (“SE”).

MCDS_TC.065 Selection of SRI trace sources

The MCDS/MCDSlight provides the capability to trace accesses to the SRI Slaves CPUx_MEMSlave, LMU0, OLDA, to trace accesses initiated by the SRI Master DMA, and to trace SPU output events.

The signal timing at these interfaces is compliant to the Infineon internal SRI bus protocol. The bus protocol consists of an address and data phase. It is not possible to select one of the above trace sources while transactions are ongoing. This can lead to permanently corrupted MCDS trace messages.

Workaround

Switch to the trace source only in a time frame when no transactions are ongoing.

MCMCAN_AI.015 Edge filtering causes mis-synchronization when falling edge at Rx input pin coincides with end of integration phase

When edge filtering is enabled (CCCRi.EFBI = '1') and when the end of the integration phase coincides with a falling edge at the Rx input pin it may happen, that the MCMCAN synchronizes itself wrongly and does not correctly receive the first bit of the frame. In this case the CRC will detect that the first bit was received incorrectly, it will rate the received FD frame as faulty and an error frame will be send.

The issue only occurs, when there is a falling edge at the Rx input pin within the last time quantum (tq) before the end of the integration phase. The last time quantum of the integration phase is at the sample point of the 11th recessive bit of the integration phase. When the edge filtering is enabled, the bit timing logic of the MCMCAN sees the Rx input signal delayed by the edge filtering. When the integration phase ends, the edge filtering is automatically disabled. This affects the reset of the FD CRC control unit at the beginning of the frame. The Classical CRC control unit is not affected, so this issue does not affect the reception of Classical frames.

In CAN communication, the MCMCAN may enter integrating state (either by resetting CCCRi.INIT or by protocol exception event) while a frame is active on the bus. In this case the 11 recessive bits are counted between the acknowledge bit and the following start of frame. All nodes have synchronized at the beginning of the dominant acknowledge bit. This means that the edge of the following start of frame bit cannot fall on the sample point, so the issue does not occur. The issue occurs only when the MCMCAN is, by local errors, mis-synchronized with regard to the other nodes, or not synchronized at all.

Glitch filtering as specified in ISO 11898-1:2015 is fully functional.

Edge filtering was introduced for applications where the data bit time is at least two tq (of the nominal bit time) long. In that case, edge filtering requires at least two consecutive dominant time quanta before the counter counting the 11 recessive bits for idle detection is restarted. This means edge filtering covers the theoretical case of occasional 1-tq-long dominant spikes on the CAN bus that would delay idle detection. Repeated dominant spikes on the CAN bus would disturb all CAN communication, so the filtering to speed up idle detection would not help network performance.

When this rare event occurs, the MCMCAN sends an error frame and the sender of the affected frame retransmits the frame. When the retransmitted frame is received, the MCMCAN has left integration phase and the frame will be received correctly. Edge filtering is only applied during integration phase, it is never used during normal operation. As integration phase is very short with respect to “active communication time”, the impact on total error frame rate is negligible. The issue has no impact on data integrity.

The MCMCAN enters integration phase under the following conditions:

- when CCCRi.INIT is set to ’0’ after start-up
- after a protocol exception event (only when CCCRi.PXHD = ’0’).

Scope

The erratum is limited to FD frame reception when edge filtering is active (CCCRi.EFBI = ’1’) and when the end of the integration phase coincides with a falling edge at the Rx input pin.

Effects

The calculated CRC value does not match the CRC value of the received FD frame and the MCMCAN sends an error frame. After retransmission the frame is received correctly.

Workaround

Disable edge filtering or wait on retransmission in case this rare event happens.

MCMCAN_AI.017 Retransmission in DAR mode due to lost arbitration at the first two identifier bits

When the MCMCAN CAN Node is configured in DAR mode (CANx.CCCRi.DAR = ’1’) the Automatic Retransmission for transmitted messages that have been disturbed by an error or have lost arbitration is disabled. When the transmission attempt is not successful, the Tx Buffer’s transmission request bit (CANx.TXBRPi.TRPz) shall be cleared and its cancellation finished bit (CANx.TXBCFi.CFz) shall be set.

When the transmitted message loses arbitration at one of the first two identifier bits, it may happen, that instead of the bits of the actually transmitted Tx Buffer, the CANx.TXBRPi.TRPz and CANx.TXBCFi.CFz bits of the previously started Tx Buffer (or Tx Buffer 0 if there is no previous transmission attempt) are written (CANx.TXBRPi.TRPz = ’0’, CANx.TXBCFi.CFz = ’1’).

If in this case the CANx.TXBRPi.TRPz bit of the Tx Buffer that lost arbitration at the first two identifier bits has not been cleared, retransmission is attempted.

When the CAN Node loses arbitration again at the immediately following retransmission, then actually and previously transmitted Tx Buffer are the same and this Tx Buffer’s CANx.TXBRPi.TRPz bit is cleared and its CANx.TXBCFi.CFz bit is set.

Scope

The erratum is limited to the case when the MCMCAN CAN Node loses arbitration at one of the first two transmitted identifier bits while in DAR mode.

The problem does not occur when the transmitted message has been disturbed by an error.

Effects

In this case it may happen, that the CANx.TXBRPi.TRPz bit is cleared after the second transmission attempt instead of the first.

Additionally it may happen that the CANx.TXBRPi.TRPz bit of the previously started Tx Buffer is cleared, if it has been set again. As in this case the previously started Tx Buffer has lost MCMCAN internal arbitration against the active Tx Buffer, its message has a lower identifier priority. It would also have lost arbitration on the CAN bus at the same position.

Workaround

None.

MCMCAN_AI.018 Tx FIFO Message Sequence Inversion

Assume the case that there are two Tx FIFO messages in the output pipeline of the Tx Message Handler (TxMH) and transmission of Tx FIFO message 1 is started:

- Position 1: Tx FIFO message 1 (transmission ongoing)
- Position 2: Tx FIFO message 2
- Position 3: --

Now a non-Tx FIFO message with a higher CAN priority is requested. Due to its priority it will be inserted into the output pipeline. The TxMH performs so called “message-scans” to keep the output pipeline up to date with the highest priority messages from the Message RAM. After the following two message-scans the output pipeline has the following content:

- Position 1: Tx FIFO message 1 (transmission ongoing)
- Position 2: non Tx FIFO message with higher CAN priority
- Position 3: Tx FIFO message 2

If the transmission of Tx FIFO message 1 is not successful (lost arbitration or CAN bus error) it is pushed from the output pipeline by the non-Tx FIFO message with higher CAN priority. The following scan re-inserts Tx FIFO message 1 into the output pipeline at position 3:

- Position 1: non Tx FIFO message with higher CAN priority (transmission ongoing)
- Position 2: Tx FIFO message 2
- Position 3: Tx FIFO message 1

Now Tx FIFO message 2 is in the output pipeline in front of Tx FIFO message 1 and they are transmitted in that order, resulting in a message sequence inversion.

Note: Within the scope of the scenario described above, in case of more than two Tx FIFO messages, the Tx FIFO message that has lost arbitration will be inserted after the next pending Tx FIFO message.

Scope:

The erratum describes the case when the MCMCAN uses both, dedicated Tx Buffers and a Tx FIFO (CAN_TXBCi.TFQM = ’0’) and the messages in the Tx FIFO do not have the highest internal CAN priority. The described sequence inversion may also happen between two non-Tx FIFO messages (Tx Queue or dedicated Tx Buffers) that have the same CAN identifier and that should be transmitted in the order of their buffer numbers (not the intended use).

Effects:

In the described case it may happen that two consecutive messages from the Tx FIFO exchange their positions in the transmit sequence.

Workaround

When transmitting messages from a dedicated Tx Buffer with higher priority than the messages in the Tx FIFO, choose one of the following workarounds:

First Workaround:

Use two dedicated Tx Buffers, e.g. use Tx Buffers 4 and 5 instead of the Tx FIFO.

The pseudo-code below replaces the function that fills the Tx FIFO.

- Write message to Tx Buffer 4
- Transmit Loop:
  - Request Tx Buffer 4 - write TXBAR.A4
  - Write message to Tx Buffer 5
  - Wait until transmission of Tx Buffer 4 completed – CAN_IRi.TC, read CAN_TXBTOi.TO4
  - Request Tx Buffer 5 - write CAN_TXBARi.AR5
  - Write message to Tx Buffer 4
  - Wait until transmission of Tx Buffer 5 completed – CAN_IRi.TC, read CAN_TXBTOi.TO5

Second Workaround:

Assure that only one Tx FIFO element is pending for transmission at any time. The Tx FIFO elements may be filled at any time with messages to be transmitted, but their transmission requests are handled separately. Each time a Tx FIFO transmission has completed and the Tx FIFO gets empty (CAN_IRi.TFE = ’1’) the next Tx FIFO element is requested.

Third Workaround:

Use only a Tx FIFO. Send the message with the higher priority also from Tx FIFO.

Drawback: The higher priority message has to wait until the preceding messages in the Tx FIFO have been sent.

MCMCAN_AI.019 Unexpected High Priority Message (HPM) interrupt

There are two configurations where the issue occurs:

Configuration A:

- At least one Standard Message ID Filter Element is configured with priority flag set (S0.SFEC = “100”/”101”/”110”)
- No Extended Message ID Filter Element configured
- Non-matching extended frames are accepted (GFC.ANFE = “00”/”01”)

The HPM interrupt flag IR.HPM is set erroneously on reception of a non-high-priority extended message under the following conditions:

1. A standard HPM frame is received, and accepted by a filter with priority flag set
   --> Interrupt flag IR.HPM is set as expected
2. Next an extended frame is received and accepted because of GFC.ANFE configuration
   --> Interrupt flag IR.HPM is set erroneously

Configuration B:

- At least one Extended Message ID Filter Element is configured with priority flag set (F0.EFEC = “100”/”101”/”110”)
- No Standard Message ID Filter Element configured
- Non-matching standard frames are accepted (GFC.ANFS = “00”/”01”)

The HPM interrupt flag IR.HPM is set erroneously on reception of a non-high-priority standard message under the following conditions:

1. An extended HPM frame is received, and accepted by a filter with priority flag set
   --> Interrupt flag IR.HPM is set as expected
2. Next a standard frame is received and accepted because of GFC.ANFS configuration
   --> Interrupt flag IR.HPM is set erroneously

Scope

The erratum is limited to:

- Configuration A:
  – No Extended Message ID Filter Element configured and non-matching extended frames are accepted due to Global Filter Configuration (GFC.ANFE = “00”/”01”).
- Configuration B:
  – No Standard Message ID Filter Element configured and non-matching standard frames are accepted due to Global Filter Configuration (GFC.ANFS = “00”/”01”).

Effects

Interrupt flag IR.HPM is set erroneously at the reception of a frame with:

- Configuration A: extended message ID
- Configuration B: standard message ID

Workaround

Configuration A:

Setup an Extended Message ID Filter Element with the following configuration:

- F0.EFEC = “001”/”010” - select Rx FIFO for storage of extended frames
- F0.EFID1 = any value - value not relevant as all ID bits are masked out by F1.EFID2
- F1.EFT = “10” - classic filter, F0.EFID1 = filter, F1.EFID2 = mask
- F1.EFID2 = zero - all bits of the received extended ID are masked out

Now all extended frames are stored in Rx FIFO 0 respectively Rx FIFO 1 depending on the configuration of F0.EFEC.

Configuration B:

Setup a Standard Message ID Filter Element with the following configuration:

- S0.SFEC = “001”/”010” - select Rx FIFO for storage of standard frames
- S0.SFID1 = any value - value not relevant as all ID bits are masked out by S0.SFID2
- S0.SFT = “10” - classic filter, S0.SFID1 = filter, S0.SFID2 = mask
- S0.SFID2 = zero - all bits of the received standard ID are masked out

Now all standard frames are stored in Rx FIFO 0 respectively Rx FIFO 1 depending on the configuration of S0.SFEC.

MTU_TC.012 Security of CPU Cache Memories During Runtime is Limited

MTU chapter “Security Applications” in the User’s Manual describes that selected memories with potentially security relevant content are initialized under certain conditions to prevent reading of their data or supplying manipulated data.

The description is correct, but the initialization of CPU cache and cache tag memories triggered by MBIST enable/disable and when mapping/un-mapping these memories to/from system address space using MEMMAP register is of limited value:

- These memories stay functional as cache in the address mapped state. Therefore software can enable address mapping and afterwards watch cache usage of the application (this is a debug feature). Even manipulation of the cache content is feasible.
- It is possible to abort an ongoing memory initialization.

The security of memory initialization during startup is not affected. Also protection of FSI0 and HSM memories is not limited.

Workaround

Handle security relevant data exclusively inside HSM. Protect the application code by locking external access (e.g. lock debug interface, prevent boot via serial interface). Consider validation of application code by HSM secure boot.

MTU_TC.017 Unexpected alarms after application reset

As described in the MTU chapter “Alarms after startup” section, in case of an application reset, there are no SSH alarms or status bits expected to be triggered.

However, this device deviates from this expected behavior, and status flags AG0.SF10 and AG1.SF10 (DMEM Uncorrectable critical error) are set also after an application reset. Correspondingly, the OPERR[0] bits of the following SSHs are also set in the corresponding MCi_FAULTSTS registers after an application reset:

- MC0 (CPU0_DMEM),
- MC34 (CPU0_DMEM1), and
- MC35 (CPU1_DMEM1)

Note: In contrast to alarms resulting from real errors, for these unexpected alarms after application reset MCi_ERRINFO = 0x0 (i = 0, 34, 35).

Workaround

The application software may clear the above mentioned alarms and errors after an application reset if MCi_ERRINFO = 0x0 (i = 0, 34, 35), and proceed.

In case these errors occur during normal application run, this shall be considered as a real error.

MTU_TC.018 Gated SRAM alarms

Due to a corner case, SRAM alarms to the SMU for SRAM errors are not correctly generated for the following modules.

- GTM: ALM6[10], ALM6[11];
- DMA, SCR: ALM6[19], ALM6[20];

- CPUx: ALMx[4], ALMx[7], ALMx[10]
  (x = 0..n; n depends on number of CPUs available in product).

Background:

From the SRAMs, the following errors are triggered to the SMU:

- ECC-correctable error: Triggered on a read access to SRAM.
- ECC-uncorrectable error: Triggered on a read access to SRAM.
- Address error: Triggered on read or write access to SRAM.

In case of an error, normally these alarms are triggered appropriately on each read or write access.

However, due to this corner case, for certain SRAMs mentioned above, the alarm is not triggered on the read or write access on which the error is generated, rather, it is generated only on the **next** access to the SRAM or to an SSH register (e.g. MCx_ECCD register).

Note: Only the SMU alarm generation is affected by this issue and not the error triggering to the module. E.g. error notification to GTM MCS still works as expected and the MCS may be stopped on an uncorrectable ECC error.

Additionally, only the alarm propagation is gated in this corner case, i.e. the error status is still correctly stored in the MCx_ECCD, MCx_FAULTSTS registers.

Workaround

For GTM & SCR SRAMs

Read the MCx_ECCD register periodically, depending on application safety considerations, for example within each diagnostic test interval.

- Corresponding SSH instances:
  - GTM: MC53..MC60;
  - SCR: MC77, MC78.

For DMA & CPU SRAMs (except DLMUx_STBY)

No workaround is recommended, because here the issue affects only the address error generation on a write access. In this case, the next read access (when the data would be used) will trigger the error.

For DLMU_STBY

The issue occurs in a corner case just before entering standby mode. Therefore, if standby mode is used and Standby RAM is enabled (PMSWCR0.STBYRAMSEL ≠ 000_B) - then just before entering standby, perform an additional dummy read to DLMU_STBY location 9000 0000_H (when using CPU0 dLMU RAM) and 9001 0000_H (when using CPU1 dLMU RAM). This dummy read triggers the alarm propagation and ensures that no alarms are lost due to standby entry.

<u>PER_PLL_TC.001</u> Peripheral PLL weakness for 25 MHz input clock when using Divider Bypass

Activation of the /1.6 Divider Bypass function of the Peripheral PLL (register setting PERPLLCON0.DIVBY = 1_B) can result in an unstable clock being generated on the f_{PLL2} output. This is due to a design weakness.

Therefore, only the Divider Bypass configuration PERPLLCON0.DIVBY = 0_B must be used.

Impacted Use Cases

The AURIX™ TC3xx User's Manual describes two recommended configuration sequences in chapter "Use Cases" of the "Clocking System" chapter:

- Setting AAA is defined for a 20 MHz crystal / clock input.
  - This setting uses PERPLLCON0.DIVBY = 0_B, therefore no change is required when using this setting.
- Setting BBB is defined for a 25 MHz crystal / clock input.
  - This setting uses PERPLLCON0.DIVBY = 1_B, therefore changes are required when using this setting.

Workaround

The configuration of the Peripheral PLL has to be selected such that the Divider Bypass function is disabled, i.e. PERPLLCON0.DIVBY = 0_B.

In the recommended configuration setting BBB for 25 MHz in the User's Manual, in Step 2 the following modification is required:

- PERPLLCON0 = $00013E00_H$ (instead of $00013E01_H$)

Consequence

With this modification, f_{PLL2} changes to 250 MHz (from 200 MHz), while f_{PLL1} remains unchanged (160 MHz).

Connected to the f_{PLL2} output are the following modules: MSC, ASCLIN (input $f_{ASCLINF}$), QSPI, and I2C (see also table “CCU Clock Options” in chapter “Clocking System”).

The 250 MHz instead the original 200 MHz would violate the maximum allowed frequency of 200 MHz for MSC, ASCLIN, and QSPI; therefore additional considerations are required for these modules.

Workaround - additional considerations for MSC

The MSC has the option to use either f_{PLL1} or f_{PLL2} as clock, whatever represents the more suitable clock frequency.

Therefore, select f_{PLL1} as clock source for MSC: based on f_{PLL1} = 160 MHz still 40 MHz and 80 MHz are possible for f_{MSC} addressing the application requirements.

Workaround - additional considerations for QSPI

As the QSPI is a synchronous communication interface the clock frequency for the QSPI defines the maximum achievable baudrate.

Changing the configuration for K3DIV from 2 to 3 would result in a frequency of f_{PLL2} = 166.667 MHz instead the targeted 200 MHz; this would reduce the maximum baudrate from 50 MBaud to 41.667 MBaud (based on the minimum allowed divider n = 4). If this baudrate still fulfills the application requirements no further impact needs to be analyzed.

The QSPI also has the option to use either f_{PLL1} or f_{PLL2} as clock, whatever represents the more suitable clock frequency.

Therefore, when selecting f_{PLL1} as clock source for QSPI, based on f_{PLL1} = 160 MHz still 40 MBaud application requirements are fulfilled.

Workaround - additional considerations for ASCLIN

As the ASCLIN is an (a)synchronous communication interface, the clock frequency for the ASCLIN defines the maximum achievable baudrate.

Changing the configuration for K3DIV from 2 to 3 result in a frequency of f_{PLL2} = 166.667 MHz instead the targeted 200 MHz; this would reduce the maximum baudrate from 50 MBaud to 41.667 MBaud (based on the minimum allowed divider n = 4). If this baudrate still fulfills the application requirements no further impact needs to be analyzed.

The ASCLIN also has the option to use either f_{PLL1} ($f_{ASCLINS}$) or f_{PLL2} ($f_{ASCLINF}$) as clock, whatever represents the more suitable clock frequency. When selecting f_{PLL1} as clock source for ASCLIN, based on f_{PLL1} = 160 MHz still 40 MBaud application requirements are fulfilled.

The ASCLIN also provides a fractional divider that can be used for baudrate generation.

Workaround - additional considerations for I2C

As the I2C is a synchronous communication interface that does not require high frequencies for its operation, and the I2C module provides a fractional divider, there should be no issue to derive the intended clock frequency from the reduced value of f_{PLL2}. Only the I2C specific clock control register requires an update.

Workaround - additional considerations if ERAY is not used

In the recommended Peripheral PLL configuration f_{PLL1} is configured with 160 MHz in order to finally provide the ERAY module with a 80 MHz module clock required by the ERAY protocol for the tick timing.

If the ERAY function is not required for the application there is also no blocking point to configure f_{PLL1} = 200 MHz.

Examples

The following table shows some examples for resulting clock frequencies based on the assumption f_{REF} = 25 MHz, P = 1, N = 32, f_{DCO} = 800 MHz:

Table 6 Configuration Options for Peripheral PLL with input clock = 25 MHz

K2	K3	DIVBY	f_{PLL1} [MHz]	f_{PLL2} [MHz]	Comments
5	2	**1**	160	200	Setting BBB (see User's Manual) - **DIVBY=1 must not be used!**
5	2	0	160	**250**	Setting BBB with DIVBY=0: **f_{PLL2} too high**
5	3	0	160	166.67	
5	4	0	160	125	
4	3	0	200	166.67	
4	4	0	200	125	

PMS_TC.005 Voltage rise at P33 and P34 up to 2.5V during start-up and power-down

The default pad behavior at P33 and P34 pins is pull-up up to 2.5V ($V_{LVDRSTSB}$) during the ramp-up phase and below 2.5V for the ramp-down phase of the V_{EVRSB} supply even if HWCFG[6] =0.

Tristate control information based on HWCFG[6] latched with V_{EXT} supply ramp can't be used within the V_{EVRSB} supply domain if the V_{EVRSB} input supply voltage has not reached its low voltage reset limit $V_{LVDRSTSB}$.

Workaround

None.

PMS_TC.006 PORST not released during Cold Power-on Reset until VDDM is available

Upon a cold power-on reset, the PORST pin is kept asserted by the PMS until the ADC Analog Supply voltage (VDDM) is above 500 mV. This might lead to an additional start-up delay dependent on when VDDM is available from the external regulator relative to the VEXT, VDDP3 and VDD supplies.

During operation, if VDDM drops below the secondary monitor undervoltage threshold, an SMU alarm is generated. If VDDM further drops below 500 mV, the dedicated ADC of the secondary voltage monitor stops converting and the Secondary Monitor Activity Counter (EVRMONSTAT1.ACTVCNT) freezes at the last value.

Workaround

The ADC Analog Supply voltage (VDDM) has to be available and needs to be above 500 mV to ensure proper release of PORST during start-up and proper functioning of secondary monitors.

PMS_TC.007 VDDP3 or VDD Overvoltage during start-up may not be detected by PBIST

In AURIX™ TC3xx devices, Power Built in Self Test (PBIST) is introduced to ensure that the supply voltages do not exceed absolute maximum limits during the start-up phase.

However, for a VDDP3 or VDD overvoltage event during start-up beyond operational upper limits, the PBIST is not able to detect this overvoltage event.

Workaround

Check the VDDP3 overvoltage condition in registers EVRSTAT (flag OV33) and EVRMONSTAT1 (field ADC33V) in software additionally during the start-up phase before enabling the corresponding SMU alarm.

Check the VDD overvoltage condition in registers EVRSTAT (flag OVC) and EVRMONSTAT1 (field ADCCV) in software additionally during the start-up phase before enabling the corresponding SMU alarm.

QSPI_TC.006 Baud rate error detection in slave mode (error indication in current frame)

According to the specification, a baud rate error is detected if the incoming shift clock supplied by the master has less than half or more than double the expected baud rate (determined by bit field GLOBALCON.TQ).

However, in this design step, a baud rate error is detected not only if the incoming shift clock has less than half the expected baud rate (as specified), but also already when the incoming shift clock is somewhat (i.e. less than double) higher than the expected baud rate.

In this case, the baud rate error is indicated in the current frame.

Workaround

It is recommended not to rely on the baud rate error detection feature, and not to use the corresponding automatic reset enable feature (i.e. keep GLOBALCON.AREN=0_B).

The baud rate error detection feature in slave mode is of conceptually limited use and is not related to data integrity. Data integrity can be ensured e.g. by parity, CRC, etc., while clocking problems of an AURIX™ master are detected by mechanisms implemented in the master.

Protection against the effects of high frequency glitches is provided by the spike detection feature in slave mode.

QSPI_TC.009 USR Events for PT1=2 (SOF: Start of Frame)

In master mode, when the interrupt on USR event is associated with Start of Frame (i.e. USREN=1_B, PT1=2 in register GLOBALCON1, BACON.UINT=1_B), then flag STATUS.USRF is not set and the interrupt is not triggered for the first frame.

Workaround

In the configuration where the interrupt on USR event is associated with Start of Frame (i.e. USREN=1_B, PT1=2 in GLOBALCON1, BACON.UINT=1_B), first transmit a “dummy” frame with this configuration. Then, for all subsequent frames, flag USRF will be set and the interrupt on USR event will be generated as expected.

QSPI_TC.010 Move Counter Mode - USR Events for PT1=4 (RBF: Receive Buffer Filled)

When a master operates in Move Counter Mode (MCCON.MCEN=1_B), and the interrupt on USR event is associated with Receive Buffer Filled (i.e. USREN=1_B, PT1=4 in register GLOBALCON1), the enable signal in BACON.UINT is only evaluated at the start of frame event.

This means in an ongoing frame the status of UINT in the first BACON control word involved determines whether flag STATUS.USRF is set and a user interrupt is generated or not. The status of UINT in following BACON control words in this frames' transmission is not considered.

Workaround

In case the Receive Buffer Filled event shall only be used as interrupt on USR event for parts of a frame, initialize e.g. BACON.UINT=1_B and GLOBALCON.PT1=4 before start of frame, and use GLOBALCON1.USREN to selectively disable/enable the user interrupt during frame transmission.

QSPI_TC.013 Slave: No RxFIFO write after transmission upon change of BACON.MSB

While a slave transmission is in progress, and if the BACON.MSB configuration is changed for the subsequent frame, then the RxFIFO write of the currently received frame may not occur.

Also in case of a TxFIFO underflow, the RxFIFO write of the currently received frame may not occur.

Workaround

As a general recommendation, in slave mode the configuration should be done before any transmission starts.

In particular to avoid the problem described above, the re-configuration of the BACON has to be done after the RxFIFO write has occurred. This implies the need for a gap between frames if a BACON update occurs.

QSPI_TC.014 Slave: Incorrect parity bit upon TxFIFO underflow

When a slave TxFIFO underflow occurs, the slave transmits only “ones” in response to a request of the master.

If parity is enabled, also the parity bit transmitted by the slave is always set to “1”. This may be incorrect, depending on data length and parity type.

Workaround

If parity is enabled, select even parity if data length is odd, and select odd parity if data length is even.

SCR_TC.015 Bit SCU_PMCON1.WCAN_DIS does not disable WCAN PCLK input

Setting bit SCU_PMCON1.WCAN_DIS to 1_B has no effect – the WCAN clock input (PCLK) is not disabled. Power consumption of the WCAN module will not decrease as expected.

Workaround

In order to keep power consumption at a minimum, the WCAN module must not be enabled (WCAN_CFG.WCAN_EN = 0_B).

SCR_TC.016 DUT response to first telegram has incorrect C_START value

Note: This problem is only relevant for tool development, not for application development.

The C_START value returned by the SCR OCDS of the DUT (device under test) in response to a first telegram is wrong.

Each monitor processed command starts with sending a telegram containing the CMD (e.g. READ_BYTE). The response to this telegram should be a telegram containing the C_START value of 0x1.

Instead, the value sent by the DUT is a random value.

Workaround

Do not
poll for TRF==1 on the first telegram of the monitor processed commands, and do not
evaluate the return value of the first telegram from the DUT. Even though the returned C_START is wrong, the returned checksum is correct, and should be checked with the theoretical C_START value of 0x01.

SCR_TC.018 SSC Receive FIFO not working

The receive FIFO of the SSC module is not working properly. An unexpected receive FIFO full indication can be set.

Workaround

Do not use the receive FIFO.

Read the received data from the receive buffer register SSC_RBL each time a receive interrupt event is signaled (flag IRCON1.RIR).

The received data must be read before the next data is received.

SCR_TC.019 Accessing the XRAM while SCR is in reset state

When accessing the XRAM while the SCR is executing a reset, the following erroneous behavior will occur:

- A read access returns 0 instead of the actual XRAM contents.
- A write access has no effect, the data will not be written to the XRAM.

Workaround

One of the following methods will avoid this problem:

1. Check the SCR reset status bit PMSWSTAT.SCRST before and after any read/write transaction to the XRAM:
   a) If the bit is set before the transaction, clear bit PMSWSTAT.SCRST and perform the desired XRAM access.
   b) If the bit is set after the transaction, clear bit PMSWSTAT.SCRST and repeat the XRAM read/write access. OR
2. Disable the SCR generated reset sources. OR
3. Disable the entire SCR (no SCR reset can occur): i.e. set
   - PMSWCR0.SCRWKEN = 0_B – wake-up via SCR disabled;
   - PMSWCR4.SCREN = 0_B – SCR disabled.

SCR_TC.020 Stored address in mon_RETH may be wrong after a break event

Note: This problem is only relevant for tool development, not for application development.

When setting a breakpoint via the SCR debugger connection on address $xxFE_H$ of an instruction, the stored address in mon_RETH is wrong if mon_RETL contains 00_H (see also section “Calculation of the return address upon a break event” in the SCR chapter). This effect will happen whenever a carry bit should be propagated from the lower 8 bits to the upper 8 bits of the address.

Workaround

If mon_RETL contains 00_H after a breakpoint was hit, the debugger tool must increment mon_RETH by 1 before performing the calculation of the return address as described in section “Calculation of the return address upon a break event” in the SCR chapter.

3 Deviations from Electrical- and Timing Specification

ADC_TC.P009 Increased TUE for G10 when using Alternate Reference

When using the alternate reference (G10CH0) for conversions on channels of converter group G10, the Total Unadjusted Error (TUE) of the conversion results may increase with temperature

- up to ±12 LSB_{12} for operation with converter reference clock f_{ADCI} = 32 MHz.
- up to ±25 LSB_{12} for operation with converter reference clock f_{ADCI} = 40 MHz.
- up to ±46 LSB_{12} for operation with converter reference clock f_{ADCI} = 53.33 MHz.

Note: This problem will not occur in the lower range of the ADC analog supply voltage (VDDM < 4.5 V), as f_{ADCI} is limited to 26.67 MHz in this case (see Data Sheet).

Recommendation

- Do not use the alternate reference of converter group G10 for f_{ADCI} > 26.67 MHz.
- Use a different converter group Gx (x≠10) when an alternate reference voltage is required for conversions with f_{ADCI} > 26.67 MHz.

4 Application Hints

ADC_TC.H026 Additional Waiting Phase in Slow Standby Mode

When a conversion is requested while slow standby mode is configured and the respective converter currently is in standby state, the extended wakeup time t_{WU} must be added to the intended sample time (see section “Analog Converter Control” in the Target Specification/User’s Manual).

While idle precharge is disabled (GxANCFG.IPE = 0_B), an additional waiting phase of 1.6 µs (@f_{ADC} = 160 MHz) is inserted automatically. Operation starts after this phase.

However, if the slow standby state is left after just 1 clock cycle, this waiting phase is omitted.

Recommendation

It is, therefore, recommended to add the specified extended wakeup time (t_{WU}) when leaving the standby state in all cases, to ensure proper operation.

AGBT_TC.H004 Configuration of registers PYCR2 and PACR2

Settings different to the reset configuration are required for the following fields of registers PYCR2 and PACR2:

- Register PYCR2.[26:24] = 100_B (CKRXTERM)
- Register PYCR2.[12:8] = 01101_B (TXOCDSLE)
- Register PACR2.[26:24] = 010_B (PICAPSEL)

ASCLIN_TC.H001 Bit field `FRAMECON.IDLE` in LIN slave mode

In LIN slave mode, bit field `FRAMECON.IDLE` has to be set to 000_B (default after reset), i.e. no pause will be inserted between transmission of bytes.

For `FRAMECON.IDLE` > 000_B, the inter-byte spacing of the ASCLIN module is not working properly in all cases in LIN slave mode (no bit errors are detected by the ASCLIN module within the inter-byte spacing).

BROM_TC.H008 CAN BSL does not support DLC = 9 and DLC = 11

The CAN Bootstrap loader (BSL) only supports messages where the number of data bytes is a multiple of 8.

Therefore, Data Length Code settings DLC = 11 (number of data bytes = 20) and DLC = 9 (number of data bytes = 12) are not allowed (see also chapter “CAN BSL flow” of chapter “AURIX™ TC3xx Platform Firmware”).

Recommendation

When using the CAN Bootstrap loader, only use settings where DLC ≠ 9 or DLC ≠ 11.

BROM_TC.H009 Re-Enabling Lockstep via BMHD

For all CPUs with lockstep option, the lockstep functionality is controlled by Boot Mode Headers (BMHD) loaded during boot upon a reset trigger.

If lockstep is disabled for a CPUx with lockstep functionality, re-enabling (e.g. via a different BMHD) is not reliably possible if warm PORST, System or Application reset is executed.

Recommendation

Use cold PORST if lockstep is disabled and shall be re-enabled upon the reset trigger.

BROM_TC.H011 Assertion of ALM7[14] after Cold/Warm PORST

After a cold or warm PORST, if the tool interface is not locked and no Halt After Reset has been requested, the startup software checks for a 64-bit signature at

3 predefined locations within EMEM as part of the prolog code handling (see chapter “Startup with Prolog Code in EMEM” in the TC3xxED documentation).

This may trigger ALM7[14] (EMEM ECC Error) if the EMEM is not ECC-clean initialized at that time.

Note: No entry into the ETRR registers is made due to this issue.

Recommendation

Ignore ALM7[14] after cold or warm PORST if the corresponding ETRR registers contain no entry.

BROM_TC.H012 Availability of V_{DDSB} during start-up

Devices with integrated EMEM have a separate EMEM SRAM Standby Power Supply (V_{DDSB}).

As documented e.g. in chapter “Power Supply Concept” in the TC3xxED documentation

- V_{DDSB} has to be supplied when V_{DD} is supplied and the EMEM is unlocked
- V_{DDSB} can be unsupplied when V_{DD} is supplied and PORST is active or the EMEM is locked
- V_{DDSB} can be supplied when V_{DD} is unsupplied and PORST is active (EMEM standby mode).

Note: If V_{DDSB} is not supplied at PORST release and the EMEM is unlocked, cross current from V_{DD} domain could lead to device damage.

Software may check availability of V_{DDSB} via bit SBRCTR.STBPON when EMEM is unlocked.

During start-up, however, there are situations where the EMEM is unlocked for a short time without checking SBRCTR.STBPON before.

Recommendation

The external system must ensure that V_{DDSB} is within the active operation range (1.25 V ± 10%, see Data Sheet) after PORST release during startup.

BROM_TC.H014 SSW behavior in case of wrong state or uncorrectable error in UCBs - Documentation Update

The boot sequence terminates and the device is put into error state (endless loop) in the following cases:

- **Wrong state** - i.e. different from CONFIRMED or UNLOCKED (in case an UCB has ORIGINAL and COPY: wrong state of the both) – for the following UCBs:
  - UCB_BMHDx, UCB_SWAP, UCB_SSW, UCB_USER, UCB_PFLASH, UCB_DFLASH, UCB_DBG, UCB_HSM, UCB_HSMCOTP0...1, UCB_HSMCFG, UCB_ECPRIO, UCB_OTP0...7, UCB_REDSEC, UCB_TEST, UCB_RETEST.
- **Uncorrectable ECC error** within the used locations when state valid (CONFIRMED or UNLOCKED) – for the following UCBs:
  - UCB_SSW, UCB_PFLASH, UCB_DFLASH, UCB_DBG, UCB_HSM, UCB_HSMCOTP0...1, UCB_ECPRIO, UCB_OTP0...7, UCB_REDSEC, UCB_RETEST.
- For UCB_SWAP ORIGINAL/COPY – according to the descriptions in User's Manual.

Recommendation

Instructions to be followed for UCB-reprogramming (in order to avoid unexpected boot termination):

- always verify the changed contents before confirming the UCB state
- strictly follow the sequence in section “UCB Confirmation” in the “Non Volatile Memory (NVM)” chapter of the User's Manual[1].

CCU_TC.H012 Configuration of the Oscillator- Documentation Update

As described in chapter „Configuration of the Oscillator” in the CCU chapter of the User's Manual, configuration of the oscillator is always required before an external crystal / ceramic resonator can be used as clock source.

1) For TC39x A-step: chapter “Program Memory Unit (PMU)” of the Target Specification.

Depending on the supply voltage ramp-up characteristics the behavior described in the following note may be observed:

Note: If the delay between ~V_{EXT}/2 (~2.5V) and V_{DD} (1.25V) at power-on is more than about 100µs then the oscillator starts to swing (crystal/resonator connected). As soon as V_{DD} is in the specified range the oscillator is set to External Input Mode and the oscillation decays. This characteristic behavior has no impact to the oscillator start-up initiated by software.

<u>CPU_TC.H016</u> List of OS and I/O Privileged Instructions - Documentation Update

The RESTORE instruction is missing in table “OS and I/O Privileged Instructions” of the TriCore™ TC1.6.2 core architecture manual - Instruction set (volume 2, V1.0 2017-01-11).

Documentation Update

The RESTORE instruction should be added in column “User-1 Mode” in table “OS and I/O Privileged Instructions” of the TriCore™ TC1.6.2 core architecture manual - Instruction set.

<u>FlexRay_AI.H004</u> Only the first message can be received in External Loop Back mode

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

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

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

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

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

- wait for no activity (`TEST1.AOx`=0 -> bus idle)

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

Now the next transmission can be requested.

FlexRay_AI.H005 Initialization of internal RAMs requires one eray_bclk cycle more

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

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

FlexRay_AI.H006 Transmission in ATM/Loopback mode

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

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

FlexRay_AI.H007 Reporting of coding errors via TEST1.CERA/B

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

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

FlexRay_AI.H009 Return from test mode operation

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

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

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

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

FlexRay_TC.H003 Initialization of E-Ray RAMs

After Power-On reset, the E-Ray RAMs hold arbitrary values which causes ECC errors (MHDS) when a read operation is performed on an E-Ray RAM location. Hence the E-Ray RAMs should be initialized always after a Power-On reset.

Recommendation

The E-Ray RAMs initialization can be performed using the CLEAR_RAMS command of the E-Ray module. A safe initialization sequence of the E-Ray RAM blocks using the CLEAR_RAMS command is described in section “CLEAR_RAMS Command” of chapter “FlexRay™ Protocol Controller (E-Ray)” in the AURIX™ TC3xx Target Specification/User’s Manual.

GETH_AI.H001 Preparation for Software Reset

When a kernel reset or software reset (via bit DMA_MODE.SWR) shall be performed, the GETH module must be clocked (MII: RXCLK and TXCLK; RMII: REFCLK; RGMII: GREFCLK and RXCLK) and be in a defined state to avoid unpredictable behavior.

Therefore, it is recommended to use the defined sequence listed below if frame transactions took place before setting bit SWR:

1. Finish running transfers and make sure that transmitters and receivers are set to stopped state:
   a) Check the RPSx and TPSx status bit fields in register DMA_Debug_Status0/1.
   b) Check that MTL_RXQ0_DEBUG, MTL_RXQi_DEBUG, MTL_TXQ0_DEBUG and MTL_TXQi_DEBUG register content is equal to zero.
   Note: it may be required to wait 70 f_{SPB} cycles after the last reset before checking if RXQSTS in MTL_RXQ0_DEBUG and MTL_RXQi_DEBUG are zero.
2. Wait until a currently running interrupt is finished and globally disable interrupts.
3. Apply kernel reset to GETH module:
   a) Deactivate Endinit protection, as registers KRST0/1 and KRSTCLR can only be written in Supervisor Mode and when Endinit protection is not active.
   Write to corresponding RST bits of KRST0/1 registers to request a kernel reset. The reset status flag KRST0.RSTSTAT may be cleared afterwards by writing to bit CLR in the KRSTCLR register.
   Re-activate Endinit protection.
   b) Wait 70 f_{SPB} cycles, then check if RXQSTS in MTL_RXQ0_DEBUG and MTL_RXQi_DEBUG are zero.
4. Configure the same mode as before (MII, RMII or RGMII) in bit field GPCTL.EPR.
5. Apply software reset by writing to the DMA_MODE.SWR bit.
   Wait 4 f_{SPB} cycles, then check if DMA_MODE.SWR = 0_B.

If coming directly from Power-on Reset (i.e. no frame transaction took place yet), it is sufficient to follow the simplified sequence:

1. Configure the desired mode (MII, RMII or RGMII) in bit field GPCTL.EPR
2. Apply software reset by writing to the DMA_MODE.SWR bit.
   Wait 4 f_{SPB} cycles, then check if DMA_MODE.SWR = 0_B.

GTM_TC.H010 Trigger Selection for EVADC and EDSADC

If the GTM output selection in the SELz bit fields for ADC triggers (registers ADCTRIGxOUTy, DSADCOUTSELxy) is changed during SW runtime, multi-bit changes may lead to unintended ADC triggering.

Recommendation

Before changing the trigger source in the GTM output selection fields SELz, ensure that the ADCs at the trigger destination will not react on intermediate state changes of the trigger signals.

GTM_TC.H019 Register GTM_RST - Documentation Update

In the current documentation, bit 0 in register GTM_RST is described as

- **Type**: r
- **Description**: Reserved - Read as zero, should be written as zero.

Documentation Update

Actually, bit 0 in register GTM_RST is implemented as follows:

- **Type**: **rw**
- **Description**: Reserved - Read as zero, **shall** be written as zero.

Note: This Application Hint relates to problem GTM-IP-316 reported by the GTM IP supplier. On this AURIX™ TC3xx device step, the reported problem has no effect, independent of the value written to bit GTM_RST.0. However, GTM_RST.0 shall always be written with 0_B as documented in the register description to ensure compatibility with future versions.

I2C_TC.H008 Handling of RX FIFO Overflow in Slave Mode

If the I2C kernel has detected a RX FIFO overflow in slave mode, a RX_OFL_srq request in generated, the incoming character is discarded, and the kernel puts a not-acknowledge on the bus and changes to listening state.

However, it does not generate an EORXP_ind signal, so that the remaining characters in the FIFO can not be moved out by means of data transfer requests.

Recommendation

Upon an RX FIFO overflow in slave mode, received data may be invalid. However, they may be read from the FIFO e.g. for analysis if required.

In order to flush the FIFO and correctly resume communication

- set bit RUNCTRL.RUN = 0_B (switch to configuration mode),
- set bit RUNCTRL.RUN = 1_B (participate in I2C communication).

MCDS_TC.H007 Program trace of CPUx (x > 0) program start not correct

Note: This problem is only relevant for development tools.

All CPUs - except CPU0 - need to be started by user software from another CPU. This user software writes the PC and starts the CPU.

If this phase is traced with MCDS, the program trace for the first two executed instructions is not correct. The start PC is not visible and depending on the trace mode, the address of the second instruction is shown twice in the trace and there can be a wrong IPI message. However these effects are limited to the first two instructions.

Nevertheless this is confusing for the user and it can be an issue for trace based code coverage tools.

Recommendations

- A trace tool can ignore the generated trace messages for the first two instructions and replace it with proper messages.

- A trace tool can notify the user that the initial trace sequence for the first two instructions at startup is not correct.

MCMCAN_TC.H006 Unintended Behavior of Receive Timeout Interrupt

On following conditions:

1. Receive timeout feature is enabled (i.e. NTRTR.RELOAD != 0), **and**
2. Received CAN frames are stored in RxFIFO 0/1 or Dedicated Rx Buffers, **and**
3. Respective New CAN frame received interrupts are disabled (i.e. bits IE.RF0NE, IE.RF1NE or IE.DRXE are 0),

then an unintended receive timeout interrupt (if enabled, i.e. NTRTR.TEIE = 1) is triggered, although a valid CAN frame is newly received and stored in the respective RxFIFO 0/1 or Dedicated Rx Buffers.

Recommendation

Enable the corresponding receive interrupt via bits IE.RF0NE, IE.RF1NE, or IE.DRXE, depending on the usage of RxFIFO0/1 or dedicated Rx Buffers for proper function of the receive timeout interrupt.

Example

If RxFIFO 0 is used, set IE.RF0NE =1.

MTU_TC.H013 First Write Access to SSH registers of SCR_RAMINT

The first write to specific SSH registers of the SCR Internal RAM SSH instance after the SSH was enabled (e.g. via MEMTEST2.SCR_RAMINT) may not be executed.

Affected Registers

The following registers of SCR_RAMINT (MC78) are affected by this problem:

- CONFIG0, CONFIG1, MCONTROL, RANGE, RDBFLy

Note: The following SSH registers of SCR_RAMINT (MC78) are not affected: ECCS, ECCD, ETRRx, ALMSRCS, FAULTSTS, ERRINFOx. Registers REVID and MSTATUS need not be considered, as they are read-only.

Recommendation

After enabling SCR_RAMINT, perform a “dummy-read” e.g. from register MSTATUS before executing the first write on one of the affected SSH registers listed above.

MTU_TC.H015 ALM7[0] may be triggered after cold PORST

During firmware start-up after cold PORST, alarm status flag AG7.SF0 (correctable SRAM error) may erroneously be set to 1, although no error occurred. This is due to a dummy read to an uninitialized SRAM by firmware.

Note: No entry into any of the ETRR registers is made due to this issue.

Recommendation

As alarms for correctable errors are uncritical in general, no action is required (alarm can be ignored). The application may only react on the error overflow.

In addition, to ensure that SMU alarm ALM7[0] does not correspond to a real SRAM correctable error, the user may refer to the ESM MCU_FW_CHECK described in the Safety Manual.

OCDS_TC.H014 Avoiding failure of key exchange command due to overwrite of COMDATA by firmware

Note: This problem is only relevant for tool development, not for application development.

After PORST the UNIQUE_CHIP_ID_32BIT is written to the COMDATA register by firmware (time point T1). Then, firmware evaluates whether a key exchange request (CMD_KEY_EXCHANGE) is contained inside of the COMDATA register at a time point (T2). If yes, firmware will expect the 8 further words (password) from the COMDATA. If no, firmware will write again the UNIQUE_CHIP_ID_32BIT value for external tools to identify the device.

If the key exchange request cannot arrive between time points T1 and T2, firmware will skip the unlock procedure and will not unlock the device. For example, the device is locked and the external tool writes the CMD_KEY_EXCHANGE value to COMDATA before T1. Then, this value is overwritten by firmware at T1. After this, firmware doesn't see the CMD_KEY_EXCHANGE value and skips the unlock procedure. The device stays locked.

Recommendation

The external tool shall write the CMD_KEY_EXCHANGE to the COMDATA register between T1 and T2. As different derivatives and firmware configurations may have different execution time, it is recommended to poll the content of COMDATA after PORST until the UNIQUE_CHIP_ID_32BIT is available. Then, the external tool shall write the CMD_KEY_EXCHANGE immediately. In this way, the overwrite of key exchange request by firmware can be avoided.

When LBIST is activated during startup, the execution time stays the same after the PORST triggered by LBIST. Therefore, the end of LBIST should be detected by the external tool. This can be achieved by polling the device state via JTAG/DAP. During LBIST, the debug interface is disabled and no response can be received. After LBIST, the response can be received normally. This symptom can be utilized to determine whether LBIST is done. The details are described in the section "Halt after PORST with DAP" in the OCDS chapter of the device documentation.

OCDS_TC.H015 System or Application Reset while OCDS and lockstep monitoring are enabled

After a System or Application Reset the Lockstep Alarm ALMx[0] gets activated if all of the following conditions are met (x = index of CPU with checker core):

1. Lockstep monitoring is enabled by BMI.LSENAx = 1_B for CPUx, AND
2. Debug System is enabled (CBS_OSTATE.OEN = 1_B), AND

3. CPUx is halted (either in boot-halt state or stopped by debugger tool or in idle mode) when reset is triggered.

Recommendation

To avoid the unintended ALMx[0] under the conditions described above, either:

- Keep the debug system disabled. OR
- Ensure all CPUs that have lockstep monitoring enabled are out of halted state before executing a System or Application reset. OR
- Use PORST instead of a System or Application reset.

<u>OCDS_TC.H016</u> Release of application reset via OJCONF may fail

Note: This problem is only relevant for tool development, not for application development.

The OJCONF.OJC7 bit field can be used to send an application reset request to the SCU. The tool sets the bit to request an application reset and has to clear the bit to release the request otherwise the device will remain in reset state.

If JTAG is used in the above case and the frequency of JTAG is very low, there is a risk that the tool is not able to release the application reset request. If DAP is used, there is a low risk that the first release of reset request may fail but the second will always work.

Recommendation

It is recommended to run JTAG above 1 MHz and execute the following instructions back to back:

IO_SUPERVISOR + IO_SET_OJCONF (release) + IO_SUPERVISOR + IO_SET_OJCONF (release).

This double releasing ensures that the reset request is released reliably.

SCR_TC.H009 RAM ECC Alarms in Standby Mode

During Standby mode, every ECC error in the RAMs of the Standby Controller (SCR) can be detected but the respective alarm signal is not propagated and not triggered by the SMU (ALM6[19], ALM6[20] and ALM6[21]).

Note: If not in Standby mode, alarm signals for ECC errors from the SCR RAMs are propagated and triggered by the SMU.

Recommendation

ECC errors from the RAMs of SCR can be checked by the application software via bit SCRECC of PMS register PMSWCR2 (Standby and Wake-up Control Register).

SCR_TC.H010 HRESET command erroneously sets RRF flag

Note: This problem is only relevant for tool development, not for application development.

The HRESET command (to reset the SCR including its OCDS) erroneously sets the RRF flag (which signals received data to the FW).

Recommendation

With the following three additional commands (a-c) after an HRESET, the issues with the HRESET command can be solved:

- Execute HRESET
  a) Execute HSTATE to remove reset bit from shift register.
  b) Perform JTAG tool reset to remove flag RRF (receive register flag).
  c) Execute HCOMRST to remove flag TRF (transmit register flag).

SCR_TC.H011 Hang-up when warm PORST is activated during Debug Monitor Mode

Note: This problem is only relevant for debugging.

When a debugger is connected and the device is in Monitor Mode (MMODE), the activation of a warm PORST will result in a hang-up of the SCR controller.

Recommendation

Perform an LVD reset (power off/on) to terminate this situation.

SCR_TC.H012 Reaction in case of XRAM ECC Error

When the double-bit ECC reset is enabled via bit ECCRSTEN in register SCR_RSTCON, and a RAM double-bit ECC error is detected, bit RSTST.ECCRST in register SCR_RSTST is set, but no reset is performed.

Recommendation

The reset of the SCR module in case of a double-bit ECC error must be performed via software.

The following steps need to be done:

- Enable the double-bit ECC reset by setting bit ECCRSTEN in register SCR_RSTCON to 1_B.
- Enable the RAM ECC Error for NMI generation by setting bit NMIRAMECC in register SCR_NMICON to 1_B.

When a RAM double-bit ECC error is detected, an NMI to the TriCore is generated, and bit RSTST.ECCRST in register SCR_RSTST is set.

The TriCore software first has to check the cause of the NMI wakeup by checking register SCR_RSTST. If bit ECCRST is set, a double-bit ECC error has occurred. In this case, do the following steps:

- Fill the XRAM memory with 0.
- Check whether an ECC error has occurred.
- If no ECC error has occurred after filling the XRAM with 0, then:
  - Reload the contents of the XRAM.
  - Perform a reset of the SCR module: Set bit SCRSTREQ in register PMSWCR4 to 1_B.

<u>SDMMC_TC.H001</u> Idle State of SDMMC0_CLK

The idle level of the card clock output SDMMC0_CLK is high while the card clock is not enabled (bit CLK_CTRL.SD_CLK_EN = 0_B).

<u>SMU_TC.H010</u> Clearing individual SMU flags: use only 32-bit writes

The SMU registers shall only be written via 32-bit word accesses (i.e. ST.W instruction), as mentioned in table “Registers Overview” of the SMU chapter in the User’s Manual.

If any other instruction such as LDMST or SWAPMSK.W is used to modify only a few bits in the 32-bit register, then this may have the effect of modifying/clearing unintended bits.

Recommendation (Examples in C Language)

- **Example 1**: To clear status flag SF2 in register AG0, use:
  - SMU_AG0.U = 0x0000 0004;
- **Example 2**: To clear status flags EF2 in register RMEF and RMSTS, use:
  - SMU_RMEF.U = 0xFFFF FFFB;
  - SMU_RMSTS.U = 0xFFFF FFFB;

Here the <REGISTER>.U implies writing to the register as an unsigned integer, which normally results in a compiler translation into an ST.W instruction.

Safety Considerations

As long as software uses only 32-bit writes to the SMU registers, there is no risk of malfunction.

In case the software does not use 32-bit writes (and for example uses bit-wise operations such as LDMST instructions instead) – then potentially unintended flags may be written and modified in the SMU registers. Depending on the application, this may potentially have an impact on safety and/or diagnostics.

Note: The SMU reaction itself (e.g. alarm action triggering) is not affected even if the software unintentionally clears additional bits by not using a 32-bit write as recommended.

SMU_TC.H012 Handling of SMU alarms ALM7[1] and ALM7[0]

The FSI RAM is used to configure the PFLASH. For security related reason, the access to this RAM is restricted. Therefore, in order to avoid accesses to this RAM through its SSH, the MBIST Controller 40 is not disclosed in the AURIX TC3xx Target Specification/User's Manual.

However, the SMU alarms ALM7[1] and ALM7[0] are set intentionally after PORST and system reset and shall be cleared by the application SW (cf. ESM[SW]:SYS:MCU_FW_CHECK in Safety Manual v1.0).

Also, in order to clear the SMU alarms ALM7[1] and ALM7[0], it is necessary to clear the alarms within this MC40.

Recommendation

Therefore, the register addresses listed below have to be written as follows:

0xF00638F0 = (16-bit write) 0x0

0xF0063810 = (16-bit write) 0x0

SRI_TC.H001 Using LDMST and SWAPMSK.W instructions on SRI mapped Peripheral Registers (range 0xF800 0000-0xFFFF FFFF)

The LDMST and SWAPMSK.W instructions in the AURIX™ microcontrollers are intended to provide atomicity as well as bit-wise operations to a targeted memory location or peripheral register. They are also referred to as Read-Modify-Write (RMW) instructions.

The bit-manipulation functionality is intended to provide software a mechanism to write to individual bits in a register, without affecting other bits. The bits to be written can be selected via a mask in the instruction. Please refer to the TriCore Architecture Manual for further information about these instructions and their formats.

Restrictions for SRI mapped Peripherals

The bit-manipulation functionality is supported only on registers accessed via the SPB bus, and is not supported on the SRI mapped peripheral range (i.e. address range 0xF800 0000 to 0xFFFF FFFF, including (if available) DMU, LMU, EBU, DAM, SRI Crossbar, SPU, CPUx SFRs and CSFRs, AGBT, miniMCDS, ...); see table “On Chip Bus Address Map of Segment 15” in chapter “Memory Map”).

On the SRI mapped peripherals, usage of these instructions always results in all the bits of a register being written, and not just specific individual bits.

Note: The instructions are still executed atomically on the bus – i.e the SRI is locked between the READ and the WRITE transaction.