Errata Sheet

Rel. 1.0, 2013-09-24

Device	**TC1724F/N, TC1728F/N**
Marking/Step	**ES-AC, AC**
Package	**see Data Sheet**

02807AERRA

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

Table 1 Current Documentation[1)]

TC1724 Data Sheet	V1.1	2012-03
TC1728 Data Sheet	V1.1	2013-09
TC1728 User's Manual	V1.0	2011-12
TriCore 1 Architecture	V1.3.8	January 2008

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

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/32-bit-microcontrollers** 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 derivative synopsis, see the latest Data Sheet/User's Manual.
   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 "FlexRay" do not apply to devices where no ERAY module is specified.
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. This device is equipped with a TriCore "TC1.3.1" core. Some of the errata have workarounds which are possibly supported by the tool vendors. Some corresponding compiler switches need possibly to be set. Please see the respective documentation of your compiler.
   For effects of issues related to the on-chip debug system, see also the documentation of the debug tool vendor.

1 History List / Change Summary

Table 2 History List

Version	Date	Remark
1.0	2013-09-24	

Note: Changes to the previous errata sheet version TC172x_AB_Errata_Sheet_v1_1 are particularly marked in column "Change" in the following tables.

Table 3 Errata fixed in this step

Errata	Short Description	Change
ADC_TC.H009	Initialisation of analog channels where both ADC & FADC functions are overlapped	Fixed
FADC_TC.H002	Initialisation of analog channels where both ADC & FADC functions are overlapped	Fixed

Table 4 Functional Deviations

Functional Deviation	Short Description	Change	Page
BCU_TC.007	Access to certain addresses in reserved area F000 2800$_H$ - F000 2FFF$_H$		10
BROM_TC.006	Baud Rate Detection for CAN Bootstrap Loader		10
CPU_TC.111	Imprecise Return Address for FCU Trap		11

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
CPU_TC.114	CAE Trap may be generated by UPDFL instruction		11
CPU_TC.117	Cached Store Data Lost on Data Cache Invalidate via Overlay		12
DMI_TC.016	CPU Deadlock possible when Cacheable access encounters Flash Double-Bit Error		14
FADC_TC.005	Equidistant multiple channel-timers		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
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		19
FlexRay_AI.090	Flag SFS.MRCS is set erroneously although at least one valid sync frame pair is received		20
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		21
FlexRay_AI.093	Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames		22
FlexRay_AI.094	Sync frame overflow flag EIR.SFO may be set if slot counter is greater than 1024		22

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
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		24
FlexRay_AI.097	Loop back mode operates only at 10 MBit/s		25
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		27
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	New	28
MSC_TC.009	Missing Receive Data Interrupt		28
MultiCAN_TC.041	Clock f_{CLC} used in Bit Timing Mode		29
MultiCAN_TC.042	CAN-to-DMA Request Connections	New	29
OCDS_TC.025	PC corruption when entering Halt mode after a MTCR to DBGSR		30
OCDS_TC.026	PSW.PRS updated too late after a RFM instruction.		31
OCDS_TC.027	BAM breakpoints with associated halt action can potentially corrupt the PC.		32
OCDS_TC.028	Accesses to CSFR and GPR registers of running program can corrupt loop exits.		33
OCDS_TC.035	Debug reset will not disable the OCDS		34
PCP_TC.023	JUMP sometimes takes an extra cycle		34
PCP_TC.032	Incorrect PCP behaviour following FPI timeouts (as a slave)		35

Table 4 Functional Deviations (cont'd)

Functional Deviation	Short Description	Change	Page
PCP_TC.041	Coincident PCP non-atomic Write and FPI RMW Access to PRAM		35
SSC_AI.022	Phase error detection switched off too early at the end of a transmission		36
SSC_AI.023	Clock phase control causes failing data transmission in slave mode		36
SSC_AI.024	SLSO output gets stuck if a reconfig from slave to master mode happens		37
SSC_AI.025	First shift clock period will be one PLL clock too short because not syncronized to baudrate		37
SSC_AI.026	Master with highest baud rate set generates erroneous phase error		38

Table 5 Application Hints

Hint	Short Description	Change	Page
ADC_AI.H002	Minimizing Power Consumption of an ADC Module		40
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion		40
BROM_TC.H002	Enabling CAN Communication in CAN Bootstrap Loader Mode		42
CPU_TC.H004	PCXI Handling Differences in TriCore1.3.1		42
CPU_TC.H005	Wake-up from Idle/Sleep Mode		44
FIRM_TC.H000	Reading the Flash Microcode Version		44

Table 5 Application Hints (cont'd)

Hint	Short Description	Change	Page
FlexRay_AI.H004	Only the first message can be received in External Loop Back mode		45
FlexRay_AI.H005	Initialization of internal RAMs requires one eray_bclk cycle more		45
FlexRay_AI.H006	Transmission in ATM/Loopback mode		46
FlexRay_AI.H007	Reporting of coding errors via TEST1.CERA/B		46
FlexRay_AI.H009	Return from test mode operation		46
FlexRay_AI.H010	Driver SW must launch CLEAR_RAMS command before reading from E-Ray RAMs		47
FlexRay_TC.H001	RAM Error Indicators need to be cleared at Module Init	New	47
FPI_TC.H001	FPI bus may be monopolized despite starvation protection		48
GPT12_AI.H001	Modification of Block Prescalers BPS1 and BPS2		48
GPTA_TC.H004	Handling of GPTA Service Requests		49
MSC_TC.H007	Start Condition for Upstream Channel		52
MSC_TC.H010	Configuration of SCU.EMSR for the EMGSTOPMSC Signal	New	53
MultiCAN_AI.H005	TxD Pulse upon short disable request		53
MultiCAN_AI.H006	Time stamp influenced by resynchronization		54
MultiCAN_AI.H007	Alert Interrupt Behavior in case of Bus-Off		54
MultiCAN_AI.H008	Effect of CANDIS on SUSACK		55
MultiCAN_TC.H002	Double Synchronization of receive input		55

Table 5 Application Hints (cont'd)

Hint	Short Description	Change	Page
MultiCAN_TC.H003	Message may be discarded before transmission in STT mode		55
MultiCAN_TC.H004	Double remote request		56
OCDS_TC.H001	IOADDR may increment after aborted IO_READ_BLOCK		56
OCDS_TC.H002	Setting IOSR.CRSYNC during Application Reset		57
OCDS_TC.H003	Application Reset during host communication		57
OCDS_TC.H007	Early Acknowledgement of Channel Suspend for Active DMA Channel		58
PWR_TC.H008	Internal pull-up device on PORST pad		59
SSC_AI.H002	Transmit Buffer Update in Master Mode during Trailing or Inactive Delay Phase		60
SSC_AI.H003	Transmit Buffer Update in Slave Mode during Transmission		61
SSC_TC.H003	Handling of Flag STAT.BSY in Master Mode		61

Table 6 Documentation Updates

Functional Deviation	Short Description	Change	Page
CCU6_TC.D001	Register CMPMODIF	New	63
DMA_TC.D001	Register DMA_CLRE	New	63
GPTA_TC.D001	Special Function Registers, Port Assignment	New	63

Table 6 Documentation Updates (cont'd)

Functional Deviation	Short Description	Change	Page
MultiCAN_TC.D001	MultiCAN I/O Control Selection and Setup	New	64

2 Functional Deviations

BCU_TC.007 Access to certain addresses in reserved area F000 2800_H - F000 $2FFF_H$

The address range from F000 2800_H to F000 $2FFF_H$ is specified as reserved, and accesses to this range should result in a bus error.

However, no bus error is generated by the SBCU (System Peripheral Bus Control Unit) upon an access comprising the following word addresses in this area:

F000 2800_H

F000 $29F8_H$

F000 $2AF4_H$

F000 $2BF0_H$

F000 $2FE0_H$ .. F000 $2FFC_H$

Workaround

None.

BROM_TC.006 Baud Rate Detection for CAN Bootstrap Loader

In a specific corner case, the baud rate detected during reception of the initialization frame for the CAN bootstrap loader may be incorrect. The probability for this sporadic problem is relatively low, and it decreases with decreasing CAN baud rate and increasing module clock frequency.

Workaround:

If communication fails, the host should repeat the CAN bootstrap loader initialization procedure after a reset of the device.

CPU_TC.111 Imprecise Return Address for FCU Trap

The FCU trap is taken when a context save operation is attempted but the free context list is found to be empty, or when an error is encountered during a context save or restore operation. In failing to complete the context operation, architectural state is lost, so the occurrence of an FCU trap is a non-recoverable system error.

Since FCU traps are non-recoverable system errors, having a precise return address is not important, but can be useful in establishing the cause of the FCU trap. The TriCore1 CPU does not generate a precise return address for an FCU trap if the cause of the FCU trap was one of the following trap types: FCD, DAE, DIE, CAE or NMI.

In each of these circumstances the return address may be invalid.

Workaround

None

CPU_TC.114 CAE Trap may be generated by UPDFL instruction

UPDFL is a User mode instruction implemented as part of the TriCore1 Floating-Point Unit (FPU), which allows individual bits of the PSW user status bits, PSW[31:24], to be set or cleared. Contrary to early revisions of the TriCore1.3.1 architecture manual, and in contrast to most other FPU instructions, the UPDFL instruction should not generate Co-Processor Asynchronous Error (CAE) traps. However, in certain circumstances the TriCore1.3.1 FPU will generate CAE traps for UPDFL instructions.

The TriCore1.3.1 FPU will generate a CAE trap upon execution of the UPDFL instruction in the following situation:

- After execution of the UPDFL instruction, one or more of the PSW[31:26] bits are set - either the PSW bit(s) are set by UPDFL or were set prior to execution and not cleared by the UPDFL instruction.
- FPU traps are enabled for one of the asserted PSW[31:26] bits, via the corresponding FPU_TRAP_CON.FxE bit being set.

- The FPU_TRAP_CON.TST CSFR bit is clear - no previous FPU trap has been generated without the subsequent clearing of FPU_TRAP_CON.TST.

Workaround

The UPDFL instruction is normally used in one of two situations:

- Clearing the FPU sticky flags held in PSW[30:26].
- Setting the FPU rounding mode bits in PSW[25:24].

In the first case, if all the PSW[31:26] bits are cleared by UPDFL, no CAE trap will be generated.

In the second case, UPDFL may still be used to set the FPU rounding mode, but in this case the remaining PSW bits, [31:26], must be cleared by UPDFL in order to avoid generation of an unexpected CAE trap.

In all other cases, where FPU traps are enabled, some other method of manipulating the PSW user status bits must be used in order to avoid extraneous CAE trap generation. For instance, if in Supervisor mode the PSW may be read using the MFCR instruction, the high order PSW bits modified and written back using the MTCR instruction.

CPU_TC.117 Cached Store Data Lost on Data Cache Invalidate via Overlay

Cached store data can be lost if the overlay system requests a data cache invalidate in the same cycle as a cache line is being written. The overlay control provides a mechanism to do a single cycle invalidate of all valid/clean lines in the data cache by writing the OCON.DCINVAL bit. Please note that there is no problem if the data cache is used exclusively for read data (e.g. flash constants).

```
Cache line state transition on DCINVAL.
   valid/clean -> (DCINVAL) -> invalid/clean
```

A normal store operation transitions the cache line to a valid/dirty state.

```
Cache line state transition on normal store operation.
    valid/clean -> (write) -> valid/dirty
    invalid/clean -> (write) -> valid/dirty
```

In the case where the write and invalidate are received in the same cycle, the dirty bit is correctly updated but the valid bit is incorrectly cleared.

```
Cache line state transition on store operation with DCINVAL
     valid/clean -> (write+DCINVAL)- > invalid/dirty
     invalid/clean -> (write+DCINVAL) -> invalid/dirty
```

This leads to a loss of data as the store data ends up being held in an invalid cache line and hence never re-read.

Workaround-1

Ensure that the data cache is never used to cache write data. This can be ensured by software design but may limit performance in some systems.

Workaround-2

Ensure that the core is never storing data when OCON.DCINVAL is asserted.

This requires the CPUs store buffers to be empty when the invalidate is asserted. This can only be done by getting the CPU to firstly flush all write data with a DSYNC command, then to write the OCON.DCINVAL to trigger an invalidate.

The following example code sequence performs the required operations:-

- Read the OCON register to get the current SHOVEN field
- Create a new OCON value with DCINVAL, OVSTRT and OVCONF bits set
- Perform a DSYNC operation to flush all write data to memory
- Write OCON with the new value.
- Read back OCON to ensure write is complete

```
;; Set up A14 with address of OCON Register
movh.a  a14,#(((0xF87FFBE0)+0x8000>>16) & 0xffff)lea
a14,[a14]((((0xF87FFBE0)+0x8000)&0xffff)-0x8000)
;; Load a15 with contents of OCON
ld.w    d15, [a14]
;; Set OCONF, DCINVAL, OVSTRT start values
movh    d14  , #0x0305
```

```
;; Combine existing SHOVEN
insert d15, d14,d15,#0,#16
; Flush all store data
dsync
;; Store New value back to OCON
st.w   [a14], d15
;; Re-read to ensure store is complete
ld.w   d15, [a14]
```

Attention: This routine must be run with interrupts disabled, either as part of an interrupt service routine or guarded by enable/disable instructions.

This routine may be run periodically or run as part of a dedicated interrupt service routine. If the latter approach is used it is suggested that an unused SRN either in the CPU or Cerberus is utilised to trigger the invalidate. In all cases the routine must be run with interrupts disabled to ensure that no writes are in progress when the invalidate occurs.

The OCON.OVCONF bit may be used to indicate the state of the invalidate operation. If it is cleared in advance, the routine above will set it when the cache invalidate operation is performed.

DMI_TC.016 CPU Deadlock possible when Cacheable access encounters Flash Double-Bit Error

A problem exists whereby the TriCore CPU may become deadlocked when attempting a mis-aligned load access to a cacheable address. The problem will be triggered in the following situation:

- The TriCore CPU executes a load instruction whose target address is not naturally aligned - a data word access which targets an address which is not word aligned, or a data / address double-word access which is not double-word aligned.
- The mis-aligned load access targets a cacheable address, whether the device is configured with a data cache or not.

- The mis-aligned load access spans two halves of the same 128-bit cache line. For instance, a data word access with address offset 6_H.
- The mis-aligned load access results in a cache miss, which will refill the 128-bit cache line / Data Line Buffer (DLB) via a Block Transfer 2 (BTR2) read transaction on the LMB, and this LMB read encounters a bus error condition in **the second beat of the block transfer**.

It should be noted that under normal operation, LMB block transfers will not result in a bus error condition being flagged on the second beat of a block transfer. However, such a condition may be encountered when accessing the on-chip Flash, if the second double-word of data accessed from the Flash (for the second half of the cache line) contains an uncorrectable double-bit error.

When this condition is triggered, the first part of the requested data is obtained from the valid first beat of the BTR2 transfer, and the second part is required from the errored second beat. In this case, no error is flagged to the TriCore CPU and the transaction is incorrectly re-started on the LMB. In the case of a Flash double-bit error, this transaction will be re-tried continuously on the LMB by the DMI LMB master and the CPU become deadlocked. This situation would then only be recoverable by a Watchdog reset.

The problem exists within the DMI DLB, which is used as a single cache line when no data cache is configured, and as a streaming buffer when data cache is present. As such the problem affects all load accesses to cacheable locations, whether data cache is configured or not, since the DLB is used in both cases.

Note: This problem affects load accesses to the on-chip Flash only. Instruction fetches which encounter a similar condition (bus error on later beat of block transfer) behave as expected and will return a PSE trap upon any attempt to execute an instruction from a Flash location containing a double-bit error.

Workaround

As described previously, this problem should not be encountered during normal operation and will only be triggered in the case of a double-bit error being detected in an access to the on-chip Flash.

However, in order to remove the possibility of encountering this issue, all load accesses to cacheable addresses within the on-chip Flash should be made

using natural alignment - word transfers should be word aligned, double-word transfers double-word aligned.

It is also possible to check for the occurrence of this problem by having some other master, such as the PCP, periodically poll the LBCU `LEATT` register to check for the occurrence of LMB error conditions, specifically if one is detected during a BTR2 read transfer from the DMI, as reported by `LEATT.OPC` and `LEATT.TAG`.

FADC_TC.005 Equidistant multiple channel-timers

The description is an example for timer_1 and timer_2, but can also affect all other combinations of timers.

Timer_1 and Timer_2 are running with different reload-values. Both timers should start conversions with the requirement of equidistant timing.

Problem description:

Timer_1 becomes zero and starts a conversion. Timer_2 becomes zero during this conversion is running and sets the conversion-request-bit of channel_2. At the end of the conversion for channel_1 this request initiates a start for channel_2. But the Timer_2 is reloaded only when setting the request-bit for channel_2 and is decremented during the conversion of channel_1.

The correct behavior would be a reload when the requested conversion (of channel_2) is started.

Therefore the start of conversion for channel_2 is delayed by maximum one conversion-time. After this delay it will be continued with equidistant conversion-starts. Please refer to the following figure.

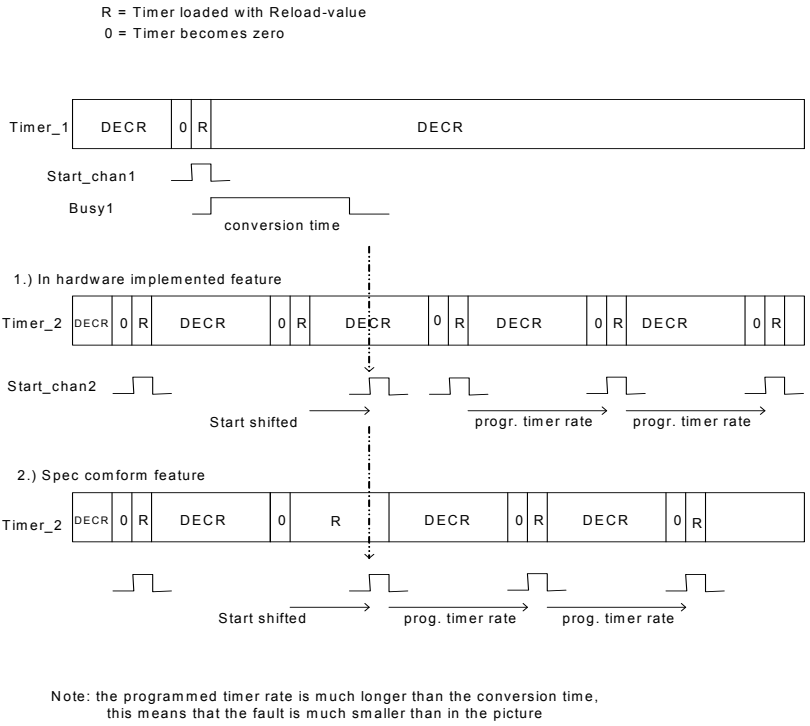


Figure 1 Timing concerning equidistant multiple timers

Workaround

Use one timer base in combination with neighboring trigger and selection by software which result has to be taken into account.

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.

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

Description:

If in the static segment the number of transmitted and received sync frames reaches gSyncNodeMax and the slot counter in the dynamic segment reaches

the value cStaticSlotIDMax + gSyncNodeMax = 1023 + gSyncNodeMax, the sync frame overflow flag `EIR.SFO` is set erroneously.

Scope:

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

Effects:

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

Workaround

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

FlexRay_AI.095 Register RCV displays wrong value

Description:

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

Scope:

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

Effects:

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

Workaround

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

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

Description:

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

Scope:

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

Effects:

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

Workaround

None.

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

Description:

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

Scope:

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

Effects:

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

Workaround

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

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

Description:

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

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

MSC_TC.009 Missing Receive Data Interrupt

A problem with receive data interrupt generation on the upstream channel occurs in a specific corner case if all of the following three conditions are met at the same time:

1. Option `ICR.RDIE` = 10_B is selected as interrupt generation condition (i.e. interrupt only if received data is not equal to 00_H), and

2. Two MSC frames (F_n, F_{n+1}) are transmitted on the upstream channel in series (i.e. after two stop bits immediately a start bit occurs), and
3. The leading edge of the start bit generated by the transmitter arrives at input SDI of the microcontroller before the end or at the boundary of its MSC stop bit cycle. This is typically the case when the transmitter uses a clock that is independent of the microcontroller clock, and the actual baud rate of the transmitter is higher or equal to the configured baud rate of the microcontroller (within the permitted range for asynchronous transfers).

In this case, the interrupt request at the end of frame F_n will not be generated and flag `ISR.URDI` will not be set.

Workarounds

1. Use a frame with a third stop bit.
2. Do not use the interrupt generation condition `ICR.RDIE` = 10_B (depending on data bits of received frame). Use e.g. `ICR.RDIE` = 01_B, and test for received data equal to zero by software.

MultiCAN_TC.041 Clock f_{CLC} used in Bit Timing Mode

Unlike described in some parts of the documentation, in Bit Timing Mode f_{CLC} is used instead of f_{CAN}. This means that the time information stored in bitfield NFCRx.CFC is measured in f_{CLC} clock cycles.

Workaround

When Bit Timing Mode is used, configure bitfield MCR.CLKSEL = 0001_B such that $f_{CAN} = f_{CLC}$.

MultiCAN_TC.042 CAN-to-DMA Request Connections

The CAN-to-DMA request assignments listed in table “CAN-to-DMA Request Connections” of chapter “Controller Area Network (MultiCAN) Controller” in the current User’s Manual version are incorrect.

For a correct description see table “DMA Request Assignment for DMA Sub-Block 0” and “DMA Request Assignment for DMA Sub-Block 1” (for devices with two DMA sub-blocks) in chapter “Direct Memory Access Controller.

OCDS_TC.025 PC corruption when entering Halt mode after a MTCR to DBGSR

In cases where the CPU is forced into HALT mode by a MTCR instruction to the DBGSR register, there is a possibility of PC corruption just before HALT mode is entered. This can happen for MTCR instructions injected via the CPS as well as for user program MTCR instructions being fetched by the CPU. In both cases the PC is potentially corrupted before entering HALT mode. Any subsequent read of the PC during HALT will yield an erroneous value. Moreover, on exiting HALT mode the CPU will resume execution from an erroneous location. .

The corruption occurs when the MTCR instruction is immediately followed by a mis-predicted LS branch or loop instruction. The forcing of the CPU into HALT takes priority over the branch resolution and the PC will erroneously be assigned the mispredicted target address before going into HALT.

- Problem sequence 1:
- 1) CPS-injected MTCR instruction to DBGSR sets HALT Mode
- 2) LS-based branch/loop instruction
- 3) LS-based branch/loop is mispredicted but resolution is overridden by HALT.
- Problem sequence 2:
- 1) User code MTCR instruction to DBGSR sets HALT Mode
- 2) LS-based branch/loop instruction
- 3) LS-based branch/loop is mispredicted but resolution is overridden by HALT.

Workaround

External agents should halt the CPU using the BRKIN pin instead of using CPS injected writes to the CSFR register. Alternatively, the CPU can always be halted by using the debug breakpoints. Any user software write to the DBGSR CSFR should be followed by a dsync.

OCDS_TC.026 PSW.PRS updated too late after a RFM instruction.

When a breakpoint with an associated TRAP action occurs, the Tricore will enter a special trap called a ’debug monitor’. The RFM instruction (return from monitor) is used to return from the debug monitor trap. After the RFM, the CPU should resume execution at the point where it left it when the breakpoint happened.On execution of the RFM instruction, a light-weight debug context is restored and the PSW CSFR is loaded with its new value. The updated value of the PSW.PRS field should then be used to select the appropriate protection register set for all subsequently fetched instructions. Because PSW.PRS can be updated too late after an RFM instruction, the instruction following an RFM potentially sees the old value of the PSW.PRS field as opposed to the new one.This can be problematic since the PSW.PRS field is crucial in terms of code protection and debug. Indeed there is a possibility that the instruction immediately following the RFM be submitted to inadequate protection rules (as defined by the old PSW.PRS field).

- Problem sequence:
- instr (monitor)
- instr (monitor)
- instr (monitor)
- RFM (monitor)
- Instruction1 // Uses debug monitor’s PSW.PRS field as opposed to newly restored one.
- instruction2

Workaround

To fix this the user needs to do the following before exiting the monitor using RFM:

.

- > Retrieve the old value of PSW from location DCX+4
- > Do a MFCR and a MTCR to copy the old value of PSW.PRS into PSW without changing other PSW fields.
- > DSYNC
- > RFM

This sequence will guarantee that all instructions fetched subsequently to the RFM will be submitted to the new PSW.PRS field.

OCDS_TC.027 BAM breakpoints with associated halt action can potentially corrupt the PC.

BAM breakpoints can be programmed to trigger a halt action. When such a breakpoint is taken the CPU will go into HALT mode immediately after the instruction is executed. This mechanism is broken in the case of conditional jumps. When a BAM breakpoint with halt action is triggered on a conditional jump, the PC for the next instruction will potentially be corrupted before the CPU goes into HALT mode. On exiting HALT mode the CPU will see the corrupted value of the PC and hence resume code execution from an erroneous location. Reading the PC CSFR whilst in HALT mode will also yield a faulty value.

Workaround

In order to avoid PC corruption the user should avoid placing BAM breakpoints with HALT action on random code which could contain conditional jumps.The simplest thing to do is to avoid BAM breakpoints with HALT action altogether. A combination of BBM breakpoints and other types of breakpoint actions can be used to achieve the desired functionality.:

Workaround for single-stepping:

An 'intuitive' way of implementing single-stepping mode is to place a halt-action BAM breakpoint on the address range from 0x00000000 to 0xFFFFFFFF. Every time the CPU is woken up via the CERBERUS it will execute the next instruction and go back to HALT mode. Unfortunately this will trigger the bug described by the current ERRATA.

The solution is to implement single-stepping using BBM breakpoints:

- 1) Create two debug trigger ranges:
- First range: 0x00000000 to current_instruction_pc (not included)
- Second range: current_instuction_pc (not included) to 0xFFFFFFFF
- 2) Associate the two debug ranges with BBM breakpoints.
- 3) Associate the BBM breakpoints with a HALT action.
- 4) Wake up the CPU via CERBERUS

- 5) CPU will execute the next instruction, update the PC and go to HALT mode.
- 6) Start again (go back to 1)

OCDS_TC.028 Accesses to CSFR and GPR registers of running program can corrupt loop exits.

Overview:

A hardware problem has been identified whereby FPI accesses to the [0xF7E10000 : 0xF7E1FFFF] region will potentially corrupt the functionality of the Tricore LOOP instruction. This is particularly relevant because the Tricore CPU CSFR and GPR registers are mapped to that region. So any access to those registers by an external agent will potentially cause the LOOP instruction not to work. Note that this problem will not happen if the CPU was halted at the time of the FPI access.

Typical bug behaviour:

The loop instruction should exit (fall through) when its loop count operand is zero. The identified problem will typically cause the loop instruction to underflow: instead of exiting when its loop count operand is zero, the loop instruction will erroneously jump back to its target with a -1 (0xFFFFFFFF) loop counter value, and then continue to iterate possibly ad infinitum. Note that the offending FPI access will not cause the bug to happen immediately but only when the loop instruction finally tries to exit.

Influencing factors:

The following factors influence the likelihood of the bug happening:

1) The bug will not happen if the LOOP instruction and its predecessor are both entirely contained in the same aligned 8-byte word.

2) The bug is much less likely to happen if the CPU is running from program cache or program scratchpad.

3) The problem will be more visible on later compiler versions which make a more intensive use of the loop instruction.

Workaround:

The workaround consists in preventing all FPI agents from accessing the [0xF7E10000 : 0xF7E1FFFF] region when the CPU is not halted.

This means that the CPU CSFR and GPR registers can't be accessed on-the-fly whilst the CPU is running. This is particularly relevant for debug tool providers who may be polling those registers as the application is running. Note that accessing FPI addresses outside of the [0xF7E10000 : 0xF7E1FFFF] region will not cause the problem to happen.

An Application Note for tool partners, describing an alternative, more complex workaround for register access within the critical region by an external tool, is available from Infineon.

OCDS_TC.035 Debug reset will not disable the OCDS

Debug reset will not clear CBS_OSTATE.OEN so OCDS stays enabled. DBGSR.SUSP is not cleared as well.

Workaround

Disable the OCDS if needed after Debug reset by writing CBS_OEC.DS. Write reset value to DBGSR e.g. to start the CPU.

PCP_TC.023 JUMP sometimes takes an extra cycle

Following a taken JUMP, the main state machine may misleadingly take an additional cycle of pause. This occurs if the already prefetched next or second next instruction after the JUMP is one of the following instructions:

- LD.P
- ST.P
- DEBUG
- Any instruction with extension .PI

This does not cause any different program flow or incorrect result, it just adds an extra dead cycle.

Workaround

None.

PCP_TC.032 Incorrect PCP behaviour following FPI timeouts (as a slave)

When PRAM is being accessed from the FPI bus and an FPI time-out occurs then this can lead to corruption or loss of the current and subsequent FPI accesses. In general an FPI time-out during an access to the PCP is unlikely since FPI time-out is usually programmed for a large number of FPI clock cycles and the only time that the FPI access cannot be immediately responded to by the PCP is during the execution of atomic PRAM instructions. FPI accesses are locked out for the entire duration of any sequence of back to back atomic PRAM instructions. The combination of a low FPI time-out setting and long sequences of atomic PRAM instructions could therefore result in FPI time-out.

Workaround

Keep the FPI time-out setting as high as possible and do not include long sequences of back to back atomic PRAM instructions. If N is the highest amount of back to back atomic PRAM instructions in any PCP channel program, FPI time-out should at least be 10 times N.

PCP_TC.041 Coincident PCP non-atomic Write and FPI RMW Access to PRAM

A PCP non-atomic PRAM write access coinciding with an FPI read/modify/write (rmw) to PRAM issued by an external master will break the atomicity of the FPI rmw instruction.

PCP non-atomic writes include instructions ST.P, ST.PI and channel context save operations.

Note: All operations will work correctly, i.e. no lost stores etc., as long as the PRAM write addresses are different.

Workaround 1

Replace the non-atomic PRAM instructions with atomic ones (e.g. XCH.PI, or MCLR/MSET.PI for semaphore operations).

If backward compatibility with other devices of the AudoXY microcontroller families is an issue/requirement, then use the following workaround:

Workaround 2

In case the PCP needs to write to the same PRAM address as an external FPI rmw instruction (i.e. a semaphore operation), replace these non-atomic store PRAM instructions with FPI based store instructions such as ST.F, ST.FI.

Alternatively, variables shared by PCP and external masters may be located in RAM areas other than PRAM, such that software tools implicitly use the FPI based store instructions such as ST.F, ST.FI.

SSC_AI.022 Phase error detection switched off too early at the end of a transmission

The phase error detection will be switched off too early at the end of a transmission. If the phase error occurs at the last bit to be transmitted, the phase error is lost.

Workaround

Don’t use the phase error detection.

SSC_AI.023 Clock phase control causes failing data transmission in slave mode

If `SSC_CON.PH` = 1 and no leading delay is issued by the master, the data output of the slave will be corrupted. The reason is that the chip select of the master enables the data output of the slave. As long as the chip is inactive the slave data output is also inactive.

Workaround

A leading delay should be used by the master.

A second possibility would be to initialize the first bit to be sent to the same value as the content of `PISEL.STIP`.

SSC_AI.024 SLSO output gets stuck if a reconfig from slave to master mode happens

The slave select output SLSO gets stuck if the SSC will be re-configured from slave to master mode. The SLSO will not be deactivated and therefore not correct for the 1st transmission in master mode. After this 1st transmission the chip select will be deactivated and working correctly for the following transmissions.

Workaround

Ignore the 1st data transmission of the SSC when changed from slave to master mode.

SSC_AI.025 First shift clock period will be one PLL clock too short because not syncronized to baudrate

The first shift clock signal duration of the master is one PLL clock cycle shorter than it should be after a new transmit request happens at the end of the previous transmission. In this case the previous transmission had a trailing delay and an inactive delay.

Workaround

Use at least one leading delay in order to avoid this problem.

SSC_AI.026 Master with highest baud rate set generates erroneous phase error

If the SSC is in master mode, the highest baud rate is initialized and CON.PO = 1 and CON.PH = 0 there will be a phase error on the MRST line already on the shift edge and not on the latching edge of the shift clock.

- Phase error already at shift edge
  The master runs with baud rate zero. The internal clock is derived from the rising and the falling edge. If the baud rate is different from zero there is a gap between these pulses of these internal generated clocks.
  However, if the baud rate is zero there is no gap which causes that the edge detection is to slow for the "fast" changing input signal. This means that the input data is already in the first delay stage of the phase detection when the delayed shift clock reaches the condition for a phase error check. Therefore the phase error signal appears.
- Phase error pulse at the end of transmission
  The reason for this is the combination of point 1 and the fact that the end of the transmission is reached. Thus the bit counter SSCBC reaches zero and the phase error detection will be switched off.

Workaround

Don’t use a phase error in master mode if the baud rate register is programmed to zero (SSCBR = 0) which means that only the fractional divider is used.
Or program the baud rate register to a value different from zero (SSCBR > 0) when the phase error should be used in master mode.

3 Deviations from Electrical- and Timing Specification

No deviations currently known.

4 Application Hints

ADC_AI.H002 Minimizing Power Consumption of an ADC Module

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

Recommendation for Minimum Power Consumption:

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

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

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

ADC_AI.H003 Injected conversion may be performed with sample time of aborted conversion

For specific timing conditions and configuration parameters, a higher prioritized conversion c_i (including a synchronized request from another ADC kernel) in cancel-inject-repeat mode may erroneously be performed with the sample time parameters of the lower prioritized cancelled conversion c_c. This can lead to

1) Symbol used depends on product family: e.g. f_{ANA} is used in the documentation of devices of the AUDO-NextGeneration family.

wrong sample results (depending on the source impedance), and may also shift the starting point of following conversions.

The conditions for this behavior are as follows (all 3 conditions must be met):

1. **Sample Time setting**: injected conversion c_i and cancelled conversion c_c use different sample time settings, i.e. bit fields `STC` in the corresponding Input Class Registers `INPCRx` (for c_c) and `INPCRy` (for c_i) are programmed to different values.
2. **Timing condition**: conversion c_i starts during the first f_{ADCI} clock cycle of the sample phase of c_c.
3. **Configuration parameters**: the ratio between the analog clock f_{ADCI} and the arbiter speed is as follows:

   $$N_A > N_D*(N_{AR}+3),$$

   with

   a) N_A = ratio f_{ADC}/f_{ADCI} (N_A = 4 .. 63, as defined in bit field `DIVA`),

   b) N_D = ratio f_{ADC}/f_{ADCD} = number of f_{ADC} clock cycles per arbitration slot (N_D = 1 .. 4, as defined in bit field `DIVD`),

   c) N_{AR} = number of arbitration slots per arbitration round (N_{AR} = 4, 8, 16, or 20, as defined in bit field `ARBRND`).

All bit fields mentioned above are located in register `GLOBCTR`.

As can be seen from the formula above, a problem typically only occurs when the arbiter is running at maximum speed, and a divider $N_A > 7$ is selected to obtain f_{ADCI}.

Recommendation 1

Select the same sample time for injected conversions c_i and potentially cancelled conversions c_c, i.e. program all bit fields `STC` in the corresponding Input Class Registers `INPCRx` (for c_c) and `INPCRy` (for c_i) to the same value.

Recommendation 2

Select the parameters in register `GLOBCTR` according to the following relation:
$N_A \le N_D*(N_{AR}+3)$.

BROM_TC.H002 Enabling CAN Communication in CAN Bootstrap Loader Mode

After completion of the download in CAN bootstrap loader mode, the module clock f_{CLC} is disabled. Therefore, code executed after download in CAN bootstrap loader mode can not directly continue communication via the CAN interface. It first needs to initialize register CAN_CLC to enable f_{CLC}.

CPU_TC.H004 PCXI Handling Differences in TriCore1.3.1

The TriCore1.3.1 core implements the improved architecture definition detailed in the TriCore Architecture Manual V1.3.8. This architecture manual version continues the process of removing ambiguities in the description of context save and restore operations, a process started in Architecture Manual V1.3.6 (released October 2005).

Several previous inconsistencies regarding the updating of the PCXI and the storing of PCXI fields in the first word of a CSA are now removed.

- CALL has always placed the full PCXI into the CSA
- BISR has always placed the full PCXI into the CSA
- SVLCX has always placed the full PCXI into the CSA
- RET has always restored the full PCXI from the CSA
- RFE has always restored the full PCXI from the CSA

From the TriCore V1.3.8 architecture manuals onwards it is also made explicit that:

- CALL, BISR and SVLCX now explicitly update the PCXI.PCPN, PCXI.PIE, PCXI.UL, PCXI.PCXS and PCXI.PCXO fields after storing the previous PCXI contents to memory.
- RSLCX now restores the full PCXI from the CSA.

However, prior to the TriCore V1.3.6 architecture manual, and as implemented by the TriCore1.3 core, the following behaviour was present:

- BISR and SVLCX previously only updated the PCXI.UL, PCXI.PCXS and PCXI.PCXO fields after storing the previous PCXI contents to memory. PCXI.PCPN and PCXI.PIE were not updated.

- RSLCX previously restored only the PCXI.UL, PCXI.PCXS and PCXI.PCXO fields of the PCXI.

The main implication of this change is that the value held in the PCXI.PCPN and PCXI.PIE fields following a BISR, SVLCX or RSLCX instruction may be different between the TriCore1.3.1 and TriCore1.3 cores. If it is necessary to determine the priority number of an interrupted task after performing a BISR or SVLCX instruction, and before the corresponding RSLCX instruction, then either of the following access methods may be used.

Method #1

For applications where the time prior to execution of the BISR instruction is not critical, the priority number of the interrupted task may be read from the PCXI before execution of the BISR instruction.

```
...
mfcr    d15, #0xFE00
bisr    #<New Priority Number>
...
```

Method #2

For applications where the time prior to execution of the BISR instruction is critical, the priority number of the interrupted task may be read from the CSA pointed to by the PCXI after execution of the BISR instruction.

```
...
bisr    #<New Priority Number>
mfcr    d15, #0xFE00            ; Copy PCXI to d15
sh.h    d14, d15, #12           ; Extract PCX seg to d14
insert  d15, d14, d15, #6, #16 ; Merge PCX offset to d15
mov.a   a15, d15                ; Copy to address reg
ld.bu   d15, [a15]0x3          ; Load byte containing PCPN
...
```

Note that contrary to the TriCore architecture specification, no DSYNC instruction is stricly necessary after the BISR (or SVLCX) instruction, in either the TriCore1.3 or TriCore1.3.1, to ensure the previous CSA contents are flushed to memory. In both TriCore1.3 and TriCore1.3.1, any lower context save

operation (BISR or SVLCX) will automatically flush any cached upper context to memory before the lower context is saved.

CPU_TC.H005 Wake-up from Idle/Sleep Mode

A typical use case for idle or sleep mode is that software puts the CPU into one of these modes each time it has to wait for an interrupt.

Idle or Sleep Mode is requested by writing to the Power Management Control and Status Register (PMCSR). However, when the write access to PMCSR is delayed e.g. by a higher priority bus access, TriCore may enter idle or sleep mode while the interrupt which should wake up the CPU is already executed. As long as no additional interrupts are triggered, the CPU will endlessly stay in idle/sleep mode.

Therefore, e.g. the following software sequence is recommended (for user mode 1, supervisor mode):

```
_disable();           // disable interrupts
do {
SCU_PMCSR = 0x1;      // request idle mode
if( SCU_PMCSR );      // ensure PMCSR is written

_enable();            // after wake-up: enable interrupts
_nop();
_nop();               // ensure interrupts are enabled
_disable();           // after service: disable interrupts
} while( !condition ); // return to idle mode depending on
                      // condition set by interrupt handler
_enable();
```

FIRM_TC.H000 Reading the Flash Microcode Version

The 1-byte Flash microcode version number is stored at the bit locations 103-96 of the LDRAM address D000 000C$_H$ after each reset, and subject to be overwritten by user data at any time.

The version number is defined as “Vsn”, contained in the byte as:

- **s** = highest 4 bit, hex number
- **n** = lowest 4 bit, hex number

Example: V21, V23, V3A, V3F, etc.

FlexRay_AI.H004 Only the first message can be received in External Loop Back mode

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

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

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

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

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

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

Now the next transmission can be requested.

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

The initialization of the E-Ray internal RAMs as started after hardware reset or by CHI command CLEAR_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_AI.H010 Driver SW must launch CLEAR_RAMS command before reading from E-Ray RAMs

After a Power-on-Reset, the RAMs used by the E-Ray module must be written once. The recommended solution is to trigger a “CLEAR_RAMS” command (via register SUCC1). “CLEAR_RAMS” fills a defined value into all memory locations.

An alternate solution is to write explicitly by SW to all RAM locations, which are intended to be read later, e.g. by writing the complete configuration and writing into all allocated message buffers, including receive buffers. The latter activity may be required if buffers are configured to store frames sent in the dynamic segment. The sent frames may be smaller than the configured buffer size. If the SW reads the amount of configured data (not the amount of received data), it may read from non-activated RAM locations.

Reading from RAM locations before at least writing once to it may cause a Parity Error Trap (TC1797) or an ECC Error Trap.

FlexRay_TC.H001 RAM Error Indicators need to be cleared at Module Init

The following ECC error notification bits must be cleared by software within the ERay kernel at the end of module initialization / before use of the module:

- Flag EERR in register EIR
- Flags EIBF, EOBF, EMR, ETBF1, ETBF2 in register MHDS

These ECC error flags might be set after reset or MBIST and would lead to an unmotivated ECC error service request if enabled (default after reset: disabled).

Recommendation

The flags listed above must be cleared by software at the end of module initialization / before use of the module. These flags are cleared by writing 1_B to the respective bit positions.

FPI_TC.H001 FPI bus may be monopolized despite starvation protection

During a sequence of back to back 64-bit writes performed by the CPU to PCP memories (PRAM/CMEM) the LFI will lock the FPI bus and no other FPI master (PCP, DMA, OCDS) will get a grant, regardless of the priority, until the sequence is completed.

A potential situation would be a routine which writes into the complete PRAM and CMEM to initialize the parity bits (for devices with parity) or ECC bits (for devices with ECC), respectively. If the write accesses are tightly concatenated, the FPI bus may be monopolized during this time. Such situation will not be detected by the starvation protection.

Workaround

Avoid 64-bit CPU to PCP PRAM/CRAM accesses.

GPT12_AI.H001 Modification of Block Prescalers BPS1 and BPS2

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

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

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

GPTA_TC.H004 Handling of GPTA Service Requests

Concerning the relations between two events (request_1, request_2) from different service request sources that belong to the same service request group y of the GPTA module, two standard cases (1, 2) and one corner case can be differentiated:

Case 1

When request_2 is generated **before** the previous request_1 has been acknowledged, the common Service Request Flag SRR of service request group y is cleared after request_1 is acknowledged. Since the occurrence of request_1 and request_2 is also flagged in the Service Request State Registers SRS*,[1] all request sources can be identified by reading SRS* in the interrupt service routine or PCP channel program, respectively.

Case 2

When request_2 is generated **after** request_1 has been acknowledged, both flag SRR and the associated flag for request_2 in register SRS* are set, and the interrupt service routine/PCP channel program will be invoked again.

Corner Case

When request_2 is generated while request_1 is in the **acknowledge phase**, and the service routine/PCP channel program triggered by request_1 is reading register SRS* to determine the request source, then the following scenario may occur:

Depending on the relations between module clock f_{GPTA}, FPI-Bus clock, and the number of cycles required until the instruction reading SRS* is executed, the value read from SRS* may not yet indicate request_2, but only request_1 (unlike case 1). On the other hand, flag SRR (cleared when request_1 was acknowledged) is not set to trigger service for request_2 (unlike case 2).

As a consequence, recognition and service of request_2 will be delayed until the next request of one of the sources connected to this service request group y is generated.

1) SRS* = abbreviation for Service Request State Registers SRSCn or SRSSn.

Identification of Affected Systems

A system will **not** be affected by the corner case described above when the following condition is true:

(1a) READ - ACK ≥ max(icu, (N-1)*FPIDIV) for FDR in Normal Mode, or

(1b) READ - ACK ≥ max(icu, N*FPIDIV) for FDR in Fractional Mode

with:

- READ = number of f_{CPU} [1] or f_{PCP} cycles between interrupt request (at CPU/PCP site) and register SRS* read operation.
  Number of cycles depends on implementation of service routine. “Worst case” with respect to corner case is minimum time:
  - READ = R_0 = 10 if instruction reading SRS* is directly located at entry point in Interrupt Vector Table in CPU Interrupt Service (sub-)routine
  - READ = R_1 = 14 if instruction reading SRS* is first instruction in CPU Interrupt Service (sub-)routine
  - Read = R_P = 16 if instruction reading SRS* is first instruction in PCP channel program
  - R_X: number of extra f_{CPU} or f_{PCP} cycles to be added to R_0, R_1, or R_P, respectively, in case instruction reading SRS* is not the first instruction in the corresponding service routine.
- ACK = number of f_{CPU} or f_{PCP} cycles between interrupt request (at CPU/PCP site) and clearing of request flag SRR
  - ACK = 7 = constant for TriCore and PCP under all conditions (independent from ICU/PICU configuration)
- icu = clock ratio between ICU and CPU clock
  - icu = 2 with bit ICR.CONECYC=1_B, icu = 4 with bit ICR.CONECYC=0_B
- N = “maximum integer value” of clock ratio f_{FPI} / f_{GPTA}
  - N = 1024 - STEP for Normal Divider mode (DM = 01_B)
  - N = (1024 DIV STEP) + 1 for Fractional Divider mode (DM = 10_B), where DIV means “integer division”
- FPIDIV = clock ratio f_{CPU} / f_{FPI} for CPU and f_{PCP} / f_{FPI} for PCP

1) f_{CPU} = f_{LMB} or f_{SRI}, depending on bus structure used in specific product.

Example 1

PCP reads register SRS* with first instruction, GPTA is configured with fractional divider, STEP = $E4_H$, CONECYC = 0_B, FPIDIV = 2 ($f_{PCP} = 2*f_{FPI}$)

This results in:

16 - 7 ≥ max(4, (1024 DIV 228 + 1)*2), or

9 ≥ max(4, (5*2)), or

9 ≥ max(4, 10), where max(4, 10) = 10

i.e. 9 ≥ 10 is false

i.e. this configuration is critical with respect to the corner case described above.

Example 2

PCP reads register SRS* with first instruction, GPTA is configured with fractional divider, STEP = $38E_H$, CONECYC = 0_B, FPIDIV = 2 ($f_{PCP} = 2*f_{FPI}$)

This results in:

16 - 7 ≥ max(4, (1024 DIV 910 + 1)*2), or

9 ≥ max(4, (2*2)), or

9 ≥ max(4, 4), where max(4, 4) = 4

i.e. 9 ≥ 4 is true

i.e. this configuration is not affected by the corner case described above.

Recommendation

In case a system is affected by the corner case described above, the service routine/PCP channel program should read the status flags in SRS* again ≥ 1 GPTA module clock cycle after the first read operation to ensure earliest possible recognition of all events, e.g.:

```
Service Routine/PCP Program Entry:
 - Read SRS*
  - if flag is set: handle requesting source, clear
      corresponding flag via register SRSCx
  - Ensure elapsed time to next read of SRS* in Loop is
      ≥ 1 GPTA module clock cycle since routine entry
  Loop:
```

```
    - Read SRS*, exit if all flags are 0
    - Handle requesting source(s), clear
      corresponding flag(s) via register SRSCx
```

or (when the GPTA module clock is relatively high) e.g.:

```
Service Routine/PCP Program Entry:
 - Ensure time to first read of SRS* in Loop is
      ≥ 1 GPTA module clock cycle since routine entry
  Loop:
    - Read SRS*, exit if all flags are 0
    - Handle requesting source(s), clear
      corresponding flag(s) via register SRSCx
```

Note: In case the condition in formula (1a) or (1b) is not true, it would be possible to add n ≥ Rx + FPIDIV - 1 NOPs (+ ISYNC for CPU) at the beginning of the service routine to extend the time until SRS is read.*
Referring to Example 1 (Rx ≥ 1 cycle is missing, FPIDIV = 2), n ≥ 2 NOPs may be added before SRS is read to make this configuration uncritical. Make sure the NOPs are not eliminated by code optimizations.*
However, basically it is still recommended to follow the general hint in paragraph "Recommendation" to improve code portability and become independent of cycle counting for individual configurations.

MSC_TC.H007 Start Condition for Upstream Channel

The reception of the upstream frame is started when a falling edge (1-to-0 transition) is detected on the SDI line.

In addition, reception is also started when a low level is detected on the SDI line while the upstream channel was in idle state, i.e.

- when the upstream channel is switched on (bit field `URR` in register `USR` is set to a value different from 000_B) and the SDI line is already on a low level, or
- after a frame has been received, and the SDI line is on a low level at the end of the last stop bit time slot (e.g. when the SDI line is permanently held low).

Therefore, make sure that the SDI line is pulled high (e.g. with an internal or external pull-up) while no transmission is performed.

MSC_TC.H010 Configuration of SCU.EMSR for the EMGSTOPMSC Signal

The emergency stop input signal EMGSTOPMSC of the MSC module is connected to the output signal of the emergency stop control logic located in the SCU. Its functionality is controlled by the SCU emergency stop register `SCU.EMSR`.

The emergency stop input line EMGSTOPMSC is used to indicate an emergency stop condition of a power device. In emergency case, shift register bits can be loaded bit-wise from the downstream data register instead from the ALTINL and ALTINH buses.

The emergency stop frame is sent out at each trigger event as long as the emergency stop signal is active. This means that in data repetition mode the emergency stop frame is repeatedly sent as long as the emergency stop signal is active.

Note: If the emergency stop signal is used by the MSC module with setting SCU.EMSR.MODE = 1_B (Asynchronous Mode), there is some low probability that the first emergency stop frame could be corrupted, but the following emergency stop frames will be correct.

Recommendation

- If the emergency stop signal is used by the MSC module, setting `SCU.EMSR.MODE` = 0_B (Synchronous Mode) is mandatory.
- Setting `SCU.EMSR.MODE` = 1_B (Asynchronous Mode) is not allowed to be used with the MSC module.

MultiCAN_AI.H005 TxD Pulse upon short disable request

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

Example for setup of the CAN disable request:

`CAN_CLC.DISR` = 1 and then `CAN_CLC.DISR` = 0

Workaround

Set all INIT bits to 1 before requesting module disable.

MultiCAN_AI.H006 Time stamp influenced by resynchronization

The time stamp measurement feature is not based on an absolute time measurement, but on actual CAN bit times which are subject to the CAN resynchronization during CAN bus operation.The time stamp value merely indicates the number of elapsed actual bit times. Those actual bit times can be shorter or longer than nominal bit time length due to the CAN resynchronization events.

Workaround

None.

MultiCAN_AI.H007 Alert Interrupt Behavior in case of Bus-Off

The MultiCAN module shows the following behavior in case of a bus-off status:

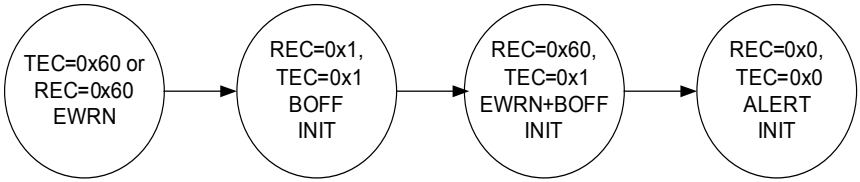


Figure 2 Alert Interrupt Behavior in case of Bus-Off

When the threshold for error warning (EWRN) is reached (default value of Error Warning Level EWRN = 0x60), then the EWRN interrupt is issued. The bus-off (BOFF) status is reached if TEC > 255 according to CAN specification, changing the MultiCAN module with REC and TEC to the same value 0x1, setting the INIT bit to 1_B, and issuing the BOFF interrupt. The bus-off recovery phase starts automatically. Every time an idle time is seen, REC is incremented. If REC = 0x60, a combined status EWRN+BOFF is reached. The corresponding interrupt can also be seen as a pre-warning interrupt, that the bus-off recovery

phase will be finished soon. When the bus-off recovery phase has finished (128 times idle time have been seen on the bus), EWRN and BOFF are cleared, the ALERT interrupt bit is set and the INIT bit is still set.

MultiCAN_AI.H008 Effect of CANDIS on SUSACK

When a CAN node is disabled by setting bit NCR.CANDIS = 1_B, the node waits for the bus idle state and then sets bit NSR.SUSACK = 1_B.

However, SUSACK has no effect on applications, as its original intention is to have an indication that the suspend mode of the node is reached during debugging.

MultiCAN_TC.H002 Double Synchronization of receive input

The MultiCAN module has a double synchronization stage on the CAN receive inputs. This double synchronization delays the receive data by 2 module clock cycles. If the MultiCAN is operating at a low module clock frequency and high CAN baudrate, this delay may become significant and has to be taken into account when calculating the overall physical delay on the CAN bus (transceiver delay etc.).

MultiCAN_TC.H003 Message may be discarded before transmission in STT mode

If `MOFCRn.STT`=1 (Single Transmit Trial enabled), bit TXRQ is cleared (TXRQ=0) as soon as the message object has been selected for transmission and, in case of error, no retransmission takes places.

Therefore, if the error occurs between the selection for transmission and the real start of frame transmission, the message is actually never sent.

Workaround

In case the transmission shall be guaranteed, it is not suitable to use the STT mode. In this case, `MOFCRn.STT` shall be 0.

MultiCAN_TC.H004 Double remote request

Assume the following scenario: A first remote frame (dedicated to a message object) has been received. It performs a transmit setup (TXRQ is set) with clearing NEWDAT. MultiCAN starts to send the receiver message object (data frame), but loses arbitration against a second remote request received by the same message object as the first one (NEWDAT will be set).

When the appropriate message object (data frame) triggered by the first remote frame wins the arbitration, it will be sent out and NEWDAT is not reset. This leads to an additional data frame, that will be sent by this message object (clearing NEWDAT).

There will, however, not be more data frames than there are corresponding remote requests.

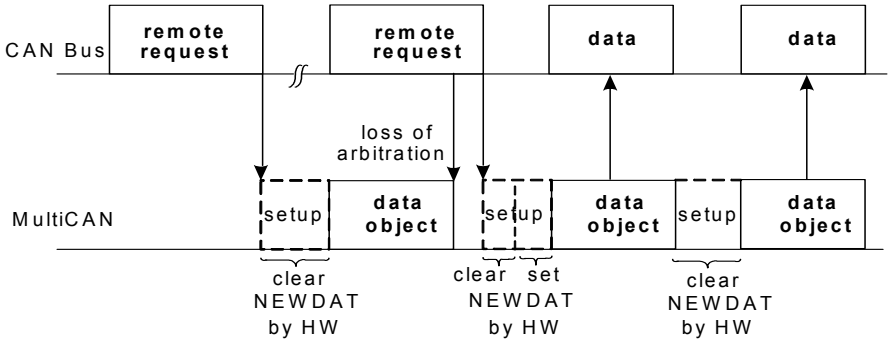


Figure 3 Loss of Arbitration

OCDS_TC.H001 IOADDR may increment after aborted IO_READ_BLOCK

If an IO_READ_BLOCK instruction is aborted by the host (switching the TAP controller to the update-DR state before enough data bits have been shifted out) it may happen under certain clock ratios that the IOADDR register is incremented nevertheless. This will result in an access to the wrong data in the succeeding IO_READ_* or IO_WRITE_* instruction.

Workaround

As the host is actively causing the abort, it should be fully aware of the situation. The workaround now simply is to rewrite the IOADDR register (using the IO_SET_ADDRESS instruction) after each aborted block transfer.

Note: This usually is done anyway at the beginning of the next transaction.

OCDS_TC.H002 Setting IOSR.CRSYNC during Application Reset

If the host is shifting in a Communication Mode IO_READ_WORD instruction in the very moment an Application Reset happens, the read request flag (CBS_IOSR.CRSYNC) may be already set after the execution of the startup software. A monitor program may be confused by this and drop out of the higher level communication protocol, especially if the host posts an instruction (with the IO_WRITE_WORD instruction) after detecting the reset.

Workaround

Two correlated activities should be incorporated in the tool software:

- After each reset the host should explicitly use CBS_IOCONF.COM_RST to reset any erroneously pending requests.
- The higher level protocol should require a specific answer to the very first command sent from the host to the device. Erroneous read requests then can be detected and skipped.

OCDS_TC.H003 Application Reset during host communication

Not only the host is able to cause resets of the device: External pins driven by the application, the internal watchdog and even the application program itself can trigger the reset generation process.

The only way to communicate reset events to the host is for Cerberus to reject the next instruction with “never-ending busy”, which should lead to communication time out on the host side.

The decision to accept or reject an instruction is done very early in the bit stream of the instruction. If an Application Reset happens after this point of time, the instruction will complete in most cases, and only the next one will be rejected.

As the temporal distance from reset event and instruction rejection is not fixed (apart from being sequential), it is highly recommended to check the IOINFO register (using the IO_SUPERVISOR instruction) each time an abnormally long busy period is experienced by the host. Especially a repetition of the rejected instruction should only be attempted if the possibility of Cerberus being in Error State has been excluded.

Workaround

Use IO_SUPERVISOR whenever a (too) long busy bit is observed.

OCDS_TC.H007 Early Acknowledgement of Channel Suspend for Active DMA Channel

This application hint is only relevant if a DMA channel is suspended for debug purposes and this DMA channel accesses a peripheral which is being suspended as well by the same event. If the DMA channel accesses memory, or if the accessed peripheral can be read/written without restrictions also in suspended state, there is no issue with the following behaviour.

When a debug suspend request is made while a DMA transfer is in progress, the specified behaviour is that the suspend request is only acknowledged on completion of the current DMA transfer.

However, in a specific corner case, if the suspend request is made in the same clock cycle as the start of the lowest priority channel with a pending access, i.e. the last DMA transfer to be executed, then the suspend request will be acknowledged immediately but the DMA transfer will continue to execute DMA moves until the current DMA transfer is complete.

Recommendation

Take into account that a peripheral may be suspended before the associated DMA triggering channel has entered the suspend state. Since this is a corner

case which will rarely happen, please restart the debug situation if there are any indicators that this corner case has occurred.

PWR_TC.H008 Internal pull-up device on $\overline{\text{PORST}}$ pad

The $\overline{\text{PORST}}$ pad includes a strong pull-down device controlled by the EVRs. This allows pin $\overline{\text{PORST}}$ to be actively driven low upon a threshold violation of (at least one of) the monitored supply voltages. For the characteristics of this pull-down see parameter $\overline{\text{PORST}}$ pad output current $I_{DDPORST}$ in the Data Sheet.

Besides this (switchable) pull-down, the $\overline{\text{PORST}}$ pad has a permanent internal weak pull-up (see parameter Pull-Up current I_{PUH} in Data Sheet).

Note that other members of the AudoMax family (without EVRs) have an internal weak pull-down on pad $\overline{\text{PORST}}$.

When supply is in the operational range, $\overline{\text{PORST}}$ pin/pad is held high by the internal weak pull-up. As a consequence, it cannot set the device into reset state in case PORST pin/bondwire breaks.

FMEA considerations of such a failure mode recommend that the device is set into a safe reset state when $\overline{\text{PORST}}$ cannot be activated from external devices (voltage regulators, safety monitors) in case critical failures are observed in the system and the microcontroller needs to be set into reset as a result.

Recommendation:

a) Connect the reset line additionally to the ESR1 pin (or other appropriate pins) to provide a redundant path to detect such a failure mode.

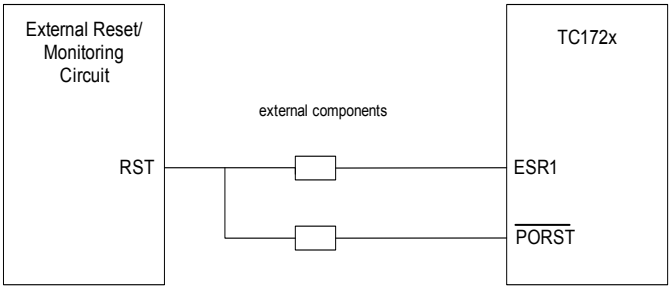


Figure 4 $\overline{\text{PORST}}$ Redundancy Path

b) Activate internal voltage monitoring for external supply which acts as a redundant supply watchdog in addition to the supply monitoring carried out by the external regulator.

SSC_AI.H002 Transmit Buffer Update in Master Mode during Trailing or Inactive Delay Phase

When the Transmit Buffer register TB is written in master mode after a previous transmission has been completed, the start of the next transmission (generation of SCLK pulses) may be delayed in the worst case by up to 6 SCLK cycles (bit times) under the following conditions:

- a trailing delay (SSOTC.TRAIL) > 0 and/or an inactive delay (SSOTC.INACT) > 0 is configured
- the Transmit Buffer is written in the last module clock cycle (f_{SSC} or f_{CLC}) of the inactive delay phase (if INACT > 0), or of the trailing delay phase (if INACT = 0).

No extended leading delay will occur when both TRAIL = 0 and INACT = 0.

This behaviour has no functional impact on data transmission, neither on master nor slave side, only the data throughput (determined by the master) may be slightly reduced.

To avoid the extended leading delay, it is recommended to update the Transmit Buffer after the transmit interrupt has been generated (i.e. after the first SCLK phase), and before the end of the trailing or inactive delay, respectively.

Alternatively, bit BSY may be polled, and the Transmit Buffer may be written after a waiting time corresponding to 1 SCLK cycle after BSY has returned to 0_B.

After reset, the Transmit Buffer may be written at any time.

SSC_AI.H003 Transmit Buffer Update in Slave Mode during Transmission

After reset, data written to the Transmit Buffer register TB are directly copied into the shift register. Further data written to TB are stored in the Transmit Buffer while the shift register is not yet empty, i.e. transmission has not yet started or is in progress.

If the Transmit Buffer is written in slave mode during the first phase of the shift clock SCLK supplied by the master, the contents of the shift register are overwritten with the data written to TB, and the first bit currently transmitted on line MRST may be corrupted. No Transmit Error is detected in this case.

It is therefore recommended to update the Transmit Buffer in slave mode after the transmit interrupt (TIR) has been generated (i.e. after the first SCLK phase).

After reset, the Transmit Buffer may be written at any time.

SSC_TC.H003 Handling of Flag STAT.BSY in Master Mode

In register STAT of the High-Speed Synchronous Serial Interface (SSC), some flags have been made available that reflect module status information (e.g. error, busy) closely coupled to internal state transitions. In particular, flag STAT.BSY will change twice during data transmission: from 0_B to 1_B at the start, and from 1_B to 0_B at the end of a transmission. This requires some special considerations e.g. when polling for the end of a transmission:

In master mode, when register TB has been written while no transfer was in progress, flag STAT.BSY is set to 1_B after a constant delay of 5 FPI bus clock cycles. When software is polling STAT.BSY after TB was written, and it finds that STAT.BSY = 0_B, this may have two different meanings: either the transfer has not yet started, or it is already completed.

Recommendations

In order to poll for the end of an SSC transfer, the following alternative methods may be used:

- either test flag `RSRC.SRR` (receive interrupt request flag) instead of `STAT.BSY`
- or use a software semaphore that is set when `TB` is written, and which is cleared e.g. in the SSC receive interrupt service routine.

5 Documentation Updates

CCU6_TC.D001 Register CMPMODIF

This documentation update refers to TC1728 User's Manual V1.0 2011-12,

- page 24-44 (= page 2139 of 2570 in pdf):
  - Bit MCC63S in register CMPMODIF is not located at bit 7 in register description, but at bit 6 as per register image.

DMA_TC.D001 Register DMA_CLRE

This documentation update refers to TC1728 User's Manual V1.0 2011-12,

- page 11-72 (= page 904 of 2570 in pdf):
  - Bits CTL[17:10] in register DMA_CLRE are not of 'rw' access type as seen in register image, but should be 'w' as per register description.

GPTA_TC.D001 Special Function Registers, Port Assignment

This documentation update refers to TC1728 User's Manual V1.0 2011-12,

- page 23-231 (= page 2074 of 2570 in pdf), Figure 23-82, GPTA ® v5 Implementation Specific Special Function Registers:
  - Registers LTCA2_SRCk are not used.
- page 23-232 (= page 2075 of 2570 in pdf), Figure 23-83, I/O Port Line Assignment, Port 3 lines:
  - P3.6 is not used for GPTA OUT signal.
- page 23-234 (= page 2077 of 2570 in pdf), Table 23-20, IOCR Assignment for GPTA ® v5 Port Lines, Port 3:
  - P3.5 and P3.6 are not used for GPTA OUT[89:88], i.e. P3.7 is connected to OUT89, P3.4 is connected to OUT88.
  - OUT[93:90] are controlled by P3_IOCR8,
  - OUT[97:94] are controlled by P3_IOCR12.

- page 23-235 (= page 2078 of 2570 in pdf), Table 23-20, IOCR Assignment for GPTA ® v5 Port Lines, Port 8 and 9:
  - OUT52 is not connected to P8.4, but OUT99 is connected to P8.4, i.e. P8.[7:4] is connected to OUT[102:99].
  - P8.[11:10] are connected to OUT[106:105], P8.[9:8] are connected to OUT[57:56].
  - P9.5 and P9.6 are not used for GPTA OUT84, OUT87, i.e. P9.7 is connected to OUT87, P9.4 is connected to OUT84.

MultiCAN_TC.D001 MultiCAN I/O Control Selection and Setup

This documentation update refers to TC1728 User's Manual V1.0 2011-12,

- page 20-119 (= page 1412 of 2570 in pdf), Chapter 20.5.4.1 Input/Output Function Selection in Ports:
  - The I/O lines for the MultiCAN module are controlled by the port input/output control registers P3_IOCR12, P4_IOCR0, P5_IOCR8, P5_IOCR12, and P9_IOCR0.