

**Device** TC1782  
**Marking/Step** BA  
**Package** see Data Sheet

## 04074AERRA

---

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

**Table 1 Current Documentation<sup>1)</sup>**

TC1784 User's Manual	V1.1	2011-05
TC1782 Data Sheet	V1.4.1	2014-05
TC1784 Data Sheet	V1.1.1	2014-05
TC1746 Data Sheet	V1.0	2012-04
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/AudoMax](http://www.infineon.com/AudoMax).

Each erratum identifier follows the pattern **Module\_Arch.TypeNumber**:

- **Module**: subsystem, peripheral, or function affected by the erratum
- **Arch**: microcontroller architecture where the erratum was firstly detected
  - **AI**: Architecture Independent
  - **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.

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

*Note: 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.*

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

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

*Note: 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.*

# 1 History List / Change Summary

**Table 2 History List**

Version	Date	Remark
1.0	2010-10-25	
1.1	2012-04-17	removed BCU_TC.006 (Polarity of bit SVM in register ECON) - covered by TC1784 User's Manual V1.1)
1.2	2016-07-12	<p>The following text modules describing Parameter Deviations have been removed from the Errata Sheet - covered by latest versions of corresponding Data Sheets:</p> <ul style="list-style-type: none"> <li>• FADC_TC.P004 (FADC DNL Error Specification for Gain 4 and 8)</li> <li>• Flash_TC.P001 (Increased Flash Programming Time)</li> <li>• MLI_TC.P002 (TREADY hold time after TCLK rising edge (t17))</li> <li>• PLL_ERAY_TC.P001 (PLL lock-in time)</li> <li>• PLL_TC.P006 (PLL lock-in time)</li> </ul>

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

**Table 3 Functional Deviations**

Functional Deviation	Short Description	Change	Page
<b>BROM_TC.006</b>	<b>Baud Rate Detection for CAN Bootstrap Loader</b>		<b>11</b>
<b>CPU_TC.111</b>	<b>Imprecise Return Address for FCU Trap</b>		<b>11</b>

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

<b>Functional Deviation</b>	<b>Short Description</b>	<b>Change</b>	<b>Page</b>
<b>CPU_TC.114</b>	<b>CAE Trap may be generated by UPDFL instruction</b>		<b>12</b>
<b>CPU_TC.117</b>	<b>Cached Store Data Lost on Data Cache Invalidate via Overlay</b>		<b>13</b>
<b>DMI_TC.016</b>	<b>CPU Deadlock possible when Cacheable access encounters Flash Double-Bit Error</b>		<b>15</b>
<b>EBU_TC.022</b>	<b>Write Data Delay Control for Asynchronous Memory Accesses</b>		<b>16</b>
<b>FADC_TC.005</b>	<b>Equidistant multiple channel-timers</b>		<b>17</b>
<b>FlexRay_AI.087</b>	<b>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</b>		<b>19</b>
<b>FlexRay_AI.088</b>	<b>A sequence of received WUS may generate redundant SIR.WUPA/B events</b>		<b>20</b>
<b>FlexRay_AI.089</b>	<b>Rate correction set to zero in case of SyncCalcResult=MISSING_TERM</b>		<b>21</b>
<b>FlexRay_AI.090</b>	<b>Flag SFS.MRCS is set erroneously although at least one valid sync frame pair is received</b>		<b>22</b>
<b>FlexRay_AI.091</b>	<b>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</b>		<b>22</b>
<b>FlexRay_AI.092</b>	<b>Initial rate correction value of an integrating node is zero if pMicroInitialOffsetA,B = 0x00</b>		<b>23</b>
<b>FlexRay_AI.093</b>	<b>Acceptance of startup frames received after reception of more than gSyncNodeMax sync frames</b>		<b>24</b>

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

<b>Functional Deviation</b>	<b>Short Description</b>	<b>Change</b>	<b>Page</b>
<b>FlexRay_AI.094</b>	<b>Sync frame overflow flag EIR.SFO may be set if slot counter is greater than 1024</b>		<b>24</b>
<b>FlexRay_AI.095</b>	<b>Register RCV displays wrong value</b>		<b>25</b>
<b>FlexRay_AI.096</b>	<b>Noise following a dynamic frame that delays idle detection may fail to stop slot</b>		<b>26</b>
<b>FlexRay_AI.097</b>	<b>Loop back mode operates only at 10 MBit/s</b>		<b>27</b>
<b>FlexRay_AI.099</b>	<b>Erroneous cycle offset during startup after abort of startup or normal operation</b>		<b>27</b>
<b>FlexRay_AI.100</b>	<b>First WUS following received valid WUP may be ignored</b>		<b>28</b>
<b>FlexRay_AI.101</b>	<b>READY command accepted in READY state</b>		<b>29</b>
<b>FlexRay_AI.102</b>	<b>Slot Status vPOC!SlotMode is reset immediately when entering HALT state</b>		<b>29</b>
<b>FlexRay_AI.103</b>	<b>Received messages not stored in Message RAM when in Loop Back Mode</b>	<b>New</b>	<b>30</b>
<b>FlexRay_AI.104</b>	<b>Missing startup frame in cycle 0 at coldstart after FREEZE or READY command</b>	<b>New</b>	<b>30</b>
<b>FlexRay_AI.105</b>	<b>RAM select signals of IBF1/IBF2 and OBF1/OBF2 in RAM test mode</b>	<b>New</b>	<b>31</b>
<b>FlexRay_AI.106</b>	<b>Data transfer overrun for message transfers Message RAM to Output Buffer (OBF) or from Input Buffer (IBF) to Message RAM</b>	<b>New</b>	<b>32</b>
<b>MSC_TC.009</b>	<b>Missing Receive Data Interrupt</b>		<b>35</b>
<b>MultiCAN_TC.041</b>	<b>Clock <math>f_{CLC}</math> used in Bit Timing Mode</b>	<b>New</b>	<b>36</b>
<b>MultiCAN_TC.042</b>	<b>CAN-to-DMA Request Connections</b>	<b>New</b>	<b>36</b>
<b>OCDS_TC.014</b>	<b>Triggered Transfer does not support half word bus transactions</b>		<b>36</b>

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

<b>Functional Deviation</b>	<b>Short Description</b>	<b>Change</b>	<b>Page</b>
<a href="#">OCDS_TC.015</a>	<a href="#">IOCONF register bits affected by Application Reset</a>		<a href="#">37</a>
<a href="#">OCDS_TC.016</a>	<a href="#">Triggered Transfer dirty bit repeated by IO_READ_TRIG</a>		<a href="#">37</a>
<a href="#">OCDS_TC.020</a>	<a href="#">ICTTA not used by Triggered Transfer to External Address</a>		<a href="#">38</a>
<a href="#">OCDS_TC.025</a>	<a href="#">PC corruption when entering Halt mode after a MTCR to DBGSR</a>		<a href="#">38</a>
<a href="#">OCDS_TC.026</a>	<a href="#">PSW.PRS updated too late after a RFM instruction.</a>		<a href="#">39</a>
<a href="#">OCDS_TC.027</a>	<a href="#">BAM breakpoints with associated halt action can potentially corrupt the PC.</a>		<a href="#">40</a>
<a href="#">OCDS_TC.028</a>	<a href="#">Accesses to CSFR and GPR registers of running program can corrupt loop exits.</a>		<a href="#">41</a>
<a href="#">OCDS_TC.035</a>	<a href="#">Debug reset will not disable the OCDS</a>		<a href="#">42</a>
<a href="#">PCP_TC.023</a>	<a href="#">JUMP sometimes takes an extra cycle</a>		<a href="#">43</a>
<a href="#">PCP_TC.032</a>	<a href="#">Incorrect PCP behaviour following FPI timeouts (as a slave)</a>		<a href="#">43</a>
<a href="#">PCP_TC.039</a>	<a href="#">PCP posted error interrupt to CPU may be lost when the queue is full in 2:1 mode</a>		<a href="#">44</a>
<a href="#">PCP_TC.040</a>	<a href="#">Coincident FPI RMW Access and PCP Atomic PRAM Instructions</a>		<a href="#">44</a>
<a href="#">SSC_AI.022</a>	<a href="#">Phase error detection switched off too early at the end of a transmission</a>		<a href="#">45</a>
<a href="#">SSC_AI.023</a>	<a href="#">Clock phase control causes failing data transmission in slave mode</a>		<a href="#">46</a>
<a href="#">SSC_AI.024</a>	<a href="#">SLSO output gets stuck if a reconfig from slave to master mode happens</a>		<a href="#">46</a>

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

Functional Deviation	Short Description	Change	Page
SSC_AI.025	First shift clock period will be one PLL clock too short because not synchronized to baudrate		46
SSC_AI.026	Master with highest baud rate set generates erroneous phase error		47

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

AC/DC/ADC Deviation	Short Description	Change	Page
AC_PER_TC.P001	Driver Settings for SSC and MLI Outputs on A2 Pads		48
FLASH_TC.P002	Increased Program Flash Erase Time $t_{ERP}$	New	48
MSC_TC.P002	Reduction of Output Levels $V_{OL}$ , $V_{OS}$ for LVDS pads		48

**Table 5 Application Hints**

Hint	Short Description	Change	Page
ADC_AI.H002	Minimizing Power Consumption of an ADC Module		51
ADC_AI.H003	Injected conversion may be performed with sample time of aborted conversion		51
ADC_TC.H013	Broken Wire Detection not implemented in Emulation Device TC178xED	New	53

**Table 5 Application Hints (cont'd)**

<b>Hint</b>	<b>Short Description</b>	<b>Change</b>	<b>Page</b>
<b>BROM_TC.H002</b>	<b>Enabling CAN Communication in CAN Bootstrap Loader Mode</b>		<b>53</b>
<b>CPU_TC.H004</b>	<b>PCXI Handling Differences in TriCore1.3.1</b>		<b>53</b>
<b>CPU_TC.H005</b>	<b>Wake-up from Idle/Sleep Mode</b>		<b>55</b>
<b>CPU_TC.H008</b>	<b>Instruction Memory Range Limitations</b>	<b>New</b>	<b>56</b>
<b>FIRM_TC.H000</b>	<b>Reading the Flash Microcode Version</b>		<b>56</b>
<b>FlexRay_AI.H004</b>	<b>Only the first message can be received in External Loop Back mode</b>		<b>57</b>
<b>FlexRay_AI.H005</b>	<b>Initialization of internal RAMs requires one eray_bclk cycle more</b>		<b>57</b>
<b>FlexRay_AI.H006</b>	<b>Transmission in ATM/Loopback mode</b>		<b>58</b>
<b>FlexRay_AI.H007</b>	<b>Reporting of coding errors via TEST1.CERA/B</b>		<b>58</b>
<b>FlexRay_AI.H009</b>	<b>Return from test mode operation</b>		<b>58</b>
<b>FlexRay_AI.H010</b>	<b>Driver SW must launch CLEAR_RAMs command before reading from E-Ray RAMs</b>		<b>59</b>
<b>FPI_TC.H001</b>	<b>FPI bus may be monopolized despite starvation protection</b>		<b>59</b>
<b>GPTA_TC.H004</b>	<b>Handling of GPTA Service Requests</b>		<b>60</b>
<b>GPTA_TC.H005</b>	<b>Switching between GT0 and GT1 in greater equal Compare Mode when <math>f_{GPTA} &lt; f_{SYS}</math></b>	<b>New</b>	<b>63</b>
<b>INT_TC.H003</b>	<b>Safe Cancellation of Service Requests</b>	<b>New</b>	<b>66</b>
<b>MSC_TC.H007</b>	<b>Start Condition for Upstream Channel</b>		<b>66</b>
<b>MSC_TC.H010</b>	<b>Configuration of SCU.EMSR for the EMGSTOPMSC Signal</b>	<b>New</b>	<b>66</b>
<b>MultiCAN_AI.H005</b>	<b>TxD Pulse upon short disable request</b>		<b>67</b>

**Table 5 Application Hints (cont'd)**

<b>Hint</b>	<b>Short Description</b>	<b>Change</b>	<b>Page</b>
<a href="#">MultiCAN_AI.H006</a>	<a href="#">Time stamp influenced by resynchronization</a>		<a href="#">68</a>
<a href="#">MultiCAN_AI.H007</a>	<a href="#">Alert Interrupt Behavior in case of Bus-Off</a>		<a href="#">68</a>
<a href="#">MultiCAN_AI.H008</a>	<a href="#">Effect of CANDIS on SUSACK</a>		<a href="#">69</a>
<a href="#">MultiCAN_AI.H009</a>	<a href="#">Behavior of MSGVAL for Remote Frames in Single Data Transfer Mode - Documentation Update</a>	New	<a href="#">69</a>
<a href="#">MultiCAN_TC.H002</a>	<a href="#">Double Synchronization of receive input</a>		<a href="#">70</a>
<a href="#">MultiCAN_TC.H003</a>	<a href="#">Message may be discarded before transmission in STT mode</a>		<a href="#">70</a>
<a href="#">MultiCAN_TC.H004</a>	<a href="#">Double remote request</a>		<a href="#">70</a>
<a href="#">MultiCAN_TC.H005</a>	<a href="#">Documentation Updates related to Controller Area Network Controller (MultiCAN)</a>	New	<a href="#">71</a>
<a href="#">OCDS_TC.H001</a>	<a href="#">IOADDR may increment after aborted IO_READ_BLOCK</a>		<a href="#">74</a>
<a href="#">OCDS_TC.H002</a>	<a href="#">Setting IOSR.CRSYNC during Application Reset</a>		<a href="#">74</a>
<a href="#">OCDS_TC.H003</a>	<a href="#">Application Reset during host communication</a>		<a href="#">75</a>
<a href="#">OCDS_TC.H007</a>	<a href="#">Early Acknowledgement of Channel Suspend for Active DMA Channel</a>		<a href="#">75</a>
<a href="#">PORTS_TC.H005</a>	<a href="#">Pad Input Registers do not capture Boundary-Scan data when BSD-mode signal is set to high</a>		<a href="#">76</a>
<a href="#">SCU_TC.H004</a>	<a href="#">Flag TRAPSTAT.SYSVCOLCKT after Power-on or System Reset</a>		<a href="#">76</a>
<a href="#">SSC_AI.H002</a>	<a href="#">Transmit Buffer Update in Master Mode during Trailing or Inactive Delay Phase</a>		<a href="#">77</a>

**Table 5 Application Hints (cont'd)**

Hint	Short Description	Change	Page
<a href="#">SSC_AI.H003</a>	<a href="#">Transmit Buffer Update in Slave Mode during Transmission</a>		<a href="#">77</a>
<a href="#">SSC_TC.H003</a>	<a href="#">Handling of Flag STAT.BSY in Master Mode</a>		<a href="#">78</a>

**Table 6 Documentation Updates**

Hint	Short Description	Change	Page
<a href="#">DMA_TC.D001</a>	<a href="#">Register DMA_CLRE</a>	New	<a href="#">79</a>
<a href="#">SSC_TC.D002</a>	<a href="#">Documentation Updates related to SSC Signal Description</a>	New	<a href="#">79</a>

## 2 Functional Deviations

### **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`.

### **EBU\_TC.022 Write Data Delay Control for Asynchronous Memory Accesses**

The EBU allows the timing of the write data driven onto the `EBU_AD(31:0)` pins to be adjusted using the `EBU_BUSWCONx.ECSE` and `EBU_BUSWAPx.EXTCLOCK` register fields. This delay mechanism is not working as specified for asynchronous write accesses:

- The time at which write data is disabled cannot be delayed by half a clock cycle. Register settings where a half clock cycle delay would be expected will result in a full clock cycle of delay.
- The time at which write data is enabled is never delayed. The bus will always be driven as if no delay was in effect. If the register settings require the data to be delayed then invalid data will be driven for the delay period.
- The time at which valid write data is driven cannot be delayed by half a clock cycle. Register settings where a half clock cycle delay would be expected will result in no delay being applied.

This results in the timing detailed in the table below, where CP1 is the first clock cycle of the command phase, DHn is the last clock cycle of the Data Hold Phase and  $T_{CLK}$  is one period of the EBU clock:

**Table 7 Write Data Signal Timing**

EXTCLOCK is set to	Driven at:		Removed at:	
	Delay Disabled	Delay Enabled	Delay Disabled	Delay Enabled
00 <sub>B</sub>	Start of CP1	Start of CP1	End of DHn <sup>1)</sup>	End of DHn + $T_{CLK}$
01 <sub>B</sub> , 10 <sub>B</sub> , 11 <sub>B</sub>	Start of CP1	End of CP1 <sup>2)</sup>	End of DHn	End of DHn + $T_{CLK}$

1) DHn indicates the final Data Hold Phase. If no Data Hold is programmed, this will be CPn, the final Command Phase.

2) Data bus will be enabled at the beginning of CP1

### Workaround

Adjust the phase lengths for the asynchronous regions to compensate for the modified signal delays.

### **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.

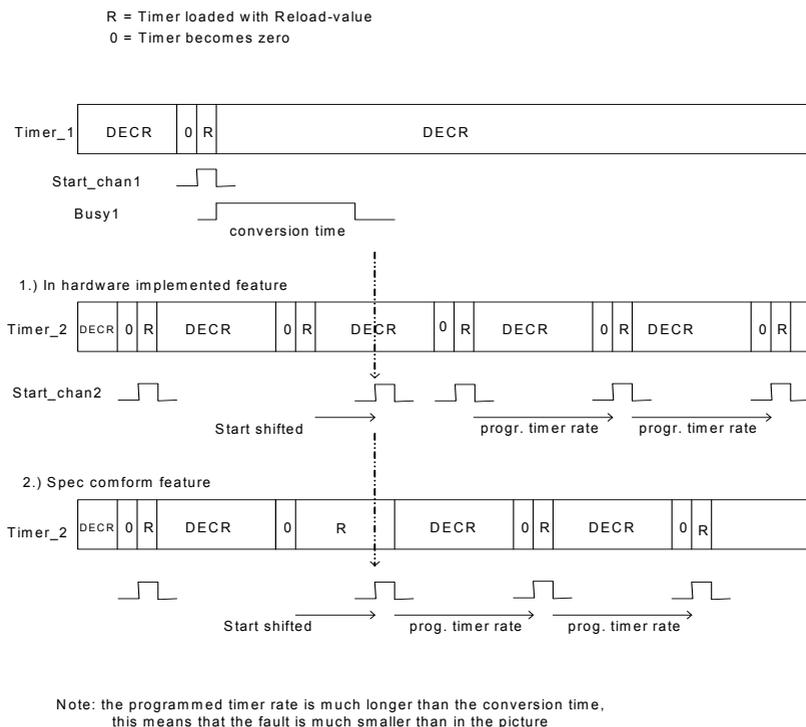
Timer\_1 and Timer\_2 are running with different reload-values. Both timers should start conversions with the requirement of equidistant timing.

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] = 00B`).

**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 ≤ 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 = 1001B)`.

**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] = 00B`). 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 vPOCISlotMode 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.

### **FlexRay AI.104 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 AL106 Data transfer overrun for message transfers Message RAM to Output Buffer (OBF) or from Input Buffer (IBF) to Message RAM**

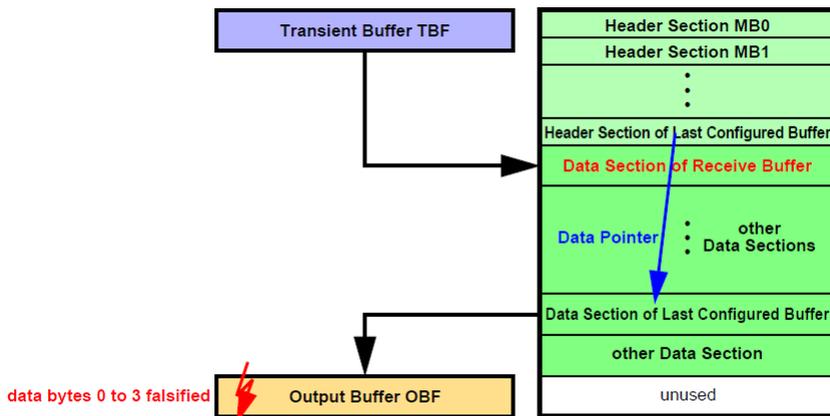
The problem occurs under the following conditions:

- 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

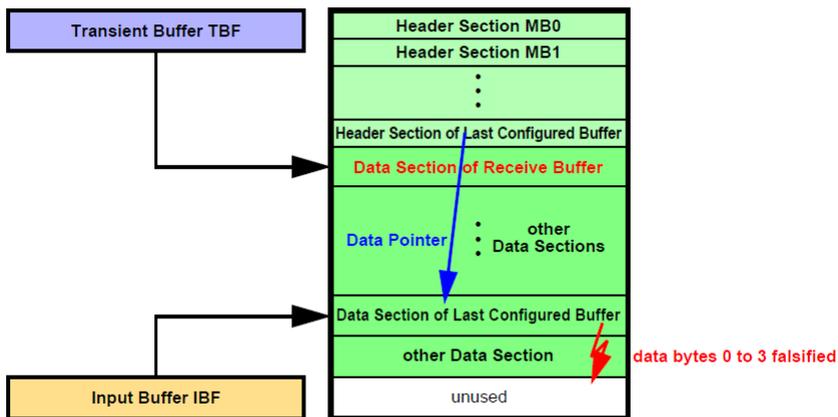
When the message buffer has its payload configured to maximum length (PLC = 127), the OBF word on address 00h (payload data bytes 0 to 3) is overwritten with unexpected data at the end of the transfer.



**Figure 2 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.



**Figure 3 Message buffer transfer from IBF to Message RAM**

**Scope:**

The erratum is limited to the case when (see [Figure 4](#) “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 2](#)).
- 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 3](#)).

**Severity:**

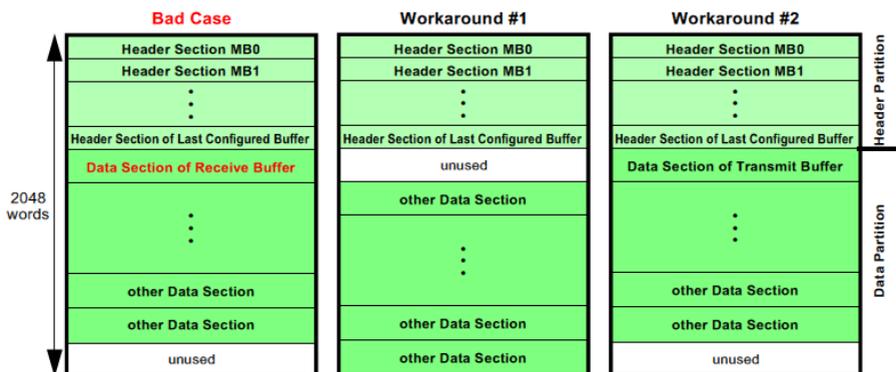
Medium, workaround available, check of configuration necessary.

**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 4 Message RAM Configurations**

**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 = 10B` is selected as interrupt generation condition (i.e. interrupt only if received data is not equal to `00H`), 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 = 10B` (depending on data bits of received frame). Use e.g. `ICR.RDIE = 01B`, 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.

### **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.014 Triggered Transfer does not support half word bus transactions**

The register bit `CBS_IOCONF.EX_BUS_HW` does not have any influence on the transaction width; only word wide transfer (32 bit) is implemented.

**Workaround**

No workaround possible. Choose source (IOADDR) and destination (ICTTA) addresses in word wide areas only.

**OCDS TC.015 IOCONF register bits affected by Application Reset**

The IOCONF register is erroneously cleared by each Application Reset. Therefore Communication Mode is entered whenever the TriCore is reset.

As the interaction with the tool is suspended anyway due to Error State of the IOClient, no immediate damage is done.

To resume interaction after leaving the Error State (IO\_SUPERVISOR instruction) however the required mode must be restored by rewriting the IOCONF register (IO\_CONFIG instruction).

**Workaround**

After detecting an Application Reset (IOINFO.BUS\_RST set) the IOCONF register should be rewritten by the tool after the Error State is left.

**OCDS TC.016 Triggered Transfer dirty bit repeated by IO\_READ\_TRIG**

The dirty bit appended to the data of an IO\_READ\_WORD instruction during Triggered Transfer mode indicates that there was at least one extra trigger event missed prior to capturing the transmitted data. The dirty bit is therefore cleared after each IO\_READ\_WORD. A consecutive IO\_READ\_TRIG instruction however will erroneously undo the clear. The next IO\_READ\_WORD will then again see a set dirty bit even if no trigger was missed.

**Workaround**

Do not issue an IO\_READ\_TRIG instruction after an IO\_READ\_WORD returned a set dirty bit.

**OCDS TC.020 ICTTA not used by Triggered Transfer to External Address**

In “Triggered Transfer to External Address” Mode bits 24...0 of the target address are fixed to the reset value of ICTTA. Only the most significant byte can be changed by IO\_SET\_TRADDR (or by writing to ICTTA).

*Note: This is the behavior of the Cerberus implemented prior to AudoNG.*

It is therefore not possible to use Cerberus as “DMA” work-alike to move trace data to the outside world via an interface like ASC.

**Workaround**

No workaround in “Triggered Transfer to External Address” mode possible, only the fixed address  $xx10F068_H$  can be used.

In “Internal Mode” however ICTTA is working as specified, so for certain use cases the intended DMA functionality can be activated by a code snippet executed by the TriCore or PCP as long as the Debug Interface is not needed concurrently.

**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\_instruction\_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.039 PCP posted error interrupt to CPU may be lost when the queue is full in 2:1 mode**

In the unlikely case where ..

- PCP 2:1 mode is enabled,
- PCP is configured to post error interrupts to CPU,
- a channel is running,
- this channel's R7.CEN is cleared,
- PCP exits this channel with posting an interrupt to the CPU,
- as a result of the posted interrupt, CPU queue becomes full,
- and the same channel is invoked again immediately with context restore optimization,

the current channel should exit with posting an error interrupt to CPU, but actually the error interrupt to CPU is lost.

## Workaround

Application software should not clear R7.CEN if there is a chance that the channel is going to be executed again.

### **PCP\_TC.040 Coincident FPI RMW Access and PCP Atomic PRAM Instructions**

If an FPI read/modify/write (rmw) instruction from an external master coincides with a PCP atomic PRAM instruction, atomicity may be broken resulting in indeterminate data for the PCP instruction. The conditions for this to occur is both instructions accessing the same PRAM memory (word) address.

PCP atomic PRAM instructions in this context include: MSET.PI, MCLR.PI, XCH.PI

### Workaround

Place a dummy FPI read in front of every PCP atomic PRAM instruction, i.e.

- Replace MSET.PI with:

```
CLR R7 0x5 (prevent nested interrupt)
NOP
LD.F R4, [R0], size=32
      (dummy load, addr setup required)
MSET.PI
```

- Replace MCLR.PI with:

```
CLR R7 0x5 (prevent nested interrupt)
NOP
LD.F R4, [R0], size=32
      (dummy load, addr setup required)
MCLR.PI
```

- Replace XCH.PI with:

```
CLR R7 0x5 (prevent nested interrupt)
NOP
LD.F R4, [R0], size=32
      (dummy load, addr setup required)
XCH.PI
```

### **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 synchronized 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.P0 = 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 too 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**

#### **AC PER TC.P001 Driver Settings for SSC and MLI Outputs on A2 Pads**

In general, AC timings are specified for the strongest/fastest possible output driver setting.

However, for SSC and MLI, the specified timings are only valid if outputs on A2 pads are configured for strong driver/medium edge mode.

#### **FLASH TC.P002 Increased Program Flash Erase Time $t_{ERP}$**

As per Data Sheet, the Program Flash Erase Time per 256 Kbyte Sector (symbol  $t_{ERP}$ ) is specified as 5 s maximum. This has been (and continues to be) ensured by a cumulative erase time test approach for all PFlash sectors.

However, the actual Program Flash Erase Time for a single 256 Kbyte Sector on a device may be up to 5.5 s maximum. This will be covered by an individual sector limit test specification as follows:

**Table 8 FLASH32 Parameters**

Parameter	Symbol	Values			Unit	Note/Test Condition
		Min.	Typ.	Max.		
Program Flash Erase Time per 256 Kbyte Sector	$t_{ERP}$	-	-	5.5 (instead of 5)	s	

#### **MSC TC.P002 Reduction of Output Levels $V_{OL}$ , $V_{OS}$ for LVDS pads**

The specified minimum values of the output voltage levels  $V_{OL}$ ,  $V_{OS}$  for LVDS pads operated in LVDS mode (see [Table 9](#)) will be reduced based on the

**Deviations from Electrical- and Timing Specification**

cumulated operating time in CMOS mode, as shown in **Table 10** (packaged devices) or **Table 11** (bare die version TC1746).

CMOS mode is active e.g. after reset until the pads are configured for LVDS mode by software, or when the pads are switched from LVDS to CMOS mode for power saving reasons.

*Note: The limits for  $V_{OD}$  (Output Differential Voltage) and  $V_{OH}$  (Output High Voltage, max. value) in LVDS mode and the parameters for CMOS mode are not affected by this drift effect.*

**Table 9 Output Levels  $V_{OL}$ ,  $V_{OH}$ ,  $V_{OS}$  for LVDS pads as per Data Sheet**

Parameter	Symbol	Values		Unit	Note / Test Condition
		Min.	Max.		
Output low voltage	$V_{OL}$	875	-	mV	as per Data Sheet
Output high voltage	$V_{OH}$	-	1 525		
Output offset voltage	$V_{OS}$	1 075	1 325		

For packaged devices (TC178y/ED), the drift is depending on stress duration, and results in reduced values for  $V_{OL}$  and  $V_{OS}$  as exemplarily shown in the following **Table 10** for 50/500/1000 hours:

**Table 10 Reduction of Output Levels  $V_{OL}$ ,  $V_{OS}$  for LVDS pads - Packaged Devices TC178y/ED**

Parameter	Symbol	Values		Unit	Note / Cumulated Operating Time in CMOS Mode
		Min.	Max.		
Output low voltage	$V_{OL}$	870	-	mV	50 h
		832	-		500 h
		801	-		1000 h
Output high voltage	$V_{OH}$	-	1 525	mV	( $V_{OH}$ max. value not affected)

**Deviations from Electrical- and Timing Specification**
**Table 10 Reduction of Output Levels  $V_{OL}$ ,  $V_{OS}$  for LVDS pads - Packaged Devices TC178y/ED (cont'd)**

Parameter	Symbol	Values		Unit	Note / Cumulated Operating Time in CMOS Mode
		Min.	Max.		
Output offset voltage	$V_{OS}$	1 070	1 325	mV	50 h
		1 032	1 325		500 h
		1 001	1 325		1000 h

For TC1746, the reduced values are shown in [Table 11](#):

**Table 11 Reduction of Output Levels  $V_{OL}$ ,  $V_{OS}$  for LVDS pads - TC1746**

Parameter	Symbol	Values		Unit	Note / Cumulated Operating Time in CMOS Mode
		Min.	Max.		
Output low voltage	$V_{OL}$	870	-	mV	50 h
Output high voltage	$V_{OH}$	-	1 525	mV	( $V_{OH}$ max. value not affected)
Output offset voltage	$V_{OS}$	1 070	1 325	mV	50 h

## 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)<sup>1)</sup> 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 AUDDO-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  $INPCR_x$  (for  $c_c$ ) and  $INPCR_y$  (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)  $N_A = \text{ratio } f_{ADC}/f_{ADCI}$  ( $N_A = 4 \dots 63$ , as defined in bit field  $DIVA$ ),
- b)  $N_D = \text{ratio } f_{ADC}/f_{ADCD} = \text{number of } f_{ADC} \text{ clock cycles per arbitration slot}$  ( $N_D = 1 \dots 4$ , as defined in bit field  $DIVD$ ),
- c)  $N_{AR} = \text{number of arbitration slots per arbitration round}$  ( $N_{AR} = 4, 8, 16, \text{ or } 20$ , 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  $INPCR_x$  (for  $c_c$ ) and  $INPCR_y$  (for  $c_i$ ) to the same value.

### Recommendation 2

Select the parameters in register  $GLOBCTR$  according to the following relation:

$$N_A \leq N_D * (N_{AR} + 3).$$

### **ADC TC.H013 Broken Wire Detection not implemented in Emulation Device TC178xED**

The Broken Wire Detection feature that is implemented in the BA-step of the TC178x production devices (PD) is not implemented in the AC-step of the related TC178xED emulation devices.

Therefore, this feature can be used in the PD, but it can not be emulated in an identical manner in the related ED. E.g. data access error traps will occur on TC178xED AC-step while trying to access the BWDENR and BWDCFRG register addresses with software designed for the TC178x BA-step production device.

### **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.

```
...  
mfcrr    d15, #0xFE00  
bistr    #<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>
mfcrr    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 strictly 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();
```

### **CPU TC.H008 Instruction Memory Range Limitations**

To ensure the processor cores are provided with a constant stream of instructions the Instruction Fetch Units will speculatively fetch instructions from up to 64 bytes ahead of the current Program Counter (PC).

If the current PC is within 64 bytes of the top of an instruction memory the Instruction Fetch Unit may attempt to speculatively fetch instructions from beyond the physical range. This may then lead to error conditions and alarms being triggered by the bus and memory systems.

#### **Recommendation**

It is therefore recommended that either the MPU is used to define the allowable executable range or that the upper 64 bytes of any memory be unused for instruction storage for the TC1.6.\* class processors. For TC1.3.\* class processors this may be reduced to 32 bytes.

### **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<sub>H</sub> 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 ( $TEST1.AO_x=0$  -> bus idle)
- set Test Multiplexer Control to I/O Test Mode ( $TEST1.TMC=2$ ), simultaneously  $TXD_x=TXEN_x=0$
- wait for activity ( $TEST1.AO_x=1$  -> bus not idle)
- set Test Multiplexer Control back to Normal signal path ( $TEST1.TMC=0$ )
- wait for no activity ( $TEST1.AO_x=0$  -> 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.

**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.

## **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*`,<sup>1)</sup> 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  $f_{\text{GPTA}}$ , 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  $\geq \max(\text{icu}, (N-1)*\text{FPIDIV})$  for FDR in Normal Mode, or

(1b) READ - ACK  $\geq \max(\text{icu}, N*\text{FPIDIV})$  for FDR in Fractional Mode

with:

- READ = number of  $f_{\text{CPU}}^{1)}$  or  $f_{\text{PCP}}$  cycles between interrupt request (at CPU/PCP site) and register  $\text{SRS}^*$  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  $\text{SRS}^*$  is directly located at entry point in Interrupt Vector Table in CPU Interrupt Service (sub-)routine
  - READ =  $R_1 = 14$  if instruction reading  $\text{SRS}^*$  is first instruction in CPU Interrupt Service (sub-)routine
  - Read =  $R_p = 16$  if instruction reading  $\text{SRS}^*$  is first instruction in PCP channel program
  - $R_x$ : number of extra  $f_{\text{CPU}}$  or  $f_{\text{PCP}}$  cycles to be added to  $R_0$ ,  $R_1$ , or  $R_p$ , respectively, in case instruction reading  $\text{SRS}^*$  is not the first instruction in the corresponding service routine.
- ACK = number of  $f_{\text{CPU}}$  or  $f_{\text{PCP}}$  cycles between interrupt request (at CPU/PCP site) and clearing of request flag  $\text{SRR}$ 
  - 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  $\text{ICR.CONECYC}=1_B$ , icu = 4 with bit  $\text{ICR.CONECYC}=0_B$
- N = "maximum integer value" of clock ratio  $f_{\text{FPI}} / f_{\text{GPTA}}$ 
  - N = 1024 - STEP for Normal Divider mode ( $\text{DM} = 01_B$ )
  - N =  $(1024 \text{ DIV STEP}) + 1$  for Fractional Divider mode ( $\text{DM} = 10_B$ ), where DIV means "integer division"
- FPIDIV = clock ratio  $f_{\text{CPU}} / f_{\text{FPI}}$  for CPU and  $f_{\text{PCP}} / f_{\text{FPI}}$  for PCP

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

**Example 1**

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

This results in:

$$16 - 7 \geq \max(4, (1024 \text{ DIV } 228 + 1) \cdot 2), \text{ or}$$

$$9 \geq \max(4, (5 \cdot 2)), \text{ or}$$

$$9 \geq \max(4, 10), \text{ where } \max(4, 10) = 10$$

i.e.  $9 \geq 10$  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,  $STEP = 38E_H$ ,  $CONECYC = 0_B$ ,  $FPIDIV = 2$  ( $f_{PCP} = 2 \cdot f_{FPI}$ )

This results in:

$$16 - 7 \geq \max(4, (1024 \text{ DIV } 910 + 1) \cdot 2), \text{ or}$$

$$9 \geq \max(4, (2 \cdot 2)), \text{ or}$$

$$9 \geq \max(4, 4), \text{ where } \max(4, 4) = 4$$

i.e.  $9 \geq 4$  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  $\geq 1$  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  $\geq 1$  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  $\geq 1$  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  $n \geq Rx + FPIDIV - 1$  NOPs (+ ISYNC for CPU) at the beginning of the service routine to extend the time until SRS\* is read.*

*Referring to Example 1 ( $Rx \geq 1$  cycle is missing,  $FPIDIV = 2$ ),  $n \geq 2$  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.*

### **GPTA\_TC.H005 Switching between GT0 and GT1 in greater equal Compare Mode when $f_{GPTA} < f_{SYS}$**

An unexpected event may occur for a Global Timer Cell GTCK hooked to Global Timer GTx ( $x = 0$  or  $1$ ) when it is re-configured (by any SPB master, i.e. CPU, PCP, DMA, ..) to a compare mode with "greater equal" compare, and the associated timer is switched to GTy ( $y = 1$  or  $0$ ), if all conditions of the following scenario are true:

1. The GPTA kernel clock  $f_{GPTA}$  is slower than the SPB clock, i.e.  $f_{GPTA} < f_{SYS}$ .
2. Timer cell GTCK is currently enabled and in a state that is not expected to cause an event (e.g. capture mode with no edge selected).
3. Register GTCXRk is currently hooked to GTx, and its write protection is disabled, i.e. GTCXRk has been read some time after it was locked the last time by a capture after compare in one shot mode.

4. Register GTCXRk and GTCCTRk are updated with back-to-back SPB write accesses (steps a) and b) in the following typical sequence):
  - a) Write GTCXRk with new value y; characteristics of value y:
    - greater equal comparison of GTx contents with value y (new GTCXRk contents) = true;
    - greater equal comparison of GTy contents with value y = !true
  - b) Write GTCCTRk to enable cell GTck in a compare mode with greater equal compare (GTCCTRk.GES = 1<sub>B</sub>), and to change the hooked Global Timer to GTy

After this sequence, an unexpected event is generated for GTck (service request, output signal, action request, etc., depending on the configured compare mode), if there was no corresponding GPTA clock to the SPB clock due to the selected  $f_{GPTA} / f_{SYS}$  ratio (clock clipping), such that the greater equal comparison is erroneously based on GTx instead of GTy. In case capture after compare and one shot mode is programmed (GTCCTRk.CAC = 1<sub>B</sub>, GTCCTRk.OSM = 1<sub>B</sub>), GTCXRk will be in write-protected state.

*Note: The present Application Hint GPTA\_TC.H005 refers to the status where cell GTck is **enabled** (see 2. above).*

*For **AUDO NextGeneration** devices, an additional complementary Application Hint GPTA\_TC.H003 exists where GTck is in **disabled** status (GTCCTRk.CEN = 0<sub>B</sub>). While in disabled status when switching the hooked Global Timer, if the recommendations given in GPTA\_TC.H003 are considered, the effect described in this Application Hint GPTA\_TC.H005 will not occur.*

## Recommendations

To avoid the unexpected event for GTck, follow one of the following recommendations:

### Recommendation 1

Configure GPTA kernel frequency  $f_{GPTA} =$  SPB (system clock) frequency  $f_{SYS}$ , i.e. set FDR.DM = 01<sub>B</sub>, FDR.STEP = 3FF<sub>H</sub>.

## Recommendation 2

Change the hooked Global Timer from GTx to GTy while GTCk is in a state that will not cause an event (see step 1 in the following typical sequence):

1. Write GTCCTRk: configure GTCk into a state that will not cause an event (e.g. capture mode with no edge selected), and also change the hooked Global Timer to GTy.
2. Read GTCXRk to ensure write protection is disabled.
3. Write GTCXRk with new value y (see also Note \*) below).
4. Write GTCCTRk to enable the cell and configure GTCk to the desired compare mode for the associated timer GTy.

*Note: \*) In case value y is in the past (“before window”) relative to the current contents of GTy, no immediate greater equal compare event will be generated after GTCk is enabled in step 4. If an immediate action is required for “past” values, repeat the write to GTCXRk from step 3 after step 4.*

## Recommendation 3

*Note: This recommendation is only valid for the clock ratio  $f_{SYS} > f_{GPTA} \geq f_{SYS}/2!$*

Insert a dummy SPB read instruction between the instructions writing to GTCXRk and GTCCTRk (see step 3 in the following typical sequence):

1. Read GTCXRk to ensure write protection is disabled.
2. Write GTCXRk with new value y (see also Note \*\*) below).
3. Perform a dummy read from any SPB register (e.g. GTCCTRk) in order to prevent consecutive back-to-back SPB write accesses to GTCXRk and GTCCTRk.
4. Write GTCCTRk to enable the cell and configure GTCk to the desired compare mode, changing the hooked Global Timer to GTy.

*Note: \*\*) In case value y is in the past (“before window”) relative to the current contents of GTy, no immediate greater equal compare event will be generated after GTCk is enabled in step 4. If an immediate action is required for “past” values, repeat the write to GTCXRk from step 2 after step 4.*

### **INT\_TC.H003 Safe Cancellation of Service Requests**

In specific cases, there might be the need to cancel a Service Request from a node SRNx before it is serviced by CPU or PCP (e.g. when two interrupt sources share the same Service Request Node).

Depending on the synchronization between CPU/PCP and the arbitration performed by the Interrupt Control Unit (ICU), it may happen that a request from SRNx is still serviced after its request flag SRR has been cleared.

Application Note AP32009 “Safe Cancellation of Service Requests” discusses these aspects in detail, including practical implementation examples. It is available in category ‘Documents’ at [www.infineon.com/32-bit-microcontrollers](http://www.infineon.com/32-bit-microcontrollers) and [www.myInfineon.com](http://www.myInfineon.com).

### **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 `000B`) 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**

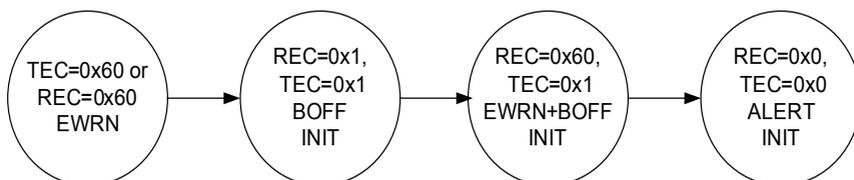
The time stamp measurement feature is not based on an absolute time measurement, but on actual CAN bit times which are subject to the CAN resynchronization during CAN bus operation. The time stamp value merely indicates the number of elapsed actual bit times. Those actual bit times can be shorter or 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 5 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<sub>B</sub>, 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<sub>B</sub>, the node waits for the bus idle state and then sets bit NSR.SUSACK = 1<sub>B</sub>.

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 AI.H009 Behavior of MSGVAL for Remote Frames in Single Data Transfer Mode - Documentation Update**

In Single Data Transfer Mode (SDT = 1<sub>B</sub>), bit MSGVAL is automatically cleared after transmission/reception of a Remote Frame.

The corresponding sections of MultiCAN sub-chapter “Single Data Transfer Mode” of the User’s Manual are copied below, with text updates marked in **bold**:

#### **Message Reception**

When a received message stored in a message object is overwritten by a new received message, the contents of the first message are lost and replaced with the contents of the new received message (indicated by MSGLST = 1<sub>B</sub>).

If SDT is set (Single Data Transfer Mode activated), bit MSGVAL of the message object is automatically cleared by hardware after the storage of a received **Data or Remote Frame**. This prevents the reception of further messages.

#### **Message Transmission**

When a message object receives a series of multiple remote requests, it transmits several Data Frames in response to the remote requests. If the data within the message object has not been updated in the time between the transmissions, the same data can be sent more than once on the CAN bus.

In Single Data Transfer Mode (SDT = 1<sub>B</sub>), this is avoided because MSGVAL is automatically cleared after the successful transmission of a **Data or Remote Frame**.

### **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**

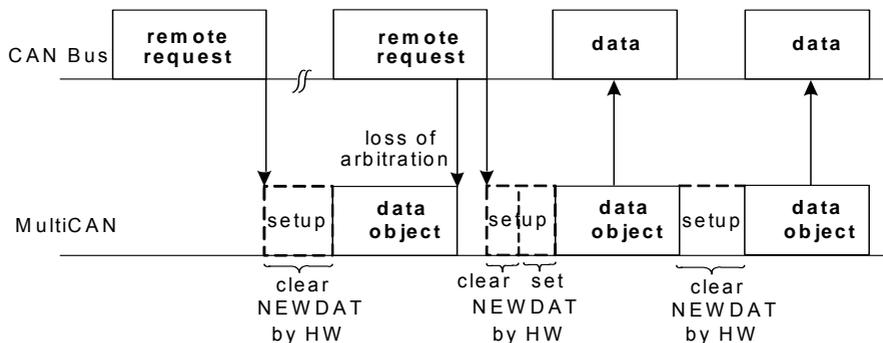
In case the transmission shall be guaranteed, it is not suitable to use the STT mode. In this case, `MOFCRn.STT` shall be 0.

### **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 6 Loss of Arbitration**

### **MultiCAN TC.H005 Documentation Updates related to Controller Area Network Controller (MultiCAN)**

Unintentionally, a wrong version of the MultiCAN chapter has been included in the TC1784 User's Manual V1.1. The following updates need to be considered when working with the MultiCAN module:

#### **1. MultiCAN Module Clock Generation**

MultiCAN Module Clock Generation is based on  $f_{FPI}$ . Therefore, in the text and in all figures within the MultiCAN chapter,  $f_{SYS}$  is to be replaced by  $f_{FPI}$ .

#### **2. Reset Behavior of Flags RXPND and TXPND in Registers MOSTATx**

Text in description of bit fields RXPND and TXPND in Registers MOSTATx to be changed to

- “RXPND is set by hardware and must be reset by software” (instead of “RXPND is not reset by hardware but must be reset by software”)
- “TXPND is set by hardware and must be reset by software” (instead of “TXPND is not reset by hardware but must be reset by software”)

### 3. Error Detection and Correction Test Register EDCR

Register EDCR to be added.

The Error Detection and Correction Register EDCR is a test register which is only writable when the direct RAM access test mode is enabled for the CAN module (SCU\_MEMTEST.MCANDRA = 1<sub>B</sub>).

**Table 12 Registers Address Space - Kernel Registers**

Register Short Name	Register Long Name	Offset Address	Description see
EDCR	Error Detection and Correction Test Register	080 <sub>H</sub>	<a href="#">Table 13</a>

**Table 13 Error Detection and Correction Test Register (080<sub>H</sub>) Reset Value: 0000 0000<sub>H</sub>**

Field	Bits	Type	Description
<b>ECCW</b>	[6:0]	rw	<b>ECC Write Data</b> This data is used as ECC for bus write access to RAM when the error detection and correction is disabled (EDCDIS = 1 <sub>B</sub> ).
<b>EDCDIS</b>	8	rw	<b>Error Detection and Correction Disable</b> 0 <sub>B</sub> Error detection and correction enabled. 1 <sub>B</sub> Error detection and correction disabled.
<b>ECCR</b>	[22:16]	rh	<b>ECC Read Data of last RAM Bus Read Access</b>
<b>SBERR</b>	24	rh	<b>Single Bit Error Flag</b> This bit indicates whether a single bit error has occurred with the last RAM read access.

**Table 13 Error Detection and Correction Test Register (080<sub>H</sub>) Reset Value: 0000 0000<sub>H</sub> (cont'd)**

Field	Bits	Type	Description
<b>DBERR</b>	25	rh	<b>Double Bit Error Flag</b> This bit indicates whether a double bit error has occurred with the last RAM read access.
<b>0</b>	7, 23, [15:9], [31:26]	r	<b>Reserved</b> read as 0, should be written with 0.

#### 4. Bit FSOE (Fast Switch Off Enable) in Register CAN\_CLC

The following footnote is to be added to the description of bit FSOE in register CAN\_CLC:

In Secure shut off mode (FSOE=1), the transmit is brought to recessive state and does not continue an ongoing CAN frame

#### 5. Bit Timing Analysis, CAN Frame Counter (NFCRx.CFC)

The following sentence is to be added to the description of the CAN Frame Counter NFCRx.CFC in section 19.3.7.3 and to the description of register NFCRx in section 19.4.2:

The value of NFCRx.CFC is valid one module cycle later when NFCRx.CFCOV is set.

#### 6. DMA Request Outputs

Text in section 19.5.4.3 (DMA Request Outputs) and Table 19-16 (CAN-to-DMA Request Connections) are to be replaced by the following section:

The interrupt output lines INT\_O0 to INT\_O1 of the MultiCAN module can be used as a DMA requestor and are able to trigger DMA transfers. For connections of interrupt output lines INT\_O[1:0] to the DMA controller, please refer to DMA Chapter 11, Section 11.4.1 DMA Request Wiring Matrix, Table 11-11 DMA Request Assignment for DMA Sub-Block 0.

**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.

### **PORTS TC.H005 Pad Input Registers do not capture Boundary-Scan data when BSD-mode signal is set to high**

The principle of Boundary-Scan is that the BSD-cells can overrule the input and output data for all functional system components (including port-input registers).

In current implementation the peripheral port input registers(P<n>\_IN) are however capturing the direct pad-input data even when the BSD-mode signal is set to high.

This limits the usage of INTEST.

Work around:

In case of INTEST, do not read port input registers.

### **SCU TC.H004 Flag `TRAPSTAT.SYSVCOLCKT` after Power-on or System Reset**

After a power-on (PORST) or System Reset, the PLL VCO loss-of-lock trap request flag `TRAPSTAT.SYSVCOLCKT` is set. As the PLL is not initialized by software at this point in time, the status of this flag is insignificant.

Therefore, flag `TRAPSTAT.SYSVCOLCKT` should be cleared by setting bit `TRAPCLR.SYSVCOLCKT`. Otherwise, an NMI trap will be generated when the corresponding source is enabled in register `TRAPDIS`.

### **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

### DMA TC.D001 Register `DMA_CLRE`

This documentation update refers to TC1784 User's Manual V1.1 2011-05,

- page 11-72 (= page 890 of 2350 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.

### SSC TC.D002 Documentation Updates related to SSC Signal Description

This documentation update refers to column “Pin Functionality” in table “Port 10 Functions” of the Ports chapter of the User's Manual, and to column “Function” of table “Pin Definitions and Functions” in the Data Sheet.

An update with corrections marked in **bold** is shown in the following [Table 14](#).

**Table 14 Update to SSC Signal Description in User's Manual (UM) and Data Sheet (DS)**

Pin	Symbol	I/O	Pin Functionality	Affected Document
P5.5	MRST2	O	SSC2 <b>Slave</b> Transmit <b>Output</b> (Slave Mode)	TC1782 DS, TC1746 DS
P5.7	SCLK2A	I	<b>SSC2</b> Clock Input (Slave Mode)	TC1784 DS
	SCLK2	O	<b>SSC2</b> Clock Output (Master Mode)	
P10.0	MRST2B	I	SSC2 Master Receive Input ( <b>Master</b> Mode)	TC1784 UM
	MRST2	O	SSC2 <b>Slave</b> Transmit <b>Output</b> ( <b>Slave</b> Mode)	
P10.1	MTSR2B	I	SSC2 Slave Receive Input ( <b>Slave</b> Mode)	TC1784 UM

**Table 14 Update to SSC Signal Description in User’s Manual (UM) and Data Sheet (DS) (cont’d)**

Pin	Symbol	I/O	Pin Functionality	Affected Document
	MTSR2	O	<b>SSC2 Master Transmit Output (Master Mode)</b>	TC1784 DS, TC1784 UM
P10.2	SCLK2B	I	<b>SSC2</b> Clock Input (Slave Mode)	TC1784 DS
	SCLK2	O	<b>SSC2</b> Clock Output (Master Mode)	