AURIX TC3xx Safety Manual

AURIX TC3xx Microcontroller Familly

v1.11
2020-10-29

Preface

Cautions

Please read this manual carefully and be sure you understand the information provided before integrating the Infineon product in a safety-related system. This document is the safety manual for the Infineon 32-bit TriCore Microcontrollers of the AURIX^{TM} TC3xx family. Although this is a document for the complete AURIX^{TM} TC3xx family, updates may target only a subset of the devices of the AURIX^{TM} TC3xx family. The Table below lists each of the supported devices together with the safety manual version history.

Table 1 Document Relevant Changes

Device	Version - Date	relevant changes
TC39x-B TC38x-A	1.0 - E10/18	Initial Release
TC39x-B TC38x-A	1.01 - E12/18	see Revision History
TC39x-B TC38x-A	1.02 - E12/18	see Revision History
TC39x-B TC38x-A	1.03 - E01/19	see Revision History
TC39x-B TC38x-A TC35x-A	1.04 - E07/19	see Revision History
TC39x-B TC38x-A TC37xEXT TC35x-A	1.05 - E10/19	see Revision History
TC39x-B TC38x-A TC37xEXT TC35x-A	1.06 - E11/19	see Revision History
TC39x-B TC38x-A TC37xEXT TC37x-A TC35x-A	1.07 - E02/20	see Revision History
TC39x-B TC38x-A TC37xEXT TC37x-A TC35x-A	1.08 - E03/20	see Revision History

Table 1 Document Relevant Changes (continued)

Device	Version - Date	relevant changes
TC39x-B TC38x-A TC37xEXT TC37x-A TC36x-A TC35x-A	1.09 - E04/20	see Revision History
TC39x-B TC38x-A TC37xEXT TC37x-A TC36x-A TC35x-A TC33xEXT	1.10 - E07/20	see Revision History
TC39x-B TC38x-A TC37xEXT TC37x-A TC36x-A TC35x-A TC33xEXT TC33x-A TC32x-A	1.11 - E10/20	see Revision History

It is the responsibility of the system integrator to ensure that the AURIX^{TM} TC3xx microcontroller hardware is suitable for the chosen application and comply with the appropriate application standards. This manual provides information and requirements for the use of the AURIX^{TM} TC3xx in safety-related systems:

- The assumptions of use specified in this manual shall be fulfilled by the system integrator. If the system integrator chooses not to implement an assumption, he has to provide an equivalent measure, or an argumentation that the assumption is not applicable for the considered system, or assess the impact of the not fulfilled assumption on the safety metrics.
- The recommendations are either proposals for the implementation of an assumption or qualitative measures usually implemented in critical systems for robustness, availability, or reliability reasons. The system integrator can choose not to fulfil a recommendation.

Please consult Infineon if you have any questions or comments relating to the information provided in this document.

Intended Audience

This manual is written for system engineers and software engineers and functional safety managers involved in the design or development of a safety-related system who are considering integrating the AURIX™ TC3xx microcontroller hardware as a safety element out of context *SEooC* into their system.

It is assumed by Infineon, that all relevant information, especially documented safety anomalies from AURIX™ TC3xx Microcontroller Safety Case Report chapter 8 table 10 (Open Problem List) are communicated internally by the product system integrator to all relevant stakeholders.

Scope and Purpose

This manual describes the product safety mechanism and the actions that shall be taken by the product system integrator to ensure the correct operation of the device. The purpose of this manual is to support the validation of the product integration regarding the usage of safety mechanisms. The description is limited to those technical actions that are required to ensure compliance with the relevant safety standards. Other documents are referenced for information that is deemed outside the scope of this manual. These documents can be found on the Infineon website www.infineon.com, or are available on request.

ISO 26262

The AURIX™ TC3xx microcontroller is intended to be used as an element of an electrical and/or electronic (E/E) system as defined by the ISO 26262 standard. It was developed as a *SEooC* in accordance with the ISO 26262-10.

The scope of the SEooC comprises:

- The AURIX™ TC3xx microcontroller hardware component.
- Assumptions of use related to the software elements that Support the integration to the AURIX™ TC3xx microcontroller hardware components in a safety related application.
- Assumptions of use related to the hardware environment including assumed external safety mechanisms.
- Assumptions of use related to the software environment.
- Assumptions of use related to the use of the safety mechanisms provided by the SEooC.
- Assumptions of use related to the phases of the SEooC.

This document is part of the safety documentation according to ISO 26262-10 A.3.10 and collects the following information:

- The description of the microcontroller safety architecture with an abstract description of microcontroller functionalities and description of safety mechanisms.
- The description of Assumptions of Use of the microcontroller with respect to its intended use.
- The description of the microcontroller configuration and related hardware and/or software procedures to control a failure after its detection.

Completing the actions described in this document will only satisfy some of the requirements defined by ISO 26262 for safety-related applications. It will be necessary to satisfy the full requirements of the ISO 26262, to use the AURIX™ TC3xx microcontroller in safety-related applications.

Table of contents

Table of contents

Table of contents

1 About this Document

1.1 Glossary Terms

In this manual the words in italic (e.g. *MCU*) are defined in the glossary. In order to avoid any kind of confusion, please refer to the definition provided in this manual only.

1.2 Safety Mechanisms Identifiers

A safety mechanism is defined as an activity or a technical solution to avoid or control systematic failures and to detect random hardware failures or control random hardware failures. While systematic failures can only be mitigated during the design phase, random hardware failures can be detected during the operation by one or several safety measures, which are classified in two main types:

- Safety Mechanism (abbreviation *SM*): technical solution internal to the *MCU* by HW or SW.
- External Safety Mechanism (abbreviation *ESM*): technical solution either in HW or SW, implemented at *System level* by the system integrator.

These two categories are used to mitigate single points and residual faults for single-point fault metric and to avoid dual-point faults from being latent for latent fault metric.

To distinguish these categories, the following conventions are used in this document:

- **SM[HW]:FB.FSB:DESCRIPTION**

SM[HW] refers to a safety mechanism that is implemented by the MCU hardware. FB (and FSB if present) refers to the functional block (and functional sub-block) of the MCU that performs the monitoring function. DESCRIPTION gives a short information about the mechanism.

- **SM[SW]:FB.FSB:DESCRIPTION**

SM[SW] refers to a safety mechanism implemented by software provided by Infineon in addition of the MCU hardware product. FB (and FSB if present) refers to the functional block (and functional sub-block) of the MCU that is monitored by the safety mechanism. DESCRIPTION gives a short information about the mechanism.

- **ESM[HW]:FB.FSB:DESCRIPTION**

ESM[HW] refers to an external safety mechanism to be implemented by the application hardware. FB (and FSB if present) refers to the functional block (and functional sub-block) of the MCU that relies on the external mechanism. DESCRIPTION gives a short information about the mechanism.

- **ESM[SW]:FB.FSB:DESCRIPTION**

ESM[SW] refers to an external safety mechanism to be implemented by *Application SW*. FB (and FSB if present) refers to the functional block (and functional sub-block) of the MCU that relies on the external mechanism. DESCRIPTION gives a short information about the mechanism.

Additionally the following term is used in the manual for describing the required actions for correctly configuring a safety mechanism.

- Safety Mechanism Configuration (abbreviation *SMC*): Initialization or configuration that the Application SW shall perform for enabling a safety mechanism used in the application.

The safety mechanism configuration does not increase diagnostic coverage of fault metrics, it is an hint given to the system integrator for the correct initialization of a given safety mechanism.

- **SMC[SW]:FB.FSB:DESCRIPTION**

SMC[SW] describes all initialization and configuration procedures that shall be implemented by Application SW for enabling the correct operation of a given safety mechanism. FB (and FSB if present) refers to the functional block (and functional sub-block) of the MCU which is configured. DESCRIPTION gives a short information about the configuration.

1.3 Safety Mechanisms table

Safety mechanism items are listed in alphabetical order in chapter 6. Each element is described in a table which includes the following fields:

- **IDENTIFIER**

the name given to the SM[]. Each safety mechanism has a unique identifier which follows the structure showed in the previous paragraph.

- **DESCRIPTION**

gives an overview of the monitoring function performed by the SM[], including the event that will provoke a reaction to a fault.

- **INIT CONDITIONS (OPTIONAL)**

Description of required configuration or link to the SMC[] for the initialization of the SM[].

- **RUNTIME CONDITIONS (OPTIONAL)**

link to the ESM[] for the operations to be performed by *Application SW* during runtime. This field is used when the SM[] does not provide an *SMU* alarm as notification.

- **TESTS (OPTIONAL)**

link to the ESM[] for Application SW operations to be performed for latent fault tests.

1.4 External Safety Mechanism table

External safety mechanism items are listed in alphabetical order in chapter 5. Each element is described in a table which includes the following fields:

- **IDENTIFIER**

the name given to the ESM[]. Each safety mechanism configuration has a unique identifier which follow the structure showed in the previous paragraph.

- **DESCRIPTION**

Provide a description of the action that the system integrator shall implement at SW or HW level for reaching the desired fault coverage. In this field the user finds requirements in order to configure the *MCU* and/or correctly monitor or react to a fault during runtime, in case an *SMU* alarm is not provided by HW.

- **NOTES (OPTIONAL)**

Additional comments that helps the system integrator understanding the requirements provided in the description field.

- **REFERENCES (OPTIONAL)**

Related chapters in User Manual or other documents.

- **RECOMMENDATIONS (OPTIONAL)**

Additional guidance or advice for the correct implementation of the ESM.

- **RELATED SAFETY MECHANISM (OPTIONAL)**

Link to safety mechanism that rely on the ESM[].

1.5 Safety Mechanism Configuration Table

Safety mechanism configuration items are listed in alphabetical order in chapter 5. Each element is described in a table which includes the following fields:

- **IDENTIFIER**

the name given to the SMC[]. Each safety mechanism has a unique identifier which follow the structure showed in the previous paragraph.

- **DESCRIPTION**

Provide a description of the steps required for correctly configuring the related SM[].

- **NOTES (OPTIONAL)**

Additional comments that helps the system integrator understanding the requirements provided in the description field.

- **REFERENCES (OPTIONAL)**

Related chapters in User Manual or other documents.

RELATED SAFETY MECHANISM

Link to safety mechanism for which the SMC[] applies.

2 AURIX TC3xx Product Overview

2.1 Overview

The AURIX™ TC3xx microcontroller combines three powerful technologies within one silicon die, achieving new levels of power, speed, and economy for embedded applications:

- Reduced Instruction Set Computing (RISC) processor architecture.
- Digital Signal Processing (DSP) operations and addressing modes.
- On-chip memories and peripherals.

DSP operations and addressing modes provide the computational power necessary to efficiently analyse complex real-world signals. The RISC load/store architecture provides high computational bandwidth with low system cost. On-chip memory and peripherals are designed to support even the most demanding high-bandwidth real-time embedded control-systems tasks. Additional high-level features of the AURIX™ TC3xx microcontroller include:

- Efficient memory organization: instruction and data scratch memories, caches.
- Serial communication interfaces - flexible synchronous and asynchronous modes.
- Multiple channel *DMA* Controller - DMA operations and interrupt servicing.
- Flexible interrupt system - configurable interrupt priorities and targets.
- Hardware Security Module.
- Flexible *CRC* Engine.
- General-purpose timers.
- High-performance on-chip buses.
- On-chip debugging and emulation facilities.
- Flexible interconnections to external components.
- Flexible power-management.

The AURIX™ TC3xx microcontroller is a high-performance microcontroller with up to six TriCore CPUs, program and data memories, buses, bus arbitration, interrupts system, DMA controller and a powerful set of on-chip peripherals. The AURIX™ TC3xx microcontroller is designed to meet the needs of the most demanding embedded control systems applications where the competing issues of price/performance, real-time responsiveness, computational power, data bandwidth, and power consumption are key design elements.
The AURIX™ TC3xx microcontroller offers several versatile on-chip peripheral units such as serial controllers, timer units, and analog-to-digital converters. Within the AURIX™ TC3xx microcontroller, all these peripheral units are connected to the TriCore CPUs and system via the System Peripheral Bus and the Shared Resource Interconnect. A number of I/O lines on the AURIX™ TC3xx microcontroller ports are reserved for these peripheral units to communicate with the external world.

The following figure shows the block diagram of an AURIX™ TC3xx microcontroller TC39x-B derivate. Please note that not all features that are shown in the block diagram are available in all TC3xx package variants.

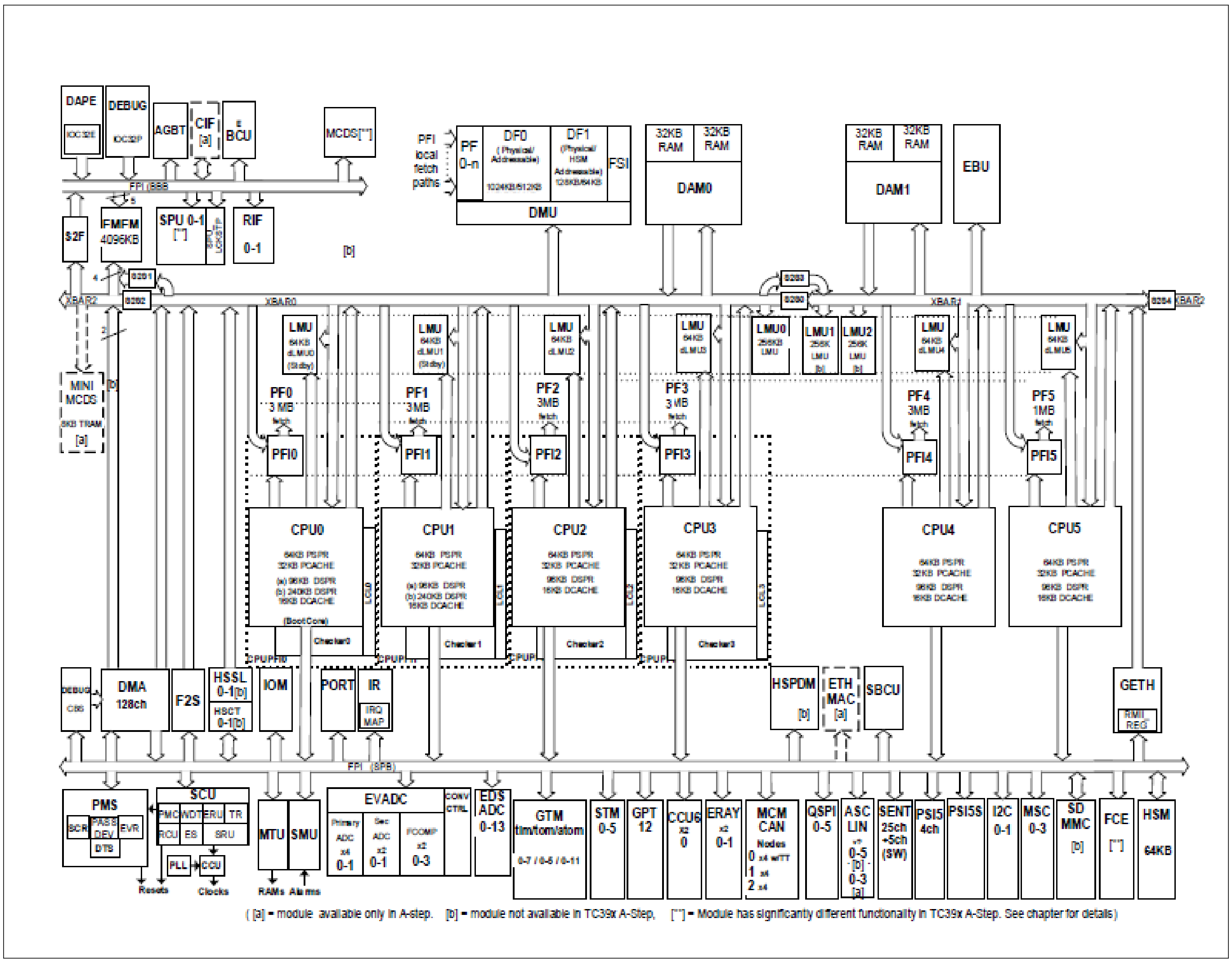


Figure 1

Figure 2

3 Assumptions on System Level

3.1 General Safety Definitions and Terms

3.1.1 Faults

According to ISO 26262-1 a fault is "an abnormal condition that can cause an element or an item to fail". Faults can be classified in the following groups:

- **SINGLE POINT FAULT - SPF (ISO26262-1.122)**

"A single point fault is a fault in an element that is not covered by any safety mechanism and leads directly to the violation of a safety goal". *SPF* are the most critical faults and during the design they need to be identified and their harmful effect reduced by introducing safety mechanisms.

- **MULTIPLE POINT FAULT - MPF (ISO26262-1.77)**

"Individual fault that, in combination with other independent faults, leads to a multilple-point failure". MPFs with higher order than 2 (dual-point fault) are, unless explictly stated, considered as safe faults.

- **LATENT FAULT - LF (ISO26262-1.71)**

"Multiple-point faults whose presence is not detected by a safety mechanism nor by the Driver within the multiple-Point fault detection interval". A fault in a safety mechanism is always classified as a *LF*, since it can violates a safety goal only in combination of another independent fault.

- **RESIDUAL FAULT (ISO26262-1.96)**

"Portion of a fault that by itself leads to the violation of a safety goal, occurring in a HW element where that portion of the fault is not covered by any safety mechanism". For example, if a safety mechanism coverage for a given failure generated by a fault is 90%, the remaining 10% is the residual fault. The existence of residual faults lead to the existence of residual risk, which is the risk remaining after the deployement of safety measures.

- **SAFE FAULT (ISO26262-1.101)**

"Fault whose occurrence will not significantly increase the probability of a violation of a safety goal". A single-point fault, a residual fault or a dual-point fault are not safe faults.

ISO26262-5 requires analytical evidence (e.g. using FMEDA) that the safety requirements of the HW architecture meet the required SPF and LF metrics (shown in the table below):

ASIL	SPFM	LFM
A	Not Relevant	Not Relevant
B	≥90%	≥60%
C	≥97%	≥80%
D	≥99%	≥90%

3.1.2 Failures

According to ISO 26262-1 a failure is the "Termination of the ability of an element to perform an action as required". A Failure can be intended as the result of a fault that has not been detected or correctly handled by an element in a safety context. The ISO 26262 standard vocabulary defines several terms for describing the different kinds of failures:

- **CASCADING FAILURE (ISO26262-1.13)**

"Failure of an element of an item causing another element or element of the same item to fail".In a CF, the failure of the second element is given by the failure of the first element, and not by the root cause itself.

Cascading Failures are dependent failures that are not Common Cause Failures. The two elements can be inside the same channel or in different channels.

- **COMMON CAUSE FAILURE (ISO26262-1.14)**

"failure of two or more elements of an item resulting from a single specific event or root cause". Common Cause Failures are Dependent Failures that are not Cascading Failures.

- **DEPENDENT FAILURES (ISO26262-1.22)**

"Failures whose probability of simultaneous or successive occurrence cannot be expressed as the simple product of the unconditional probabilities of each of them". Dependent failures include common cause failures and cascading failures.

- **MULTIPLE-POINT FAILURES (MPF) (ISO26262-1.76)**

"failure resulting from the combination of several Independent faults which leads directly to the violation of a safety goal". The violation of a safety goal for a MPF implies the presence of all independent faults. Dual-point failures are MPF of order of 2.

- **SINGLE-POINT FAILURE (SPF) (ISO26262-1.121)**

"failure that results from a single-point fault and leads directly to the violation of a safety goal".

- **FAILURE IN TIME (FIT)**

"the number of failures that can be expected in one Billion (10^9) device-hours of operation". FIT is the unit of measurement for "Probabilistic Metric for Random HW failures".

3.1.3 Relationships between faults, errors and failures

The terms faults, errors and failures are strictly related. In general, a fault generates an error that leads the element to loose the capability to perform a function as required. Errors (and failures) can have two different causes:

- **SYSTEMATIC (HARDWARE OR SOFTWARE)**

due to design or specifications issues, these errors are present in all devices of a given family and are reproduceable and deterministic. Software for its nature can only generate systematic errors. They can only be solved by a change in the design process.

- **RANDOM HARDWARE**

caused by particular physical conditions (temperature, age degradation, EM radiation, etc), they follow probabilistic distribution and therefore associated risk can be calculated and mitigated. They can be permanent, intermittent or transient failures.

It must be taken into account that a failure at component level (e.g. *MCU*) is a fault at item level (e.g. vehicle). Therefore, at vehicle level, faults from different causes can lead to the same failure. A failure at vehicle level can lead to hazards if additional factors permit the failure to contribute to an accident scenario. The following figure shows the progression of faults to errors to failures at component level and item level.

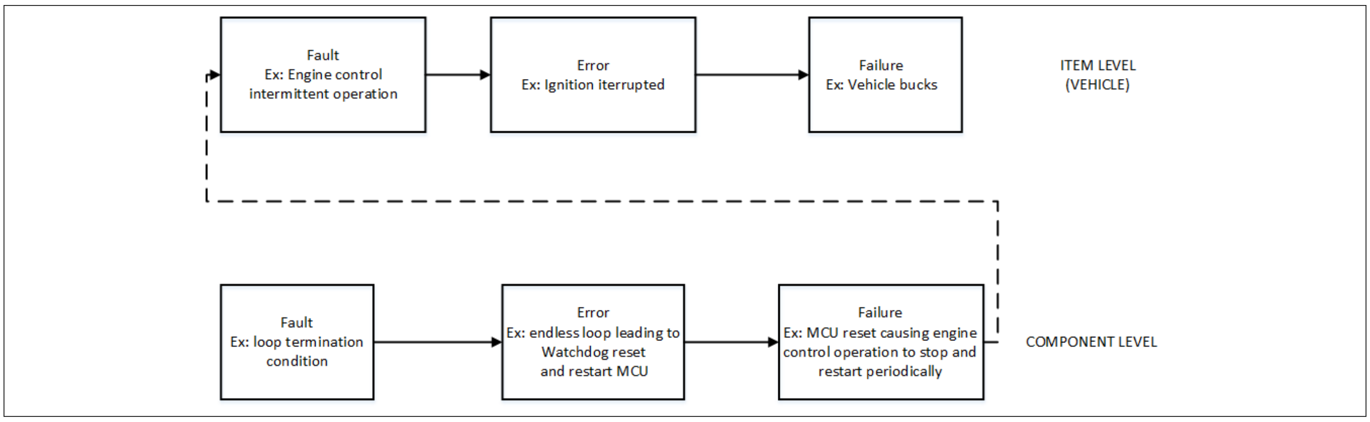

Fault
Ex: Engine control intermittent operation
Error
Ex: Ignition iterrupted
Failure
Ex: Vehicle bucks
ITEM LEVEL (VEHICLE)
Fault
Ex: loop termination condition
Error
Ex: endless loop leading to Watchdog reset and restart MCU
Failure
Ex: MCU reset causing engine control operation to stop and restart periodically
COMPONENT LEVEL

Figure 3

3.1.4 Timing Considerations

The main scope of a safe system is the ability to detect one or several faults and bring the system into a state with a reasonable level of risk (safe state). Therefore, it is crucial that the system reaction to a given fault is faster than the occurrence of possible hazards generated by this fault. The ISO 26262 introduce different terms which define a specific time interval in the context of functional safety.

- **FAULT TOLERANT TIME INTERVAL (ISO26262-1.45)**

"Time span in which a fault or faults can be present in a system before an hazardous event occurs". This quantity identifies how fast the overall system shall react to the presence of a fault. Its value can greatly vary and shall be defined by the system integrator.

- **MULTIPLE-POINT FAULT DETECTION INTERVAL (ISO26262-1.78)**

"Time span to detect multiple-point fault before it can contribute to a multiple-point failure." Since a multiple-point fault *MPF* have the potential to violate the safety goal only when in combination with another MPF, the detection time for MPF is normally defined as a driving cycle (typical 12h).

- **DIAGNOSTIC TEST INTERVAL (ISO26262-1.26)**

"Amount of time between the execution of online diagnostic tests by a safety mechanism". This quantity identifies the maximum time interval needed by a safety mechanism for detecting a fault (either *SPF* or MPF), from the moment the fault occurs. For SW-based safety mechanisms, it represents how often the SW monitoring routine is executed. For HW-based safety mechanisms, it represents the time needed by the HW parts to recognize the presence of the fault and communicate it (e.g. via an alarm, an interrupt, etc.) to the system (including the *Application SW*). In TC3xx family, the *DTI* of HW-based safety mechanisms worst case (which shall be taken as reference for every case) is 125 us.

- **FAULT REACTION TIME (ISO26262-1.44)**

"Time-span from the detection of a fault to reaching the safe state". This time is the maximum of the reaction time of all involved functional safety mechanisms consisting of internal processing time and external indication time.

All the above terms must be considered in the correct context of the MCU as *SEooC*. The Fault tolerant time interval *FTTI* is applicable only at system-level, since it is strictly correlated to the potential presence of a safety hazard. At the same time, the Fault Reaction Time *FRT* meaning differs when applied to the *MCU* or at system-level. As an example, upon the event of a fault, the MCU FRT is considered complete when the configured reaction is performed (e.g. *FSP* activation), while at system-level this time is considered be part of the diagnostic test interval DTI.

The following relation shall always be valid at any point in time:

- FTTI > DTI + FRT

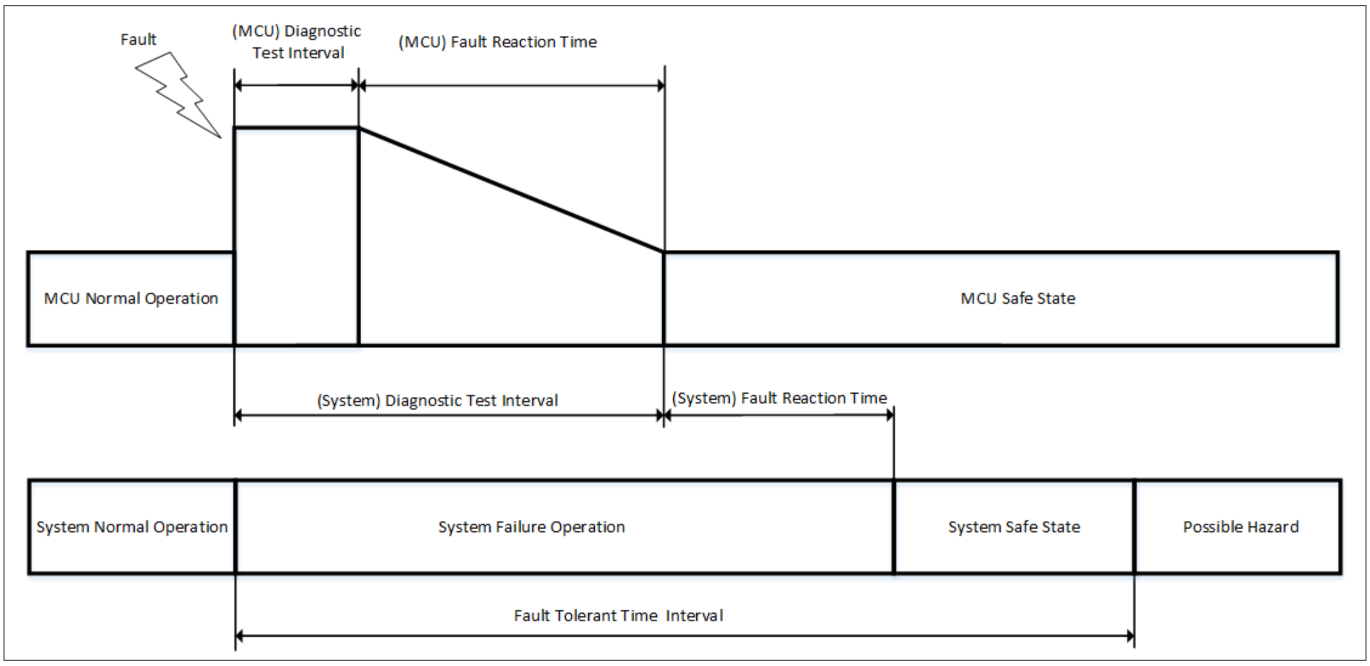

Fault
(MCU) Diagnostic Test Interval
(MCU) Fault Reaction Time
MCU Normal Operation
MCU Safe State
(System) Diagnostic Test Interval
(System) Fault Reaction Time
System Normal Operation
System Failure Operation
System Safe State
Possible Hazard
Fault Tolerant Time Interval

Figure 4

3.2 AURIX TC3xx Operation Overview

3.2.1 Purpose of the SEooC

The AURIX™ TC3xx is an *MCU* developed for various automotive applications. Since its use is not tailored for a specific item, according to ISO 26262-10, the AURIX™ TC3xx is an *SEooC* hardware component. As ISO 26262-10 highlights, the development of an MCU starts with an assumption of the *System level* attributes and requirements. It is the responsibility of the system integrator to integrate in his product the SEooC assumptions of use. According to ISO 26262 classification, the MCU is an hardware component, part of a system which performs a set of functions at item level. A system as it is defined in ISO 26262-1 is composed at least by 3 related elements: a sensor, controller and an actuator. The figure shows the typical use of the AURIX™ TC3xx in the context of an Electronic Control Unit [ECU].

- Inputs are provided by one or more sensors at System level.
- The signals are processed by HW components on the [ECU] and forwarded to the input channels of the microcontroller.
- The microcontroller processes the data and provides outputs to drive one or multiple actuators, or transmits the outputs to another [ECU] via a communication network.

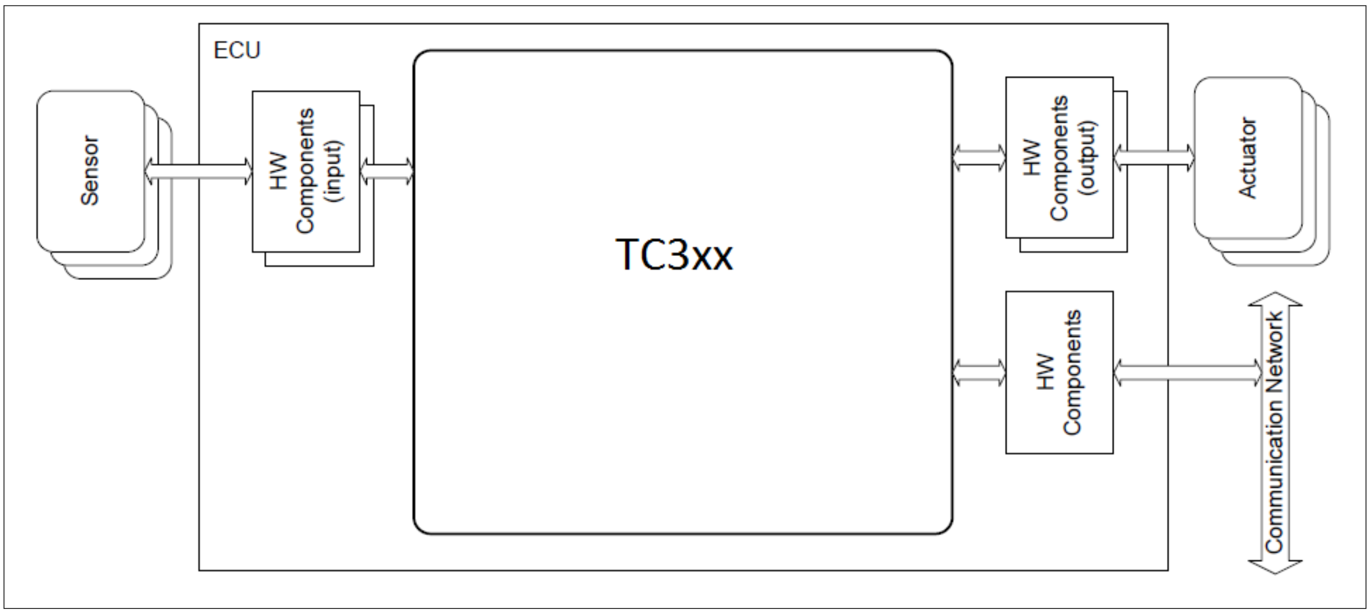


Figure 5

Certain features of the AURIX™ TC3xx microcontrollers are not considered in SEooC development scope as they are intended to be used only during the system development phase. Consequently, in the productive stage of an [ECU], these features shall not be used by the system integrator. The features outlined in the table below shall be considered as “Development-Only” features.

Functional Block	Functions	Function Name	Remarks
GTM	TRIGGER_Data_Generatio n	DTRACE	Interface to trigger tracing of internal GTM signals. Regarded as not being part of any application after development phase is finalized
MCMCAN	Receive	DEBUG_RX	Interface used for debugging purpose to receive DAP frame over CAN frame. Development-only feature as debugging is not part of the safety relevant application assumption
MCMCAN	Transmit	DEBUG_TX	Interface used for debugging purpose to transmit DAP frame over CAN frame. Development-only feature as debugging is not part of the safety relevant application assumption

SCU	Watchdog	WDTDebugSuspend	This function shall only be used during Debug-operation in order to avoid unintended alarms/resets
TRACE	Other interfaces	TRACE	This feature should be used only during development to document and analyze the run-time behavior of a software
DEBUG	All	DEBUG	Debug feature (OCDS, MCDS) shall only be activated and used during system development phase and shall be deactivated before release for production
CPU	OVERLAY	OVERLAY	Overlay is typically used for evaluation of SW calibration parameters. Once the latter is finalized, the feature shall be deactivated
CPU	Data Acquisition	OLDA	Online Data Acquisition feature shall only be used during SW evaluation phase
Firmware	Data Storage in EMEM	EMEM-Prolog	This feature is aimed to be used for calibration purposes. It shall be deactivated together with final storage of SW in PFLASH
CAN/LIN	Code Loading	BSL	This feature is only used for initial device programming. It is not needed anymore once the application software is stable

3.2.2 MCU Reset State

The *MCU* has different sources of reset, depending on the event that triggered the reset. Depending on the reset type, different modules of the MCU are affected. The following list describes the different reset types:

- **COLD POWER-ON RESET**

A Cold Power-on Reset is a reset which is triggered for the first time during a system power-up or in response to a temporary power failure. The pins and internal states are placed immediately into their default state when the trigger is asserted. The system is placed in a defined state, and all registers keep their reset values as long as the reset is asserted. *DSPR*, *PSPR* and *LMU* are re-initialized only after a cold-power on reset. In all other reset types, their content and redundancy are not affected.

- **WARM POWER-ON RESET**

A warm reset is triggered while the system is already operational and the supplies remain stable. It is used to return the system to a known state. On a warm reset request, port pins are immediately placed in default state, all internal peripherals and *CPU* are re-initialized. *PORST* assertion keeps the MCU in warm power-on reset type.

- **SYSTEM RESET**

As for Warm Power-on Reset, this reset leads to an initialization into a defined state of the complete system. This type of reset can be triggered intentionally by the *Application SW* by configuring *SFR*.

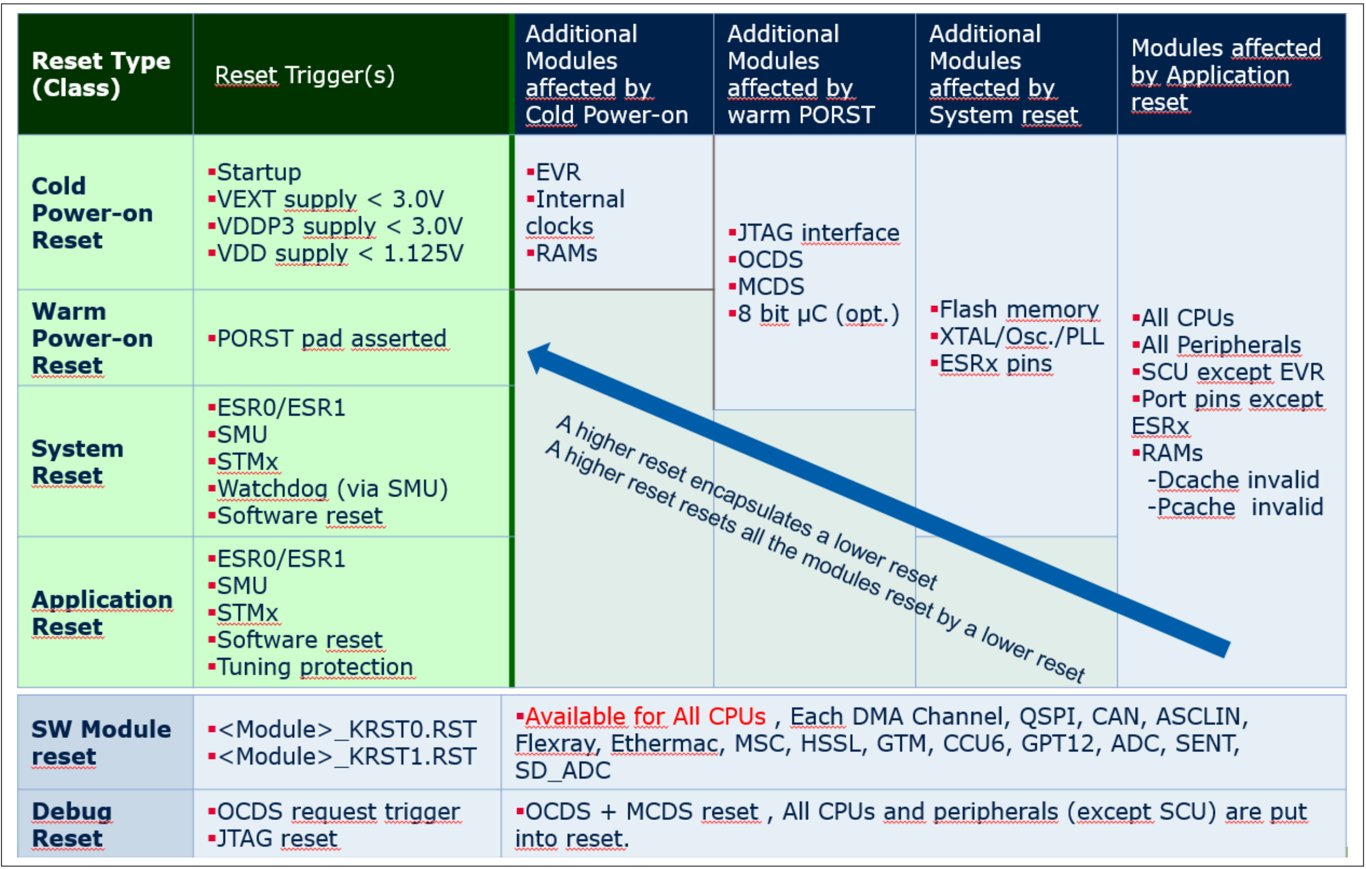
- **APPLICATION RESET**

This reset leads to an initialization into a defined state of the complete application system of all peripherals, all CPU, all pins and part of the *SCU*. As for system reset, the trigger sources can be configured by Application SW.

- **MODULE RESET**

Individual reset commands can be sent by authorized masters to a single module without any impact on the rest of the system.

This figure shows the different kind of resets and their effects on the various parts of the MCU.



Reset Type (Class)	Reset Trigger(s)	Additional Modules affected by Cold Power-on	Additional Modules affected by warm PORST	Additional Modules affected by System reset	Modules affected by Application reset
Cold Power-on Reset	▪Startup ▪VEXT supply < 3.0V ▪VDDP3 supply < 3.0V ▪VDD supply < 1.125V	▪EVR ▪Internal clocks ▪RAMs	▪JTAG interface ▪OCDS ▪MCDS ▪8 bit µC (opt.)	▪Flash memory ▪XTAL/Osc./PLL ▪ESRx pins	▪All CPUs ▪All Peripherals ▪SCU except EVR ▪Port pins except ESRx ▪RAMs -Dcache invalid -Pcache invalid
Warm Power-on Reset	▪PORST pad asserted				
System Reset	▪ESR0/ESR1 ▪SMU ▪STMx ▪Watchdog (via SMU) ▪Software reset				
Application Reset	▪ESR0/ESR1 ▪SMU ▪STMx ▪Software reset ▪Tuning protection				
SW Module reset	▪<Module>_KRST0.RST ▪<Module>_KRST1.RST	▪Available for All CPUs , Each DMA Channel, QSPI, CAN, ASCLIN, Flexray, Ethermac, MSC, HSSL, GTM, CCU6, GPT12, ADC, SENT, SD_ADC			
Debug Reset	▪OCDS request trigger ▪JTAG reset	▪OCDS + MCDS reset , All CPUs and peripherals (except SCU) are put into reset.			

A higher reset encapsulates a lower reset
A higher reset resets all the modules reset by a lower reset

Figure 6

3.2.3 MCU Operating Modes

In any point in time, the *MCU* will be in one of the following modes:

- **COMPLETELY UNPOWERED**

The MCU is not supplied and no energy is provided to the device. All internal circuitry is not active.

- **RESET MODE**

While in this state, the I/O pins are in reset state and the internal modules are kept in a known state and are inactive.

- **POWER-UP MODE**

When the voltage supplies reach a certain threshold, the internal circuitry is activated and all modules and memories are initialized. If initial tests are successful, CPU0 is released from reset and the *SSW* is executed. During this phase the safety mechanisms are initialized and tested by the SSW.

- **RUN MODE**

If all tests are successful, the CPU0 Releases the other *CPU* and the *Application SW* is executed. All safety mechanisms are active. This is the normal state of the MCU during operation.

- **INTERNAL ERROR MODE**

Because of an internal failure, the MCU outputs are potentially dangerous. The user can configure the *FSP* error pin to signal this state.

- **STANDBY MODE**

This mode is entered on a SW request or due to a ESR1 assertion event. In this mode, only the standby controller and the wake-up unit are active. All other peripherals and CPU are switched off. Ports are set in their default state. No safety mechanisms are active.

- **SLEEP MODE**

This mode is entered by SW request. CPU code exection is halted and CPU are held in idle state. Peripherals are in sleep state and ports keep their programmed value. No safety mechanisms are active.

- **DEBUG MODE**

This mode is enabled for allowing the developer to access HW registers and functions during the development phase. In debug mode, the safety mechanisms are active but there is no guarantee that the debug module *OCDS* in the device will not interfere with the HW. Therefore, no safety-relevant applications should run in this mode. The only mode assumed in the safety concept of the MCU is the run mode.

3.2.4 Boot and Startup Procedure

The sequence that the *MCU* follows from an unpowered state can be divided in three different phases, during which several safety mechanisms are executed to guarantee that all safety relevant parts are working correctly. Latent faults tests as prescribed by ISO 26262-5 are also executed during this time.

- **ANALOG POWER-UP**

When the external power supply reaches 2.4V, the internal circuitry is activated and the *PMS* checks the 100 MHz clock. If the clock is stable, then ***SM[HW]:PMS:PBIST*** is automatically executed. Only if this test is successful and VDD,VDDP3 VEXT voltage above respective primary reset thresholds PORST output is de-asserted. This stage is executed only if the device comes from an unpowered state. For all other reset types this phase is skipped.

- **BOOT FIRMWARE**

Immediately after PORST release the system firmware (*FW*) execution is started. The FW resides in BootROM and it is partially configurable by user via the UCB registers. The execution of the FW is performed by the CPU0, while all others *CPU* are held in halt state. Depending on the event that triggered the reset, the FW will execute a complete or partial initialization of the device.

For example, after a cold power-on reset the content of RAM is not defined, so an *MBIST* executed on unitialized memories will fail since data and ECC contents are not coherent. The FW offers the possibility to automatically initialize CPU and *LMU* RAM depending on the value of HF_PROCONRAM.RAMINSEL and HF_PROCONRAM.LMUINSEL registers. ***SM[HW]:MCU:LBIST*** can also be automatically executed by FW after cold power-on reset, as specified in***SMC[SW]:MCU:LBIST_CFG***.

During the execution, the FW performs several checks as specified in ***SM[SW]:FW: MCU_STARTUP_PREOS_SSW***. In case the *SSW* or the *CHSW* encounters a problem, there are two possible kinds of reactions:

- SSW automatically brings the MCU in error state.

- CHSW indicates the detected failure via STMEMx registers. The *Application SW* shall verify the content of these registers during the startup as highlighted in the following paragraph.

- **APPLICATION SW STARTUP**

At the end of FW sequence, the program counter will jump to the first user code instruction and Application SW execution will begin. It is assumed that the Application SW startup is executed from a [Lockstep CPU]. During Application SW startup, the user is responsible for executing a number of operations for ensuring the absence of latent faults and correctly initialize the MCU before starting the runtime execution. In particular the Application SW shall:

- execute ***SM[HW]:MCU:LBIST*** (if not already performed during FW).
- evaluate the result of *LBIST* as described in ***ESM[SW]:MCU:LBIST_RESULT***.
- configure, run and check the result of ***SM[HW]:PMS:MONBIST*** for ensuring the absence of latent faults in the secondary voltage monitors and standby *SMU* alarm path.
- check the correct execution of FW as described in ***ESM[SW]:SYS:MCU_FW_CHECK***.
- verify the correct configuration settings values installed by FW as described in ***ESM[SW]:SYS:MCU_STARTUP***.
- test the functionality of the SMU core alive monitor and the reaction in the stand-by SMU by injecting an error and check the result as described in ***ESM[SW]:SMU:ALIVE_ALARM_TEST***.
- Execute test of all relevant functional blocks as described ***ESM[SW]:SMU:REG_MONITOR_TEST***.
- configure, run and check the result of ***SM[HW]:VMT:MBIST*** for ensuring the absence of faults in RAM. These operations are described in ***SMC[SW]:VMT:MBIST*** and ***ESM[SW]:VMT:MBIST*** respectively.
- ensure to enable all SMU alarms relevant for the application. In particular, the user shall re-enable ALM8[0] and ALM8[3] that shall be disabled during the oscillator and PLL configuration procedure.

The following image shows the sequence of safety mechanisms involved from an unpowered state to a full operational state.

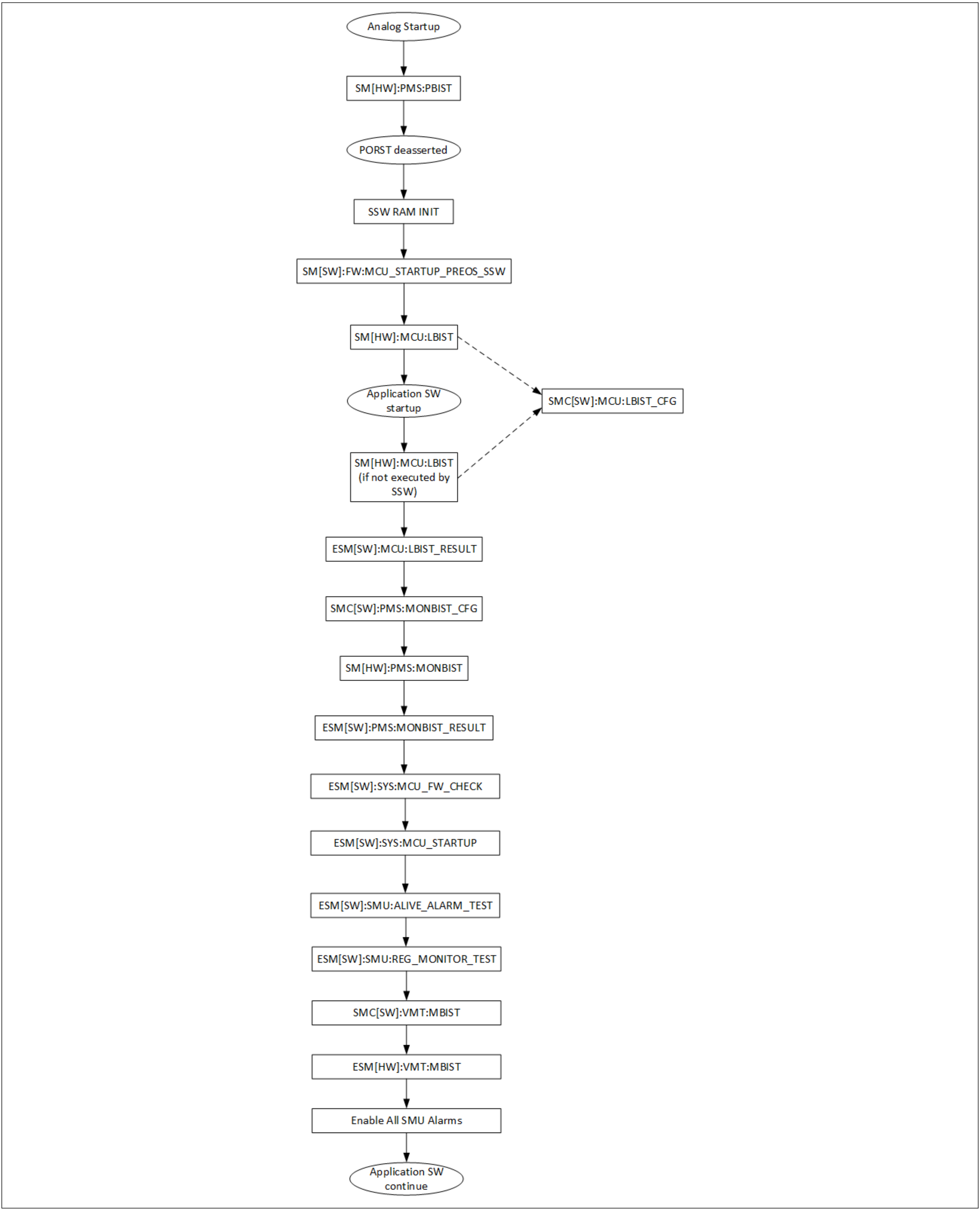

Analog Startup
SM[HW]:PMS:PBIST
PORST deasserted
SSW RAM INIT
SM[SW]:FW:MCU_STARTUP_PREOS_SSW
SM[HW]:MCU:LBIST
Application SW startup
SMC[SW]:MCU:LBIST_CFG
SM[HW]:MCU:LBIST (if not executed by SSW)
ESM[SW]:MCU:LBIST_RESULT
SMC[SW]:PMS:MONBIST_CFG
SM[HW]:PMS:MONBIST
ESM[SW]:PMS:MONBIST_RESULT
ESM[SW]:SYS:MCU_FW_CHECK
ESM[SW]:SYS:MCU_STARTUP
ESM[SW]:SMU:ALIVE_ALARM_TEST
ESM[SW]:SMU:REG_MONITOR_TEST
SMC[SW]:VMT:MBIST
ESM[HW]:VMT:MBIST
Enable All SMU Alarms
Application SW continue

Figure 7

NOTE: Depending on the Application, additional ESM[SW] at startup are needed. For example, in case non Lockstep CPU is used for safety-relevant tasks, then all ESM[SW] regarding non-LS CPU need to be taken into account.

3.2.5 Failure management

In case of a failure detection by a safety mechanism an *SMU* alarm event is generated. The severity of each alarm shall be configured according to the requirements of the safety application: per default every alarm reaction is disabled with the exception of the watchdog timeout and recovery timer alarms. It is responsibility of the system integrator to determine what kind of action to activate in response to a failure. For each alarm in the SMU, one of the following internal actions can be chosen:

- **NO ACTION**

The alarm handling is disabled and no action is taken. This is the default value for all alarms.

- **INTERRUPT REQUEST**

generate an interrupt request to any of the *CPU*s, concurrent interrupts to several CPUs can be configured.

- **NMI TRAP REQUEST**

An *NMI* request is forwarded to the *SCU*, which determines to which CPU issue the trap.

- **RESET REQUEST**

Depending on the configuration selected, an application or system reset request is forwarded to the SCU.

- **CPU RESET REQUEST**

Triggers a reset to one or more CPU.

In parallel, each alarm can trigger one or both of the following external actions:

- **PORT EMERGENCY STOP**

This feature provides a fast reaction without the intervention of SW. The selected output ports are immediately set into a defined state.

- **FAULT SIGNALING PROTOCOL**

This protocol enables the *MCU* to report a fault to an external safety controller in order to control the safe state of the system.

For some of the safety mechanisms, other actions that do not involve the SMU are possible. When available, a safety mechanism can trigger the following actions:

- **INTERRUPT REQUEST**
- **CPU TRAP**
- **APPLICATION SW HANDLER**

The following figure gives an overview of failure management.

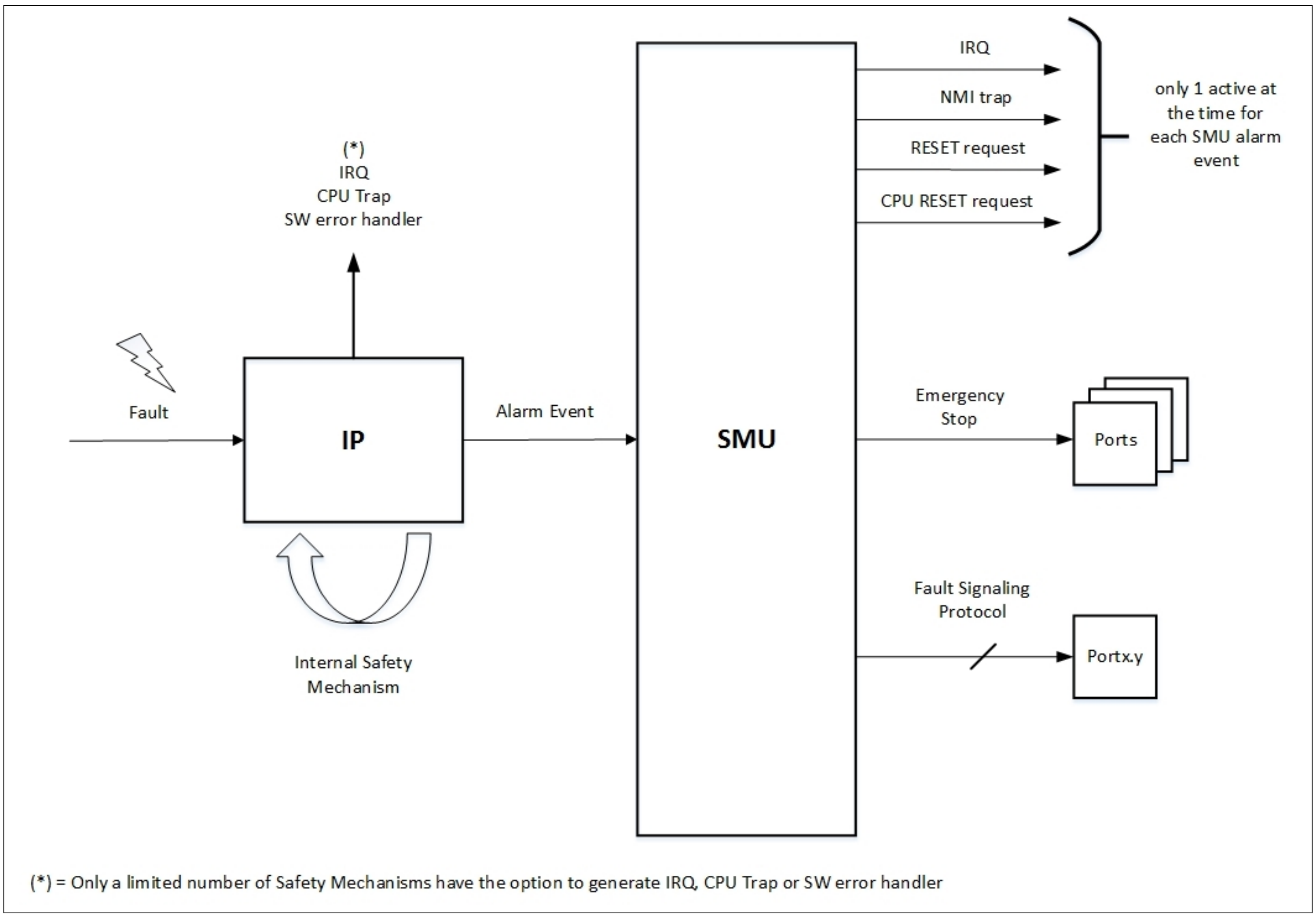

IRQ
NMI trap
RESET request
CPU RESET request
only 1 active at the time for each SMU alarm event
(*)
IRQ
CPU Trap
SW error handler
Fault
IP
Alarm Event
SMU
Emergency Stop
Ports
Internal Safety Mechanism
Fault Signaling Protocol
Portx.y
(*) = Only a limited number of Safety Mechanisms have the option to generate IRQ, CPU Trap or SW error handler

Figure 8

3.3 System Level Hardware Requirements

3.3.1 Introduction

AURIX™ TC3xx has been developed as a *SEooC* for operating in an E/E system as an [ECU]. For its correct functioning, a number of requirements has to be fulfilled by the system integrator, either by *System level* HW elements, *Application SW* routines or design measures. The main purpose of these measures is to ensure the correct operation of the device, and in case of a malfunctioning of the *MCU*, bring the entire system in a safe state.

AURIX™ TC3xx offers several safety mechanisms that can be applied for monitoring the correct behaviour of the MCU and the external power supply. The figure below shows a typical connection of the AURIX™ TC3xx with an external IC providing both external voltage supply and safety monitor.

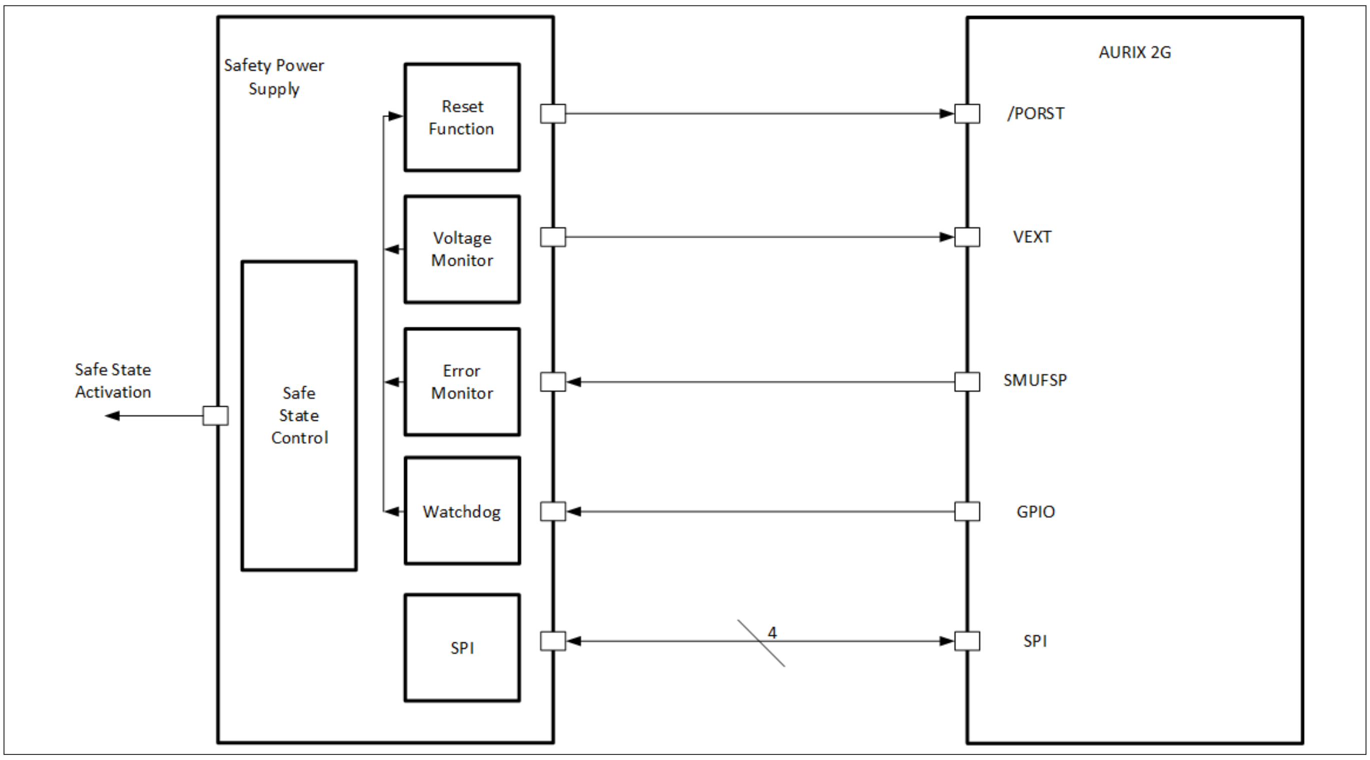


Figure 9

Note: This is only one possible implementation. If it is assumed the external safety mechanisms described in this section are available at System level, they may be distributed over several components or even other [ECU], depending on the system architecture and its capabilities to transition to the system safe state.

As the figure above shows, there are several signals that are used for connecting the microcontroller with the external IC. The external component should have the capability to bring the entire system into a safe state using one or more dedicated outputs, called safe state activation signals. It is the responsibility of the system developer to identify in which scenarios he shall transition the system to a safe state because of a malfunctioning of the MCU.

3.3.2 External Voltage Supply

The *MCU* is an active electronic component, therefore, in order to function correctly, all the operating voltages and operating conditions have to be met during the operation of the device. If the microcontroller is not operating within the specified operating range or the external supply voltages are subject to transients (spikes, oscillations, etc.) the MCU may not function correctly and lead the system to an hazardous state. The following conditions on the external voltage have to be guaranteed for the correct functioning of the device:

- ***ESM[HW]:PMS:VEXT_VEVRSB_ABS_RATINGS***

For the same reason, an external filter shall prevent electromagnetic noise and other sources of disturbances to couple to the supply source and degrade the quality of the supply voltage.

- ***ESM[HW]:PMS:VEXT_FILTER***

An external device shall supervise the supply voltage of the MCU and in case of an overvoltage condition is detected, the system shall disable the supply voltage of the MCU. A deviation above the specified maximum admitted voltage, can lead to permanent failures and physical damage to the device.

- ***ESM[HW]:PMS:VEXT_VEVRSB_OVERVOLTAGE***

The microcontroller supports multiple external supply modes, depending if the external voltage supply provide one or more rails to the MCU.

3.3.3 Error Monitor

Each alarm generated in the *SMU* can be configured to activate one or more of the following measures for signaling an *MCU* malfunction:

- *FSP* activation, described in ***ESM[HW]:SYS:FSP_ERROR_PIN_MONITOR*** (see Errata SMU TC.H013)
- *ES* activation, described in ***ESM[HW]:SYS:ES_ERROR_PIN_MONITOR***
- *Application SW* notification via NMI or ISR, described in ***ESM[HW]:SYS:SW_ERROR_PIN_MONITOR***

3.3.4 External Time-Window Watchdog

During the system development, it shall be considered a scenario where the *MCU* is not able to signal an internal failure. This situation can be generated by different causes, for example if the code execution is delayed or completely stopped, or the complete MCU is not active due to a failure to the power management system. By using an external time-window watchdog to monitor the correct operation of the MCU , it is possible to detect these failures and perform the appropriate reaction at *System level*. The use of an external watchdog reduces common mode failures and enhances the hardware diversity and ability to eliminate potential dependent faults. The following assumption is made:

- ***ESM[HW]:SYS:WATCHDOG_FUNCTION***

4 Architecture for Management of Faults

4.1 HW Self Tests for Latent Fault Metric Support

4.1.1 Introduction

According to ISO 26262, latent faults are undetected *MPF*. These faults can be either:

- an MPF affecting mission logic which cannot be detected nor perceived
- an undetected fault in a safety mechanism

Both categories mentioned above are classified as *LF* and their presence need to be identified according to LF metrics within the "Multiple-Point Fault Detection Interval" (ISO 26262 - 1.78).

TC3xx family offers four different HW-based self tests (BIST) that participate to LF metrics and can be executed automatically by the firmware (if enabled by the user) during boot time or by the *Application SW* during runtime (except PBIST, which is automatically executed after cold power-on reset):

- ***SM[HW]:PMS:PBIST***: Power Built-in Self Test.
- ***SM[HW]:MCU:LBIST***: Logic Built-in Self Test.
- ***SM[HW]:PMS:MONBIST***: Monitor Built-in Self Test.
- ***SM[HW]:VMT:MBIST***: Memory Built-in Self Test.

During the *MCU* runtime, several others safety mechanisms (SM[] and ESM[]) participate to LF metric by detecting *SPF* in the mission logic (ISO26262 - C.6). The combination of startup and runtime safety measures allows the TC3xx familiy to reach the ASIL-D LF target metrics as specified in the ISO26262 (LFM>=90%).

4.1.2 Power Built-In Self Test (PBIST)

A fault in the external power supply can lead the *MCU* to behave in an unpredictable manner and directly lead to the violation of a safety goal. Therefore faults related to supply voltages are treated as single-point faults and several safety mechanisms in the *PMS* are dedicated to monitor during runtime the voltage levels and in case of under- or over-voltage generate an *SMU* alarm. According to ISO 26262-5, a fault in one of these voltage monitors is considered a latent fault, since in combination of a fault in the power supply would lead to an undetected failure in the MCU.

The test of primary power supply is called *PBIST* and identified with the following name in this document:

- ***SM[HW]:PMS:PBIST***

It is executed before cold PORST release and allows the testing of supply levels, power functions and voltage monitors. The internal EVRPR Pre-regulator VDDPD voltage based on the primary low power bandgap (PLPBG) is tested using secondary monitor ADC against the secondary bandgap (SHPBG) at supply ramp-up. This allows to monitor the bandgap voltages against each other during start-up and the device continue to remain in reset state till the test has passed. The overvoltage and undervoltage thresholds are given by the reset values of the EVROVMON, EVROVMON2, EVRUVMON and EVRUVMON2 registers and cannot be changed, so PBIST has no possible configuration from user.

4.1.3 Logic Built-In Self Test (LBIST)

The *LBIST* is a HW safety mechanism identified with the following name:

- ***SM[HW]:MCU:LBIST***

It can be used to detect latent faults in the digital logic of the *MCU*. The LBIST structure applies pseudo-random patterns generated by a PRPG (Pseudo-Random Pattern Generator) to a full-scan circuit in parallel and compacts the test responses into a signature with a MISR (Multiple-InputSignature Register):

The user can configure some test parameters as described here:

- ***SMC[SW]:MCU:LBIST_CFG***.

There are two configurable ways to start LBIST execution: as a part of boot-up sequence or by *Application SW* in MCU functional mode. It must be considered that once the LBIST is started, the MCU state is lost and the device is not functionally available. At the end of the test, LBIST terminates with a system reset. The result of the test can be evaluated by Application SW via RSTSTAT, LBISTCTRL2 and LBISTCTRL3 registers:

- ***ESM[SW]:MCU:LBIST_RESULT***

During LBIST no digital logic is available. External measures shall be implemented for monitoring the correct execution of the LBIST:

- ***ESM[HW]:MCU:LBIST_MONITOR***

Since an unintentional LBIST activation during runtime leads to a loss of the MCU functionality, two SMU alarms are triggered:

- ALM8[5] LBIST alarm: this alarm signal shall be activated if the LBIST-FSM is NOT in the Reset/Idle state or if some global LBIST-enable signals are active. Consequently LBIST-alarm-signal is always active during LBIST execution.
- ALM8[21] test-mode alarm : this alarm signal is activated if any of the TCU's general test-mode enabling signals are in unintended active state. Consequently Test-Mode-Alarm signal is always active if the MCU is operating in general test-mode.

4.1.4 Monitor Built-in Self Test (MONBIST)

To increase the latent fault coverage, an additional test called *MONBIST* is performed after *PBIST* and PORST release:

- ***SM[HW]:PMS:MONBIST***

This test is executed for covering potential faults on secondary voltage monitors and associated alarm paths and *FSP* error pin routed to the stand-by *SMU*. MONBIST shall be configured by *Application SW* as described here:

- ***SMC[SW]:PMS:MONBIST_CFG***

It is able to distinguish failures in the secondary voltage monitors domain or in the standby SMU. At the end of the procedure, the MONBIST results are available and shall be evaluated by Application SW as highlighted here:

- ***ESM[SW]:PMS:MONBIST_RESULT***

4.1.5 Memory Built-in Self Test (MBIST)

Faults on memory cells of a SRAM are considered latent faults in ISO 26262. The AURIX TC3xx offers a HW self-test (*MBIST*) that can be executed on every SRAM instance:

- ***SM[HW]:VMT:MBIST***

As ISO 26262 prescribes, latent faults shall be covered at least once per driving cycle. The user is free to chose when executing the MBIST (start-up, runtime, shut-down). The following preconditions shall be considered:

- Before starting the test, the SRAM shall to be initialized with correct ECC data.
- During the test, the SRAM cannot be accessed.

The module can be configured for performing a test sequence where all addresses defined by a range are read and written following a certain test pattern. The test parameters can be programmed by user CONFIG0, CONFIG1 and MCONTROL registers:

- ***SMC[SW]:VMT:MBIST***

The test results can be checked by *Application SW* in ECCD register:

- ***ESM[SW]:VMT:MBIST***

NOTE: MBIST can be programmed for executing several test patterns. Nevertheless, Infineon provides information on power consumption and duration time only for Non-Destructive Test (NDT).

4.2 Functional Blocks

4.2.1 MCU Function - Processing

4.2.1.1 CPU

4.2.1.1.1 Nominal Functionality

The TC3xx family utilises the TC1.62P core hardware, which is based on TC1.6P core with enhancements regarding memory distribution and protection and other aspects. Additionally, depending on the variant, up to four *CPU* are protected by a lockstep mechanism, which allows to run up to ASIL D applications. From a functional perspective the two CPU categories (lockstep and non-lockstep) offer the same performance.

4.2.1.1.2 Monitoring Concept

CPU memory and time protection

The *CPU* offer several HW measures for protection on memory and resource accesses, as well as timer-based mechanisms for detecting timing violations of SW. These mechanisms are listed in "***Coexistence of HW/SW elements ASIL D***" chapter.

CPU is a slave node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms described in in "***Coexistence of HW/SW elements ASIL D***" chapter.

Lockstep CPU

Depending on the variant, a TC3xx offers up to 4 lockstep CPU. The LS-CPU monitoring is based on hardware redundancy with online monitoring of the outputs. The logic covered by this hardware redundancy is called the CPU area of duplication and comprises:

- TriCore TC1.62P core
- CPU SFR and CSFR
- master and slave interfaces to *SRI*
- master interface to *SPB*
- interface to the interrupt router
- interface to the *SCU*
- interface to the program memories (*PMI*)
- interfaces to the data memories (*DMI*)
- interfaces to the program flash (*PFI*)

The lockstep monitoring function will compare the outputs from the master and checker core and report that a failure has occurred to the *SMU* for appropriate reaction:

- ***.SM[HW]:CPU:TRICORE_LOCKSTEP***

Each lockstep comparator has the capability to run a self-test in order to detect malfunctions in its logic:

- ***SM[HW]:CPU:TRICORE_LOCKSTEP_SCC***

In a similar way, the PFI is protected against *RHF* by a lockstep control logic:

- ***SM[HW]:CPU:CONTROL_REDUNDANCY***

A continuous *BIST* is executed in the PFI lockstep:

- ***SM[HW]:CPU:CONTROL_REDUNDANCY_SCC***

Memory interfaces of non-LS *CPU* are not protected by redundancy schemes. In case the System Integrator wants to use non-LS memories for running ASIL-D applications from a LS-CPU, additional measures are needed at *Application SW* level:

- ***ESM[SW]:CPU:DATA_INTEGRITY***

Swap configuration is monitored during runtime:

- ***SM[HW]:CPU:SWAP_CONFIGURATION_PROTECTION***

Non-Lockstep CPU

The non-LS CPU has an identical architecture compared to the LS CPU, but it does not include the redundant *CPU* (called checker core) and the comparator output logic. While the performances of the non-LS and LS-CPU are the same, non-LS CPU cannot rely on redundant HW elements for *RHF* detection. Therefore a combination of SW-based mechanisms is needed for covering single-point and latent faults.

The SW-*BIST* provided by Infineon is a routine with the specific purpose of providing coverage on permanent faults on the fetch and pipeline units of the non-LS CPU core.

- ***SM[SW]:CPU:SBST***

In combination with the CPU *SBST* safety mechanism, *Application SW* is in charge of executing with the same frequency these ESM:

- ***ESM[SW]:CPU:INTERNAL_BUS_MONITOR***
- ***ESM[SW]:CPU:SFR_TEST***

For latent faults, non-LS CPU relies on Application SW for detection of permanent faults affecting non-LS CPU safety mechanisms logic:

- ***ESM[SW]:CPU:AP_CHECK***
- ***ESM[SW]:CPU:CODE_MPU_CHECK***
- ***ESM[SW]:CPU:DATA_MPU_CHECK***
- ***ESM[SW]:CPU:BUS_MPU_INITCHECK***

Swap configuration is monitored during runtime:

- ***SM[HW]:CPU:SWAP_CONFIGURATION_PROTECTION***

CPU RAM

Each *CPU* utilises different RAM blocks as local memories:

- CPU.DSPR: 96 kB of Data Scratch-Pad SRAM (240kB for CPU0/1)
- CPU.PSPR: 64 kB of Program Scratch-Pad SRAM
- CPU.DCACHE: 16 kB of Data Cache
- CPU.PCACHE: 32 kB of Program Cache
- CPU:DTAG: Data Cache TAG

- CPU.PTAG: Program Cache TAG
- CPU.DLMU: 64 KB of Local Memory Unit

CPU RAM can be affected by transient or permanent faults and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms names are assigned to each CPU memory instance. (e.g. CPU.DSPR, CPU.DLMU).

The RAM of the LS-CPU are NOT protected by the lockstep mechanism, which covers only the memory interfaces (*PMI* or *DMI*). Nevertheless, LS-CPU memories are classified as ASIL-D, allowing the user to execute ASIL-D SW from a LS-CPU in combination with local and remote LS-CPU RAM.
Non-LS CPU memories inherit the ASIL-B level from the non-LS CPU. In case the user wants to perform ASIL-D r/w operations by using a non-LS CPU memory from a LS-CPU, he shall take care of monitoring data corruption as highlighted by this ESM:

- ***ESM[SW]:CPU:DATA_INTEGRITY***

4.2.1.2 FCE

4.2.1.2.1 Nominal Functionality

The *FCE* is an hardware unit connected as slave to the *SPB* that provide an acceleration engine for *CRC* algorithms. The FCE offers eight different channels that can be individually configured and can be run in parallel by concurrent software tasks or operating system services, offloading the *CPU* in computing such algorithms. FCE can be considered a safety mechanism that increase error detection on data or registers configuration by means of information redundancy, as described here:

- ***SM[HW]:FCE:CRC***

4.2.1.2.2 Monitoring Concept

FCE is classified as safety mechanism, so fault affecting its functionality are classified as latent faults and are covered by *LBIST*. At runtime, incorrect *CRC* results delivered by the FCE are not detected by any safety mechanism or external safety measure.

FCE is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***"chapter.

4.2.1.3 STM

4.2.1.3.1 Nominal Functionality

The *STM* is a free-running 64-bit timer that is enabled immediately after application reset and can be read by *Application SW*. Each *CPU* has a dedicated STM. The STM can be configured to generate compare-match *ISR* by using dedicated registers.

4.2.1.3.2 Monitoring Concept

The *STM* can deliver incorrect counter results or generating wrong *ISR* because of failures on hardware or configuration registers. The STM is not part of the duplication area of the *CPU*, so no specific hardware is dedicated to monitor the correct behaviour of the timer. In case the STM is used in safety-relevant applications, the *Application SW* is in charge of make plausibility checks using an independent timer as described here:

- ***ESM[SW]:STM:MONITOR***

STM is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.1.4 HSM

4.2.1.4.1 Nominal Functionality

The HSM is a separate processor subsystem dedicated for security tasks. It is connected as master and slave to the *SPB* bus. Its use is not meant for safety-relevant applications. For security reasons this module is described in a separate documentation. Please contact your Infineon representative for further information.

4.2.2 MCU Function - Non Volatile Memory

4.2.2.1 NVM

4.2.2.1.1 Nominal Functionality

The *NVM* is dedicated on storing data or programs in flash memories and provide interface with other memories or with the *CPU*. It is composed by the following parts:

- Data Flash: Flash banks used to store application data. Two banks available (DF0 and DF1).
- Program Flash: Flash banks used to stored *Application SW* code. One bank with local access available per CPU.
- UCB: This block contains User Configuration Blocks [UCB], used for user configuration of the device. It is part of DF0.
- Flash Standard Interface: Executes erase, program and verify operations on all flash memories.
- Configuration Sector: This block contains device-specific settings not accessible by the user.
- BootRom: Contains the firmware executed by the device at start-up.
- DMU: Interface the [FSI] and *PFI* with data flash, [UCB] and [CFS].
- Program Flash Interface: provides fast connection between each program flash bank and the related CPU.

4.2.2.1.2 Monitoring Concept

PFLASH

The content of the *PFLASH* banks is crucial since it contains the code executed by the *CPU*. Any fault leading to transient or permanent modifications of the PFLASH can lead to severe failures not detectable by SW. Therefore the *NVM* offers dedicated safety mechanisms for the monitoring of several failures affecting the PFLASH.

Each 256-bit block is protected by an enhanced *EDC*/*ECC* monitor which is able to detect up to *TBE* and correct *SBE* and *DBE*:

- ***SM[HW]:NVM.PFLASH:ERROR_MANAGEMENT***
- ***SM[HW]:NVM.PFLASH:ERROR_CORRECTION***

In case the ECC signals an error in a PFLASH bank, the *Application SW* is in responsible to perform additional checks:

- ***ESM[SW]:NVM.PFLASH:WL_FAIL_DETECT***

NOTE:

In case of *SBAB* overflow system Integrator shall evaluate the most appropriate reaction (e.g. no alarm reaction).

In case of *DBAB* overflow alarm reaction depends on the ASIL requirements allocated to the function using the NVM:

- For QM functions using the NVM, system Integrator shall evaluate the most appropriate reaction (e.g. no alarm reaction)
- for ASIL functions using the NVM, system Integrator shall consider the NVM as non-operational and take appropriate reaction.

In both cases the system integrator has to take measures to prevent interference from the non-operational function.

Rational: In case of DBAB overflow the probability of undetectable multi-bit-error is in the same range as a further double bit error.

In case of [MBAB] overflow alarm reaction depends on the ASIL requirements allocated to the function using the NVM:

- For QM functions using the NVM, system Integrator shall evaluate the most appropriate reaction (e.g. no alarm reaction)
- for ASIL functions using the NVM, system Integrator shall consider the NVM as non-operational and take appropriate reaction.

In both cases the system integrator has to take measures to prevent interference from the non-operational function.

To address latent fault metric of the ECC logic, the PFLASH ECC/EDC decoder is monitored by two HW safety mechanisms:

- ***SM[HW]:NVM.PFLASH:ECC_ONLINE_CHECKER***
- ***SM[HW]:NVM.PFLASH:EDC_COMPARATOR***

The PFLASH read path is protected by a redundant logic by using a lockstep architecture:

- ***SM[HW]:NVM.PFLASH:CONTROL_REDUNDANCY***

To address latent faults of the read lockstep, stuck-at faults are detected by error injection:

- ***SM[HW]:NVM.PFLASH:CONTROL_REDUNDANCY_SCC***

Unintended PFLASH operations and configuration changes are continuously monitored (see Note):

- ***SM[HW]:NVM.PFLASH:INCORRECT_OPERATION_PROTECTION***
- ***SM[HW]:NVM.PFLASH:WAIT_STATE_PROTECTION***
- ***SM[HW]:NVM:REDINVFF***

Note: due to an HW bug, the above three safety mechanisms are affected by errata FLASH_TC.051.

Relevant configuration registers are protected against unwanted changes:

- ***SM[HW]:NVM.PFLASH:FLASHCON_MONITOR***

In combination to all these HW-based safety mechanisms, before start fetching safety-relevant code from a PLFASH bank, or upon every content update, the user shall execute these checks:

- ***ESM[SW]:NVM.PFLASH:INTEGRITY_CHECK***
- ***ESM[SW]:NVM.PFLASH:UPDATE_CHECK***

NVM

While *PFLASH* has several mechanisms protecting specifically this area of the memory, the following mechanisms are HW measures against faults which may affect different parts of the *NVM*. In particular, influences on NVM content given by faulty conditions on [FSI], are monitored by the following measure:

- ***SM[HW]:NVM:ARRAY_POINTER***

NVM settings installed during production test may be corrupted during runtime, therefore a safety mechanism is provided in order to avoid faults to become latent:

- ***SM[HW]:NVM:MISR***

NVM is a slave node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

NVM RAM

NVM has a dedicated RAM for FSI interface. The RAM can be affected by transient or permanent faults that can corrupt data and it has the safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to NVM.FSIRAM block.

Note: FSI RAM *SSH* registers are not accessible by User therefore:

- the ESM[SW]:NVM.FSIRAM:REG_MONITOR_TEST is not implementable and not listed in the FSI RAM safety mechanisms.

- In case one of the *SMU* alarms related to FSI RAM (ALM7[0], ALM71[1], ALM7[2]) is triggered, *Application SW* shall evaluate the RAM that generated the alarm, since it is OR-ed with other RAM blocks (LMU RAM0,1,2, DAM0,1). In case FSI RAM is the source of ALM7[1] - Uncorrectable error, the User shall react with a System Reset (or higher). For the other alarms (ALM7[0] and ALM7[2]), the System Integrator can define a reaction according to its application constraints.

4.2.3 MCU Function - Volatile Memory

4.2.3.1 EMEM

4.2.3.1.1 Nominal Functionality

The *EMEM* is a dedicated memory that contains RAM blocks (EMEM Tiles) which can be alternatively used for [ADAS] applications, calibration or trace data storage. The EMEM has interfaces to *SRI* and *BBB*.

4.2.3.1.2 Monitoring Concept

The *EMEM* can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms like all other SRAM blocks as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to EMEM.RAM block.

In addition to the SRAM safety mechanisms, the EMEM offers several HW measures for further fault detection capabilty.

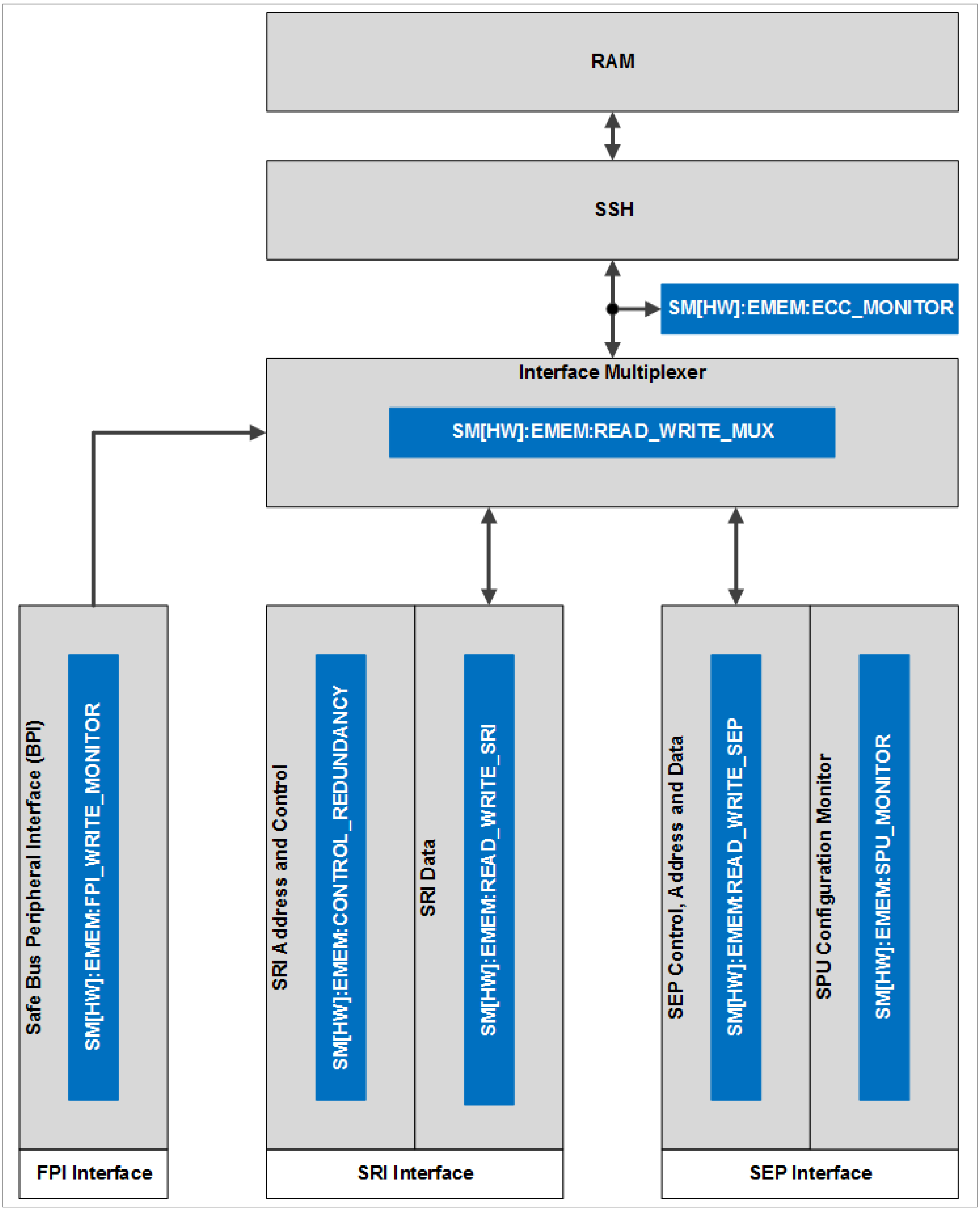

RAM
SSH
SM[HW]:EMEM:ECC_MONITOR
Interface Multiplexer
SM[HW]:EMEM:READ_WRITE_MUX
Safe Bus Peripheral Interface (BPI)
SM[HW]:EMEM:FPI_WRITE_MONITOR
SRI Address and Control
SM[HW]:EMEM:CONTROL_REDUNDANCY
SRI Data
SM[HW]:EMEM:READ_WRITE_SRI
SEP Control, Address and Data
SM[HW]:EMEM:READ_WRITE_SEP
SPU Configuration Monitor
SM[HW]:EMEM:SPU_MONITOR
FPI Interface
SRI Interface
SEP Interface

Figure 10

Depending on the interface, different HW mechanisms are implemented. For the *SRI* interface, which connects the EMEM to the *CPU* and *DMA*, data integrity is protected by the following:

- ***SM[HW]:EMEM:READ_WRITE_SRI***

Control logic between EMEM and SRI is protected by two specific mechanisms built to operate in a lockstep configuration:

- ***SM[HW]:EMEM:CONTROL_REDUNDANCY***
- ***SM[HW]:EMEM:CONTROL_REDUNDANCY_SCC***

The *FPI* interface with EMEM is protected against faults in the [BPI] which may lead to changes in EMEM configuration registers:

- ***SM[HW]:EMEM:FPI_WRITE_MONITOR***

The *SPU* modules have a dedicated data communication to EMEM, protected by the following HW measures:

- ***SM[HW]:EMEM:SPU_MONITOR***
- ***SM[HW]:EMEM:READ_WRITE_SEP***

The accesses to EMEM are monitored by an HW mechanism in the multiplexer:

- ***SM[HW]:EMEM:READ_WRITE_MUX***

The EMEM is protected against *SBE* by *ECC*. This mechanism is monitored by HW:

- ***SM[HW]:EMEM:ECC_MONITOR***

Since EMEM is used for storing data for ADAS applications, the *Application SW* is responsible for periodically check the correct configuration of the tiles during runtime as described in this external safety mechanism:

- ***ESM[SW]:EMEM:READ_CONFIGURATION***

In case EMEM is used for storing ASIL-D data, additional measures shall be taken:

- ***ESM[SW]:EMEM:DATA_INTEGRITY***

EMEM is a slave node of the SRI, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.3.2 LMU

4.2.3.2.1 Nominal Functionality

The *LMU* is an *SRI* peripheral providing access to volatile memory resources. Its primary purpose is to provide up to 256 kB of local memory for general purpose usage.

4.2.3.2.2 Monitoring Concept

The *LMU* can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms, like all other SRAM blocks as described in "***MCU Function - Volatile Memory***" paragraph in this chapter.

Additionally, the LMU interface with the *SRI* is protected by additional measures:

- ***SM[HW]:LMU:READ_WRITE_ECC***

- ***SM[HW]:LMU:CONTROL_REDUNDANCY_SCC***
- ***SM[HW]:LMU:CONTROL_REDUNDANCY***

The *ECC* output protecting the LMU is continuously monitored:
- ***SM[HW]:LMU:ECC_MONITOR***

LMU is a slave node of the SRI, therefore is protected by:
- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.3.3 SRAM

4.2.3.3.1 Nominal Functionality

Several blocks in the *MCU* have one or more dedicated SRAM. Memories belonging to *CPU* are mapped in the address space in segments 1 and 3-7 and can be accessed directly by masters, while peripherals [SRAM] are connected to the *SPB* and are mapped to the segment F. In this document, the various SRAM are identified with the name of the functional block where the memory is located. In case one functional block has multiple SRAM, each memory has a unique identifier.

The following table shows all SRAM instances available in the TC39x.

Functional block	memory name
CPU	DSPR
	PSPR
	DTAG
	PTAG
	PCACHE
	DCACHE
	DLMU
SPU	CONFIG
	BUFFER
	FFT
DMA	RAM
GTM	RAM
EMEM	RAM
LMU	RAM
PSI5	RAM
MCMCAN	RAM
CIF	RAM
HSPDM	RAM
NVM	FSIRAM

Functional block	memory name
TRACE	RAM
ERAY	RAM
AMU	LMU_DAM
GETH	RAM
SDMMC	RAM
SCR	RAM

4.2.3.3.2 Monitoring Concept

Data in SRAM may be corrupted by permanent hardware faults such as bit stuck-at and transient hardware faults such as soft errors caused by a neutron or alpha particle. Errors during a write, caused for example by a fault in the addressing logic, may also corrupt the data in SRAM, while errors during read can lead to a wrong or corrupted loaded data. Several safety mechanisms are allocated to each [SRAM] module and monitor the correct R/W operations.

The functional blocks and related memory names are declared in the table in the previous chapter. In this paragraph CPU.DSPR is taken as example.

Different fault categories are monitored and detected by SRAM safety mechanisms. Data integrity in the memory is protected by an *ECC/EDC* mechanism, which performs *SBE* correction and *DBE* detection:

- ***SM[HW]:CPU.DSPR:ERROR_CORRECTION***

NOTE: CPU.PTAG do not offer SBE error correction capabilities. SBE and DBE are detected and an uncorrectable error is generated as described in ***SM[HW]:CPU.PTAG:ERROR_DETECTION***.

In case a memory location is reporting an error, its address is stored into dedicated registers, allowing *Application SW* to react appropriately:

- ***SM[HW]:CPU.DSPR:ERROR_MANAGEMENT***

Address logic is also monitored by HW measures during R/W operations:

- ***SM[HW]:CPU.DSPR:ADDRESS***

The SRAM surrounding logic *SSH* is also monitored by HW mechanisms, which detects several faulty events which may lead to malfunctioning of the system:

- ***SM[HW]:CPU.DSPR:CONTROL***
- ***SM[HW]:CPU.DSPR:SM_CONTROL***

Some part of the SSH is protected by redundant registers, so transient faults affecting safety-relevant registers are detected:

- ***SM[HW]:CPU.DSPR:REG_MONITOR***

This redundant logic contains a *BIST* for latent fault detection:

- ***SM[HW]:CPU.DSPR:REG_MONITOR_TEST***

The test shall be executed by Application SW as specified here:

- ***ESM[SW]:CPU.DSPR:REG_MONITOR_TEST***

Considerations for SRAM Error Handling:

1: in case of SBE correction, the alarm can be configured for no reaction.
2: in case of DBE, address error detection, or ETRR overflow the alarm reaction depends on the ASIL requirements allocated to the function using the RAM:

- For QM functions using the RAM, System Integrator shall evaluate the most appropriate reaction (e.g. no alarm reaction)
- for ASIL functions using the RAM, after the first event, the System shall perform an Application Reset and re-execute the application. In case a second error event is detected within the same driving cycle, the function which uses the memory shall be considered non-operational. The system integrator has to take measures to prevent interference from the non-operational function.

3: in case of error event signaled by FAULTSTS.OPERR or FAULTSTS.MISCERR bit fields, the alarm reaction depends on the ASIL requirements allocated to the function using the RAM:

- for QM functions using the RAM, System Integrator shall evaluate the most appropriate reaction (e.g. no alarm reaction)
- for ASIL functions using the RAM, after the first event, the System shall perform a reaction as described in TC3xx User Manual v1.0, chapter 13.5.2.1.3. In case a second error event is detected within the same driving cycle, the function which uses the memory shall be considered non-operational. The system integrator has to take measures to prevent interference from the non-operational function.

4: The error analysis shall be performed only at module level i.e. using the SSH ECCD, ERRINFO and FAULTSTS registers provided, to find the type of error. Error localization to an address level within an SSH (i.e. trying to translate the ETRR address entries) shall be avoided.
EXAMPLES:

- The SSH_M_CAN10 has an UCE – then MCAN0 module shall not be used further, and switched off (System integrator shall ensure that there is no interference to the rest of the system), but MCAN1 (SSH_M_CAN20) or MCAN2 (SSH_M_CAN21) could be still used.
- SSH_GTM_MCS* - The MCS shall be not used and shall be switched off, but the DPLL can be used. The system integrator shall ensure that the MCS RAM is not used at all.

4.2.3.4 VMT

4.2.3.4.1 Nominal Functionality

The *VMT* contains the HW for *MBIST* functionality:

- ***SM[HW]:VMT:MBIST***

Details on MBIST can be found in "***Memory Built-in Self Test (MBIST)***" chapter.

4.2.3.4.2 Monitoring Concept

MBIST is classified as safety mechanism, so fault affecting its functionality are classified as latent faults and are covered by *LBIST*.

VMT is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.4 MCU Function - ADAS

4.2.4.1 RIF

4.2.4.1.1 Nominal Functionality

The *RIF* acts as an 32-bit interface between internal or external *ADC* channels with the *SPU* module.

4.2.4.1.2 Monitoring Concept

The *RIF* is used in ADAS applications, where a high level of safety is required. Therefore different parts of the RIF are monitored by HW safety mechanisms. *CRC* redundancy technique is used for increasing fault coverage on configuration registers and the data interfaces with the *MMIC* (input stage) and *SPU* (output stage).

- ***SM[HW]:RIF:SPU_INTERFACE***
- ***SM[HW]:RIF:CFG_MONITOR***
- ***SM[HW]:RIF:ERROR_HANDLING***

In case of an error detected in the MMIC interface via an *ISR*, the *Application SW* shall perform additional actions:

- ***ESM[SW]:RIF:ERROR_HANDLING***

Redundancy is applied to RIF datapath and safety mechanisms for increasing data integrity:

- ***SM[HW]:RIF:CONTROL_MONITOR***

Safety mechanisms are monitored by a dedicated HW measure:

- ***SM[HW]:RIF:SM_CHECK***

During rutime, the Application SW is responsible for monitoring the status of this safety mechanism:

- ***ESM[SW]:RIF:SM_CHECK***

RIF is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.4.2 SPU

4.2.4.2.1 Nominal Functionality

The Signal Processing Unit *SPU*is a semi-autonomous accelerator for performing Fast Fourier Transforms (FFTs) on data from one or more dedicated *ADC* interfaces. The SPU uses a three stage, streaming architecture to provide data pre-processing, FFT, and data post-processing operations. The SPU uses the Radar Memory to store datasets and has internal buffer memories which are used to store the data currently progressing through the processing pipeline.
The SPU is composed mainly by these parts:

- SPU CORE : computational unit for FFT calculations.
- SPU LOCKSTEP : full redundancy in case the second SPU is used as lockstep unit.
- SPU RAMs : used for store data (FFT, BUFFER) and configuration. (CONFIG).

4.2.4.2.2 Monitoring Concept

SPU Core

The *SPU* offers several safety mechanisms that monitors the correct behaviour of the unit. During runtime, the SPU configuration data and control flow of the operation is periodically checked by these two safety mechanisms:

- ***SM[HW]:SPU:ACTIVE_CFG_CHECK***
- ***SM[HW]:SPU:CONTROL_FLOW_SIGNATURE***

Additionally, the *Application SW* shall monitor the SPU control flow signature as described here:

- ***ESM[SW]:SPU:CONTROL_FLOW_SIGNATURE***

The SPU interfaces with *RIF* and *EMEM* are protected by the following safety mechanisms:

- ***SM[HW]:SPU:BYPASS_CRC***
- ***SM[HW]:SPU:EMEM_TILE***
- ***SM[HW]:SPU:EMEM_INTERFACE***

The second SPU instance can be configured for full redundancy (comparison of control and data outputs), partial redundancy (comparison of control only) or no redundancy (no comparison) as highlighted here:

- ***SM[HW]:SPU:REDUNDANCY***
- ***SM[HW]:SPU:PARTIAL_REDUNDANCY***

In case SPU is not configured for full redundancy, additional external measures shall be implemented at SW level:

- ***ESM[SW]:SPU:DATA_INTEGRITY***

A class of faults in the SPU could cause a deadlock in the SPU. A SW-based self test (SBST) is provided and will detect and signal an error in case the test execution time will take longer than expected:

- ***SM[SW]:SPU:SBST***

The integrity of some SPU safety mechanisms is monitored by a dedicated safety mechanism:

- ***SM[HW]:SPU:SM_CHECK***

Application SW is responsible to periodically check the status of the safety mechanism:

- ***ESM[SW]:SPU:SM_CHECK***

SPU is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

SPU RAM

SPU has three different types of RAM, each of which dedicated to specific usage. Each RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to SPU.CONFIG, SPU.BUFFER and SPU.FFT blocks.

4.2.4.3 HSPDM

4.2.4.3.1 Nominal Functionality

The [HSPDM] is an *SPB* peripheral used for generate one or two independent bit-streams for controlling external analog voltage using a low-pass filter. The 1-bit bit-streams are data read from the internal SRAM.

4.2.4.3.2 Monitoring Concept

The operation of the [HSPDM] is not monitored by HW safety mechanisms. Only the internal SRAM has dedicated as described in ***"MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to HSPDM.RAM block.

[HSPDM] is a slave node of the *SRI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.5 MCU Function - MCU Interconnect

4.2.5.1 SRI

4.2.5.1.1 Nominal Functionality

The SRI Fabric connects the *CPU*, the *DMA* module, and other high bandwidth requestors to high bandwidth memories and other resources for instruction fetches and data accesses. The *SRI* interconnect supports parallel transactions between SRI masters and independent SRI slaves.

4.2.5.1.2 Monitoring Concept

Any R/W operation can be affected by several faults during the address phase or the data phase, resulting in incorrect or missing data, wrong addressing and so on. *SRI* slaves are protected by HW mechanim checking the correct address space:

- ***SM[HW]:SRI:SRI_ADDRESS_MAP_MONITORING***

NOTE: this mechanism is instantiated for SRI, *NVM*, *EBU*, *EMEM*, *LMU*, *CPU*, *AMU.LMU_DAM*.

All SRI nodes (masters and slaves) are protected against integrity errors:

- ***SM[HW]:SRI:SRI_TRANSACTION_INTEGRITY***

NOTE: this mechanism is instantiated for SRI, NVM, EBU, EMEM, LMU, CPU, AMU.LMU_DAM, *GETH*, *DMA*, *HSSL*.
A detailed overview of SRI masters and slaves connections can be found in Aurix TC3xx User Manual v1.0.0, chapter 4.3.4.

SRI detects and triggers an *SMU* alarm and *ISR* in case of an SRI protocol error:

- ***SM[HW]:SRI:ERROR_HANDLING***

The *Application SW* shall execute the ISR and analyze the cause of the SRI error:

- ***ESM[SW]:SRI:ERROR_HANDLING***

Swap configuration is monitored during runtime:

- ***SM[HW]:SRI:SWAP_CONFIGURATION_PROTECTION***

4.2.5.2 FPI

4.2.5.2.1 Nominal Functionality

The *FPI* connects the high speed peripherals (*CPU* and *DMA*) to the medium and low bandwidth peripherals. The AURIX™ TC3xx family has up to two FPI Bus instances:

- System Peripheral Bus *SPB*: Main non-ADAS system and communication peripherals.
- Back Bone Bus *BBB*: Emulation Device related and ADAS related peripherals, available in ADAS / Emulation devices only.

4.2.5.2.2 Monitoring Concept

Any [R/W] operation can be affected by several faults during the address phase or the data phase, resulting in incorrect or missing data, wrong addressing and so on. The *FPI* has three dedicated safety mechanism that are responsible for monitoring the correct functioning of the bus:

- ***SM[HW]:FPI:ERROR_HANDLING***
- ***SM[HW]:FPI:TRANSACTION_INTEGRITY***
- ***SM[HW]:FPI:ADDRESS_MAP_MONITORING***

4.2.6 MCU function - MCU Communication

4.2.6.1 DMA

4.2.6.1.1 Nominal Functionality

The *DMA* moves data from source modules to destination modules without the intervention of the *CPU* or other on chip devices. A data move is defined by a DMA configuration data. A DMA channel operation is initiated by a DMA hardware request or a DMA software request.

4.2.6.1.2 Monitoring Concept

DMA Engine

During *DMA* operations, transactions can be subject to permanent or transient faults that can affect the success of the data moves in several ways.

Data can be corrupted during the move, so an *EDC* is implemented on the internal data path and a *CRC* is used to detect data corruption at the destination:

- ***SM[HW]:DMA:DATA_EDC***
- ***SM[HW]:DMA:DATA_CRC***

DMA source or destination address can be corrupted, resulting in wrong data at the destination. a CRC polynomial is used for detecting address modifications:

- ***SM[HW]:DMA:ADDRESS_CRC***

Data or address CRC shall be compared by *Application SW* at the completion of the transaction:

- ***ESM[SW]:DMA:ADDRESS_CRC***

Faults in the DMA move engine can lead to lost or delayed transactions that can be detected from the Application SW by using timestamp:

- ***SM[HW]:DMA:TIMESTAMP***

In case DMA timestamp is used, Application SW is responsible to check that the value is within the expected deadline:

- ***ESM[SW]:DMA:TIMESTAMP***

DMA requests are monitored by a dedicated safety mechanism:

- ***SM[HW]:DMA:REQUEST_MONITOR***

Lost transactions can be detected in case the DMA is configured for single transactions:

- ***SM[HW]:DMA:SINGLE_TRANSACTION***

DMA offers the possibility to configure via Application SW different resource partitions and provide access only to authorized masters. Two dedicated safety mechanisms monitor unauthorized accesses during runtime:

- ***SM[HW]:DMA:SV***
- ***SM[HW]:DMA:STI***

Additionally to *SMU* alarms, the DMA creates an *ISR* that provides diagnostic information on the error:

- ***SM[HW]:DMA:ERROR_HANDLING***

The Application SW is in charge of service the ISR and take the correct reaction:

- ***ESM[SW]:DMA:ERROR_HANDLING***

In case an error is reported by one of the above safety mechanims, the Application SW to elaborate the most appropriate reaction:

- ***ESM[SW]:DMA:SUPERVISION***

DMA is a master node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

DMA RAM

DMA has 16kB RAM in the segment 15 of the memory map. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to DMA.RAM block.

4.2.6.2 IR

4.2.6.2.1 Nominal Functionality

The *IR* is responsible for scheduling service requests (also called interrupts) to the correct service provider. In the TC3xx architecture, a service request can be raised by internal peripherals, external hardware or *Application SW*. The service providers are all *CPU* and the *DMA*.

4.2.6.2.2 Monitoring Concept

The *IR* is a critical block, since a fault in its logic can affect one or more service providers or *ISR* coming from HW or SW. The IR is connected to all internal functional blocks, so potentially a failure in a peripheral can generate malfunctions in the IR and propagate to (*CPU* or *DMA*). The correct behaviour of the IR and its monitoring functions during runtime are a crucial part of the safety measures implemented in the TC3xx architecture. This is achieved by a combination of internal and external safety mechanisms as explained in this paragraph.

IR configuration is write-protected from *SPB* accesses by dedicated access protection registers:

- ***SM[HW]:IR:FFI_CONTROL***

Service request node (SRN) information are protected by an *ECC* mechanism with *DBE* detection capability. ECC is checked whenever the SRN with an pending service request was accepted by the selected service provider:

- ***SM[HW]:IR:ISP_MONITOR***

An undetected fault to IR configuration can lead to severe malfunctioning of the *MCU*, therefore the *SFR* are monitored by three HW mechanisms:

- ***SM[HW]:IR:FPI_WRITE_MONITOR***
- ***SM[HW]:IR:REG_MONITOR***
- ***SM[HW]:IR:CFG_MONITOR***

In addition to these HW measures, *Application SW* is responsible to evaluate the ISR traffic during runtime:

- ***ESM[SW]:IR:ISR_MONITOR***

4.2.7 MCU Function - MCU Infrastructure

4.2.7.1 PMS

4.2.7.1.1 Nominal Functionality

The *PMS* controls the power generation of voltages rails used by all internal modules from an external 5V or 3.3V supply. It also monitors the voltage regulators during all states of the *MCU*.

4.2.7.1.2 Monitoring Concept

Latent faults in the *PMS* are covered by *PBIST* and *MONBIST* safety mechanisms and are explained in "***HW Self Tests for Latent Fault Metric Support***" chapter.

During runtime, permanent and transient faults affecting the voltage regulators operation may cause an incorrect voltage level of the internal core voltages. Therefore the external voltage (VEXT) and several internal supplies (VDD, VDDPD, VDDP3, VDDM, VDDPM, VEVRSB) are monitored and an *SMU* alarm is generated in case the voltage is outside the programmed range:

- ***SM[HW]:PMS:VEXT_MONITOR***
- ***SM[HW]:PMS:VDD_MONITOR***
- ***SM[HW]:PMS:VEVRSB_MONITOR***
- ***SM[HW]:PMS:VDDPD_MONITOR***
- ***SM[HW]:PMS:VDDP3_MONITOR***
- ***SM[HW]:PMS:VDDM_MONITOR***

NOTE: In addition to secondary voltage monitors on VDD, VEXT and VDDP3 (ALM9[3] and ALM9[5]), primary voltage monitor generates ALM9[16] and ALM9[17] based on thresholds configured in HSMUVMON and HSMOVMON registers. These alarms can be used as redundant indication for UV/OV on VDD, VEXT and VDDP3.

Transient faults on PMS logic are monitored during runtime:

- ***SM[HW]:PMS:REG_MONITOR***

As explained in "***External Voltage Supply***" chapter, additional measures have to be taken at system level to monitor the external supply voltage.

PMS is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.7.2 CLOCK

4.2.7.2.1 Nominal Functionality

The [CLOCK] functional block includes the clock generation unit, the clock scaling, the clock distribution and individual clock configuration for each *MCU* peripheral. The main function of the [CLOCK] is to generate from a single external source like an oscillator or a crystal internal clocks at different frequencies and distribute these clocks to all functional blocks in the MCU.

4.2.7.2.2 Monitoring Concept

Permanent and latent faults in the [CLOCK] can lead to wrong frequency of one or more clock signals. Since the clock signals are distributed to all peripherals, the [CLOCK] is a potential source of common-cause failures. Therefore failures detection of this block is an important part of the safety implemented in the TC3xx devices.

External oscillator frequency and internal back-up clock are used as input source for the internal *PLL*. Input frequencies are monitored by HW safety mechanism:

- ***SM[HW]:CLOCK:OSC_MONITOR***

A special HW design prevents tranfer of glitches to PLL output.

- ***SM[HW]:CLOCK:PLL_GLITCH_FILTER***

NOTE: This mechanism is an HW measure for failure avoidance. It does not perform any monitoring function nor report alarms via *SMU*.

In case a fault occurs to the external clock source, the PLL can lose its lock. In this case, the clock source will be automatically switched to internal source so the *MCU* will continue to operate:

- ***SM[HW]:CLOCK:PLL_LOSS_OF_LOCK_DETECTION***

The back-up clock and the output frequencies of the PLL are monitored and in case of an out-of-range event, an SMU alarm is generated:

- ***SM[HW]:CLOCK:ALIVE_MONITOR***

The *Application SW* is responsible for performing plausibility checks of the main system clocks against peripheral clocks:

- ***ESM[SW]:CLOCK:PLAUSIBILITY***

[CLOCK] is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.7.3 RESET

4.2.7.3.1 Nominal Functionality

The *RESET* is responsible for configuring and controlling reset events based on the reset trigger source that generates the request. More information about reset are available in "***MCU Reset State***" chapter.

4.2.7.3.2 Monitoring Concept

RESET configuration registers are protected against transient and permanent faults that may lead to severe mulfunctions of the *MCU* (e.g. unexpected reset during runtime). Therefore two HW measures are implemented at *FPI* interface and in the RESET logic:

- ***SM[HW]:RESET:FPI_WRITE_MONITOR***
- ***SM[HW]:RESET:FF_MONITORING***

RESET is a slave node of the *SRI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.7.4 SCU

4.2.7.4.1 Nominal Functionality

The *SCU* is a cluster of sub-modules which control various system functions, including:

- Reset Control: handles the various reset triggers from internal or external modules. This function is covered by the RESET functional block.
- External Request Handling (*ERU*): generate *ISR* or *SMU* alarm from an external source like an input pin.
- Emergency Stop (*ES*): in response of an emergency event, places the *PES* into a defined state without any SW interaction.
- System Registers: contains miscellaneous control registers associated with various system functions.
- Watchdog Timers (*WDT*): controls the global *MCU* safety watchdog and endinit functions.
- Trap Generation([TRAP]): determine which CPU shall receive a trap based on the trap event trigger.

4.2.7.4.2 Monitoring Concept

In case of a severe failure occurred in the *MCU*, a dedicated port can be configured to act as an emergency stop in order to flag an emergency event to the external HW:

- ***SM[HW]:SCU:EMERGENCY_STOP***

ES can be triggered by internal *SMU* alarms (if configured) or by a transition of the input port configured as ES input:

- ***SM[HW]:SCU.ERU:EXT_ALARM_GENERATION***

SCU contains LCLCON0/1 and LCL registers, which configure the *CPU* lockstep mode. This logic is protected by HW measure against transient and permanent faults:

- ***SM[HW]:SCU:LOCKSTEP_MONITOR***

The SCU *SFR* STCON is protected by start-up protection which is set by *SSW* at the end of the firmware procedure:

- ***SM[HW]:SCU:STARTUP_PROTECTION***

The SCU controls one safety watchdog timer, shared across all CPU, which is used for monitoring the activity of the "safety endinit" feature:

- ***SM[HW]:SCU:SAFETY_WATCHDOG***

In the same way, each CPU offers a watchdog timer,, which is used for monitoring the activity of the "endinit" feature:

- ***SM[HW]:SCU:ENDINIT_WATCHDOG***

NOTE: the "endinit" and "safety endinit" mechanisms are applied to several blocks and they are described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.7.5 SCR

4.2.7.5.1 Nominal Functionality

The *SCR* is an XC800 8-bit microcontroller that run during the standby mode of the *MCU*. Its main function is to control the wake-up signals and some I/O pads (marked with SCR) during the MCU standby period.

4.2.7.5.2 Monitoring Concept

SCR has 8kB XRAM for code and data available for User. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to SCR.RAM block.

NOTE: additional information on how to handle XRAM errors can be found in Aurix TC3xx User Manual v1.0.0, chapter 11.2.3.4.6.

4.2.7.6 DTS

4.2.7.6.1 Nominal Functionality

The *DTS* is composed by two sensors which measure the die temperature DTS and core die temperature *DTSC*.

4.2.7.6.2 Monitoring Concept

The *DTS* and *DTSC* sensors monitor die and core temperature during the *MCU* operation and generate *SMU* alarms if the measured temperature exceed the specified limits:

- ***SM[HW]:DTS:TEMPERATURE_MONITOR***

In case of a temperature alarm is generated by one of the sensors, the measures of both DTS and DTSC need to be compared by *Application SW*:

- ***ESM[SW]:DTS:DTS_RESULT***

DTS is a slave node of the *SRI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8 MCU function - MCU Interfaces

4.2.8.1 ERAY

4.2.8.1.1 Nominal Functionality

The *ERAY* module performs communication according to the FlexRay™ protocol specification v2.1,developed for automotive applications.

4.2.8.1.2 Monitoring Concept

ERAY Protocol

The *ERAY* protocol is not protected by any specific HW safety mechanisms. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:ERAY:SAFE_COMMUNICATION***

ERAY is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

ERAY RAM

ERAY has two 4kB XRAM for buffer storage. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to ERAY.RAM block.

4.2.8.2 GETH

4.2.8.2.1 Nominal Functionality

The *GETH* module includes an Ethernet media access controller (MAC) and physical layer (PHY). It allows connection of the *MCU* to a 10/100/1000 Mbps network.

4.2.8.2.2 Monitoring Concept

GETH Protocol

The *GETH* protocol is not protected by any specific HW safety mechanism. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:GETH:SAFE_COMMUNICATION***

GETH is a master node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

GETH RAM

GETH has 4kB RAM in the segment 15 of the memory map. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to GETH.RAM block.

4.2.8.3 HSSL

4.2.8.3.1 Nominal Functionality

The *HSSL* module implements transport layer tasks and hands over the data to another module which provides data link layer and physical layer services, data serialization and transmission.

4.2.8.3.2 Monitoring Concept

The *HSSL* protocol is not protected by any specific HW safety mechanism. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:HSSL:SAFE_COMMUNICATION***

HSSL is a master node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.4 MCMCAN

4.2.8.4.1 Nominal Functionality

The *MCMCAN* implements Bosch M_CAN and classical CAN and CAN-FD protocol controller.

4.2.8.4.2 Monitoring Concept

MCMCAN RAM

Each *MCMCAN* node has 16kB or 32kB message RAM in the segment 15 of the memory map. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to MCMCAN.RAM block.

MCMCAN protocol

The *MCMCAN* protocol is not protected by any specific HW safety mechanism. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:MCMCAN:SAFE_COMMUNICATION***

MCMCAN is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.5 QSPI

4.2.8.5.1 Nominal Functionality

The main purpose of the *QSPI* module is to provide synchronous serial communication with external devices using clock, data-in, data-out and slave select signals. The focus of the module is set to fast and flexible communication: either point-to-point or master-to-many slaves communication.

4.2.8.5.2 Monitoring Concept

The *QSPI* protocol is not protected by any specific HW safety mechanism. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:QSPI:SAFE_COMMUNICATION***

QSPI is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.6 PORT

4.2.8.6.1 Nominal Functionality

Each *PORT* module controls up to 16 General Purpose Input/Output (*GPIO*) port lines which are connected to pads connected to device pins/balls. Each GPIO is configurable by *Application SW* to act as capture input of analog signals or digital input/output.

4.2.8.6.2 Monitoring Concept

The *PORT* functionality is not monitored during runtime by any internal HW mechanism. Failures on PORT are protected by measures at *Application SW* level as described by these 2 ESM:

- ***ESM[SW]:PORT:REDUNDANCY***
- ***ESM[SW]:PORT:LOOPBACK***

PORT is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.7 PSI5

4.2.8.7.1 Nominal Functionality

The *PSI5* module implements PSI5 protocol specification V1.3. Four channels are available.

4.2.8.7.2 Monitoring Concept

PSI5 RAM

PSI5 module has 11x256B blocks of RAM in the segment 15 of the memory map. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to PSI5.RAM block.

PSI5 Protocol

The *PSI5* protocol is not protected by any specific HW safety mechanism. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:PSI5:CHANNEL_REDUNDANCY***

PSI5 is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter

4.2.8.8 PSI5S

4.2.8.8.1 Nominal Functionality

The PSI5-S IP-module performs communication according to the *PSI5* specification. It communicates with the external world via one ASC interface for all channels.

4.2.8.8.2 Monitoring Concept

PSI5-S is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.9 SENT

4.2.8.9.1 Nominal Functionality

The *SENT* module implements the digital pulse scheme for reporting sensor information via Single Edge Nibble Transmission (SENT) encoding, according to SENT SENT specification J2716 JAN2010.

4.2.8.9.2 Monitoring Concept

The *SENT* protocol is not protected by any specific HW safety mechanism. *Application SW* is responsible for implementing safe communication as described here:

- ***ESM[SW]:SENT:CHANNEL_REDUNDANCY***

SENT is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.10 I2C

4.2.8.10.1 Nominal Functionality

The *I2C* module provides a multi-master serial bus compliant to the I2C protocol.

4.2.8.10.2 Monitoring Concept

I2C is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.11 ASCLIN

4.2.8.11.1 Nominal Functionality

The main purpose of the *ASCLIN* module is to provide asynchronous serial communication using the LIN protocol. Additionally, the module supports the synchronous *SPI* communication.

4.2.8.11.2 Monitoring Concept

ASCLIN is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.12 MSC

4.2.8.12.1 Nominal Functionality

The *MSC* is a serial interface that is especially designed to connect external power devices. The serial data transmission capability minimizes the number of pins required to connect such external power devices. Parallel data information (coming from the timer units) or command information is sent out to the power device via a high- speed synchronous serial data stream (downstream channel).

4.2.8.12.2 Monitoring Concept

MSC is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.13 EBU

4.2.8.13.1 Nominal Functionality

The *EBU* controls the transactions for *SRI*-based systems between on-chip controller cores and external resources such as memories and peripherals. The EBU is internally connected to the SRI by a single slave interface.

4.2.8.13.2 Monitoring Concept

EBU is a slave node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.14 SDMMC

4.2.8.14.1 Nominal Functionality

The purpose of the *SDMMC* module is to enable communication to external managed NAND Flashes using the SD or eMMC interface. The focus of the module is set to the communication with a single embedded (eMMC) memory device or a single SD-card.

4.2.8.14.2 Monitoring Concept

SDMMC RAM

Each *SDMMC* has 1 kB buffer RAM in the segment 15 of the memory map. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to SDMMC.RAM block.

SDDMC Module

SDMMC is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.8.15 CIF

4.2.8.15.1 Nominal Functionality

The Camera and ADC Interface Module (*CIF*) provides 16-bit wide parallel read interface that can be used to connect camera sensors and external *ADC*.

4.2.8.15.2 Monitoring Concept

CIF RAM

The *CIF* RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms, like all other SRAM blocks as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to CIF.RAM block.

CIF Module

CIF is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.9 MCU Function - Analog Acquisition

4.2.9.1 CONVCTRL

4.2.9.1.1 Nominal Functionality

The *CONVCTRL* summarizes control functions which are common for all A/D converters implemented in the product. The following functions are provided:

• Phase Synchronizer**:** provides a clock enable signal to synchronize the clock signals of all analog blocks.

• Bias Configuration**:** enables/disables the bias distribution.

• BIST Configuration**:** configures the available test functions and returns status information.

The converter control block contains the registers which are required to configure the associated functions.

4.2.9.1.2 Monitoring Concept

The use of *CONVCTRL* for safety-related applications is described in "***Functional Use Cases***" chapter.

CONVCTRL is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.9.2 EDSADC

4.2.9.2.1 Nominal Functionality

The *EDSADC* module provides a series of analog input channels connected to on-chip modulators using the Delta/Sigma (DS) conversion principle. Digital input channels accept data streams from external modulators.

4.2.9.2.2 Monitoring Concept

The use of *EDSADC* for safety-related applications is described in "***Functional Use Cases***" chapter.

EDSADC is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.9.3 EVADC

4.2.9.3.1 Nominal Functionality

The *EVADC* provides a series of analog input channels connected to several clusters of Analog/Digital Converters using the Successive Approximation Register (SAR) principle to convert analog input values (voltages) to discrete digital values. An analog multiplexer selects one of several input channels and a dedicated control logic with several request sources defines the sequence of consecutive conversions.

4.2.9.3.2 Monitoring Concept

The use of *EVADC* for safety-related applications is described in "***Functional Use Cases***" chapter.

EVADC is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.10 MCU Function - Timers

4.2.10.1 GTM

4.2.10.1.1 Nominal Functionality

The *GTM* contains a module framework with sub-modules of different functionality. These sub-modules can be combined in a configurable manner to form a complex timer module that serves different application domains and different classes within one application domain.

4.2.10.1.2 Monitoring Concept

GTM RAM

GTM has 1 MB RAM in the segment 15 of the memory map. RAM can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms common to all SRAM blocks, as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to GTM.RAM block.

GTM Module

The use of *GTM* for safety-related applications is described in "***Functional Use Cases***" chapter.

GTM is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.10.2 CCU6

4.2.10.2.1 Nominal Functionality

The *CCU6* is a high-resolution 16-bit capture and compare unit with application-specific modes, mainly for AC drive control. Special operating modes support the control of Brushless DC-motors using Hall sensors or Back-EMF detection.

4.2.10.2.2 Monitoring Concept

The use of *CCU6* for safety-related applications is described in "***Functional Use Cases***" chapter.

CCU6 is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.10.3 GPT12

4.2.10.3.1 Nominal Functionality

The *GPT12* contains 2 general purpose 16-bit timers which are used for time measurements, event counting, pulse generation and other purposes.

4.2.10.3.2 Monitoring Concept

The use of *GPT12* for safety-related applications is described in "***Functional Use Cases***" chapter.

GPT12 is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.11 MCU Function - Signal Processing Powertrain

4.2.11.1 AMU.LMU_DAM

4.2.11.1.1 Nominal Functional

The *AMU.LMU_DAM* main purpose is to provide two high-performance ASC modelling units. Each AMU contains an IEEE floating point datapath and is able to compute the ASC algorithm. As a support, it also contains a high-speed DMA engine (ADMA). If the primary function of the AMU is not required, its local scratch pad RAM can be used as system SRAM via the DAM address space.

4.2.11.1.2 Monitoring Concept

The *AMU.LMU_DAM* can be affected by transient or permanent faults that can corrupt data and have the same safety mechanisms, like all other SRAM blocks as described in "***MCU Function - Volatile Memory***" paragraph in this chapter. The safety mechanisms are assigned to AMU.LMU_DAM block.

In case AMU.LMU_DAM is used for storing ASIL-D data, additional measures shall be taken:

- ***ESM[SW]:AMU.LMU_DAM:DATA_INTEGRITY***

AMU.LMU_DAM is a slave node of the *SRI*, therefore is protected by:

- SRI safety mechanisms described in "***Monitoring Concept***" chapter in the SRI.
- common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.12 MCU Function - Safety Mechanism

4.2.12.1 SMU

4.2.12.1.1 Nominal Functionality

The *SMU* centralizes all the alarm signals related to the different hardware and software-based safety mechanisms. Each alarm can be individually configured to trigger internal actions and/or notify externally the presence of faults via a *FSP*. the SMU is composed by 2 main parts:

- SMU Core: located in the core domain, it responds to all alarms generated by modules supplied with SPB clock.
- SMU stby: located in the stand-by domain, it responds to all alarms generated by modules supplied with BACK clock.

More details on SMU can be found in "***Failure management***" chapter.

4.2.12.1.2 Monitoring Concept

The *SMU* offers several HW safety mechanisms for managing failures and activate the required responses. Two timers are available for detecting delays of the *CPU* in case of *NMI* or *ISR*:

- ***SM[HW]:SMU:RT***

SMU core and stand-by domains act in a redundant manner and they can monitor the activity of the counterpart during runtime:

- ***SM[HW]:SMU:CCF_MONITOR***
- ***SM[HW]:SMU:ALIVE_MONITOR***

In case the *FSP* is activated either as an SMU reaction to a fault or by *Application SW*, a dedicated alarm is set:

- ***SM[HW]:SMU:FSP_MONITOR***

The system developer is in charge of implementing initial checks on SMU as also described in "***Boot and Startup Procedure***":

- ***ESM[SW]:SYS:MCU_FW_CHECK***
- ***ESM[SW]:SMU:ALIVE_ALARM_TEST***

Unintened data insertion in the SMU *SFR* are monitored during runtime:

- ***SM[HW]:SMU:FPI_WRITE_MONITOR***

SMU is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.13 Debug and Test functionalities

4.2.13.1 AGBT

4.2.13.1.1 Nominal Functionality

The *AGBT* implements an interface called Aurora Gigabit Trace for debug of the data coming from the *CIF*.

4.2.13.1.2 Monitoring Concept

AGBT is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.13.2 TRACE

4.2.13.2.1 Nominal Functionality

The *TRACE* module contains dedicated logic units for tracing purposes:

- Multi Core Debug Solution (*MCDS*): an on-chip trigger and trace solution for finding complex errors and for sampling selective data for advanced measurements.
- Mini MCDS: The MINIMCDS is an area optimized subset of the regular MCDS for the observation of just one trace source.
- Trace Memory (TRAM): It stores the trace data which is written by the Debug Memory Controller of the MINIMCDS. It shall not be used for code execution, data storage or overlay memory.

4.2.13.2.2 Monitoring Concept

TRACE functionality is disabled during operation. TRACE is protected by HW safety mechanism for providing *FFI*:

- ***SM[HW]:TRACE:FFI_CONTROL***

TRACE is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.13.3 DEBUG

4.2.13.3.1 Nominal Functionality

The *DEBUG* functional block supports debug, test and calibration over JTAG and DAP interface.

4.2.13.3.2 Monitoring Concept

DEBUG functionality is disabled during operation. This block is protected by HW safety mechanisms for providing *FFI*:

- ***SM[HW]:DEBUG:CFG_MONITOR***

DEBUG is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.2.13.4 DFX

4.2.13.4.1 Nominal Functionality

The *DFX* contains the HW for *LBIST* functionality. Details on LBIST can be found in "***Logic Built-In Self Test (LBIST)***" chapter.

4.2.13.4.2 Monitoring Concept

DFX functionality is disabled during operation. DFX is protected by HW safety mechanisms for providing *FFI*:

- ***SM[HW]:DFX:FFI_CONTROL***
- ***SM[HW]:DFX:FFI_MONITOR***

DFX is a slave node of the *FPI*, therefore is protected by common access protection safety mechanisms, described in "***Coexistence of HW/SW elements ASIL D***" chapter.

4.3 Safety Related Functions

4.3.1 Introduction

The following paragraphs show the main Safety Related Functions of the TC3xx family and their relationship with the TLSR (Top-Level Safety Requirement). The complete description of the AURIX TC3xx TLSR is presented in Safety Case Report.

As already mentioned in the preface, the AURIX TC3xx has been developed according to ISO 26262 *SEooC*, without targeting a specific item or application. Therefore during the design phase, most common use cases have been taken and from these safety requirements have been derived. The Safety Related Functions are the following:

Nr.	Safety Related Function	Relevant Functional Blocks
1	Safe Computation ASIL D	LS-CPU, NVM, LMU, AMU, EMEM, STM, FCE, VMT, SCU, DMA, SRI, FPI
2	Safe Computation ASIL B	NLS-CPU, NVM, LMU, EMEM, STM, FCE, DMA, SRI, FPI
3	ADAS Processing ASIL C	EMEM, SPU, RIF
4	Analog Acquisition ASIL D	EVADC, EDSADC, CONVCTRL
5	Analog Acquisition ASIL B	EVADC, EDSADC, CONVCTRL
6	Digital Acquisition ASIL D	GTM, CCU6
7	Digital Actuation ASIL B/D	GTM, CCU6
8	Sensor Acquisition ASIL B/D	PSI5, SENT
9	External Communication ASIL D	ERAY, MCMCAN, GETH, HSSL, QSPI
10	Safe State Support ASIL D	SMU
11	Avoidance or Detection of Common Cause Failures ASIL D	PMS, CLOCK, RESET, DTS, DFX, PORT, DEBUG, SCU
12	Coexistence of HW/SW elements ASIL D	common to many functional blocks in *MCU*

According to ISO 26262, ASIL levels are allocated to the functionality performed at vehicle level, so each block in the MCU inherits its ASIL from the safety requirements allocated to the functionality performed at vehicle level. Therefore the ASIL level shown in the table identifiy HW architectural metrics of the functional use case itself, not of the single functional block.

The ASIL level of Safety Related Function is reached with the following assumptions:

- Relevant safety mechanisms (SM[HW] or SM[SW]) are correctly configured.
- Alarms reactions are correctly configured depending on the application needs.
- Relevant external safety mechanisms (ESM[SW] or ESM[HW]) are correctly implemented.
- Safety Related Function 10, 11 an 12 are always correctly implemented.
- In case the system integrator does not implement one of the ESM[SW] listed in the related Safety Related Function, it is his responsibility to find an argumentation.
- In case one of the functional blocks is not used by the application, no safety requirements are allocated to the block itself. Therefore the related ESM[SW] are not relevant.

4.3.2 Application Dependent Functions

4.3.2.1 Safe Computation ASIL D

The Safety Related Function "Safe Computation ASIL D" describes the common use of the *MCU* for the execution of SW and internal data transfers in ASIL-D applications. It is related to the following TLSR:

- TLSR Software Execution Platform ASIL D
- TLSR Internal MCU Communication ASIL D

It is composed by the following Functional Use Cases:

FUC number	FUC description	Notes
0	LS-CPU accesses to its own SRAM and its local PFLASH of the NVM in the standard address map.	
1	LS-CPU accesses to its own SRAM and its local PFLASH of the NVM in the SOTA address map.	
2	LS-CPU accesses to remote LS-CPU's memory, the LMU and remote PFLASH of the NVM.	
3	LS-CPU accesses to remote NLS-CPU's memory, the AMU.RAM and the EMEM.	see Assumption 1
4	LS-CPU accesses to the safety related peripherals.	see Assumption 2
5	LS-CPU executes the startup software stored in the NVM.BROM to install MCU configuration.	
6	DMA or LS-CPU transaction from SRI source to SRI destination	
7	DMA or LS-CPU transaction from SRI source to FPI destination	
8	DMA or LS-CPU transaction from FPI source to SRI destination	

Assumption 1: While accessing these memories, data is needed to be loaded or stored with a checksum generated by TriCore instruction or by the *FCE* depending on the application use case.

Assumption 2: The safety-related peripherals considered in this *FUC* are FCE, *SCU*, *VMT*, *IR* and *STM*.

These FUC are common for many applications, since they describe normal uses of MCU in the context of automotive E/E systems.

Safe Computation ASIL D

For this Safety Related Function, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes

LS-CPU	ESM[SW]:CPU.*:*	1. only ESM[SW] for LS-CPU to be implemented.
NVM	ESM[SW]:NVM.*:*	
LMU	ESM[SW]:LMU.RAM:*	
AMU	ESM[SW]:AMU:*.*	
EMEM	ESM[SW]:EMEM:*.*	
STM	ESM[SW]:STM:*	
VMT	ESM[SW]:VMT:*	
FCE	ESM[SW]:FCE:*	
SCU	N.A.	
DMA	ESM[SW]:DMA.*:*	
SRI	ESM[SW]:SRI:*	
FPI	N.A.	

4.3.2.2 Safe Computation ASIL B

The Safety Related Function "Safe Computation ASIL B" describes the common use of the *MCU* for the execution of SW and internal data transfers in ASIL-B applications. It is related to the following TLSR:

- TLSR Software Execution Platform ASIL B
- TLSR Internal MCU Communication ASIL D

It is composed by the following Functional Use Cases:

FUC number	FUC description	Notes
0	NLS-CPU accesses to its own SRAM and its local PFLASH of the NVM in the standard address map.	
1	NLS-CPU accesses to its own SRAM and its local PFLASH of the NVM in the SOTA address map.	
2	NLS-CPU accesses to remote LS-CPU’s memory, the LMU, the EMEM and remote PFLASH of the NVM.	
3	NLS-CPU accesses to the AMU.RAM.	see Assumption 1
4	NLS-CPU accesses to the safety related peripherals.	see Assumption 2
5	DMA or NLS-CPU transaction from SRI source to SRI destination	
6	DMA or NLS-CPU transaction from SRI source to FPI destination	
7	DMA or NLS-CPU transaction from FPI source to SRI destination	

Assumption 1: While accessing to AMU.RAM, data is needed to be loaded or stored with a checksum generated by TriCore instruction or by the *FCE* depending on the application use case.

Assumption 2: The safety-related peripherals considered in this *FUC* are FCE, *SCU* and *STM*.

Safe Computation ASIL B

For this Safety Related Function, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
NLS-CPU	ESM[SW]:CPU.*:*	1. only ESM[SW] for NLS-CPU to be implemented.
NVM	ESM[SW]:NVM.*:*	
LMU	N.A.	
EMEM	ESM[SW]:EMEM:*.*	except ESM[SW]:EMEM:DATA_INTEGRITY
VMT	ESM[SW]:VMT:*	
FCE	ESM[SW]:FCE:*	
STM	ESM[SW]:STM:*	
DMA	ESM[SW]:DMA.*:*	
SRI	ESM[SW]:SRI:*	
FPI	N.A.	

NOTE: This Safety Related Function differs from the "Safe Computation ASIL D" only for the fact that the Application SW runs on a non-lockstep *CPU*. With the exception of the CPU domain, all ESM[SW] implemented for the "Safe Computation ASIL D" are valid also for example.

4.3.2.3 ADAS Processing ASIL C

The Safety Related Function "ADAS Processing ASIL C" describes the use of the ADAS cluster of the *MCU* for ASIL C applications. It is related to the following TLSR:

- TLSR ADAS Processing ASIL C

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	• One RIF receives up to 4 RX antenna data from external Radar Front End device. • Two SPU receive the same data from one RIF or two RIF and performs identical signal processing operations. • While data from one RIF or two RIF is processed by	This functional use case is so called 'Full SPU redundancy' mode.

	two SPU, each process result (control signal and address information) and result data can be compared with the process result of the other SPU. • Only 1 SPU (SPU0) sends the processed data to the EMEM. The accesses from the second SPU (SPU1) will be ignored. • Data can then be further processed by reading data back from the EMEM, reprocessing and writing the new results back to the EMEM. For these operations, the two SPU instances again operate on identical data and perform identical signal processing operations	
1	• Two RIF receives up to 8 RX antenna data from two external Radar Front End devices. • Each of the two SPU receives the data from one of the two RIF and performs identical signal processing operations. • While the different data from each RIF is processed by each of the two SPU, each process operation (control signal and address information) can be compared with the process operation of the other SPU. • Each of the two SPU sends different processed data to the EMEM. • Data can then be further processed by reading data back from the EMEM, reprocessing and writing the new results back to the EMEM. For these operations, the two SPU instances again operate on different data but perform identical signal processing operations. • SBST shall be executed within the defined FTTI by the system	This functional use case is so called 'Partial SPU redundancy' mode.

	integrator in order to detect the random hardware failures of the SPU.	
2	• 1 RIF receives up to 4 RX antenna data from external Radar Front End device. • One SPU receives the data from one or two RIF and performs signal processing. • One SPU sends its own processed data to the EMEM. • Data can then be further processed by the SPU by reading data back from the EMEM, reprocessing and writing the new results back to the EMEM. • SBST shall be executed within the defined FTTI by the system integrator in order to detect the random hardware failures of the SPU.	This use case is so called Non-redundancy mode.

SPU full redundancy mode

For this *FUC*, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EMEM	ESM[SW]:EMEM.*:*	except ESM[SW]:EMEM:DATA_INTEGRITY
RIF	ESM[SW]:RIF:*	
SPU	ESM[SW]:SPU:SM_CHECK ESM[SW]:SPU:EXECUTION_TIME_CHECK	SM[HW]:SPU:REDUNDANCY active

SPU partial redundancy mode

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EMEM	ESM[SW]:EMEM.*:*	except ESM[SW]:EMEM:DATA_INTEGRITY
RIF	ESM[SW]:RIF:*	
SPU	ESM[SW]:SPU:SBST	SM[HW]:SPU:PARTIAL_REDUNDANCY active

	ESM[SW]:SPU:SM_CHECK ESM[SW]:SPU:EXECUTION_TIME_C HECK ESM[SW]SPU:DATA_INTEGRITY	

SPU Non-redundancy mode

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EMEM	ESM[SW]:EMEM.*:*	except ESM[SW]:EMEM:DATA_INTEGRITY
RIF	ESM[SW]:RIF:*	
SPU	ESM[SW]:SPU:SBST ESM[SW]:SPU:SM_CHECK ESM[SW]:SPU:EXECUTION_TIME_C HECK ESM[SW]SPU:DATA_INTEGRITY ESM[SW]:SPU:CONTROL_FLOW_SIG NATURE	SM[HW]:SPU:CONTROL_FLOW_SIG NATURE active

4.3.2.4 Analog Acquisition ASIL D

The Safety Related Function "Analog Acquisition ASIL D" describes the use of *EVADC* and *EDSADC* for performing acquisition of analog signals in ASIL D applications. It relates to the following TLSR:

- TLSR Analog Acquisition ASIL D

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	Redundant safety-related analog signals are delivered by the system and redundantly processed by internal resources of the EVADC. The results of the redundant processing are transported from the EVADC to a system volatile memory via the *CPU* or the *DMA* and compared by the CPU.	see Assumptions 1,2
1	Redundant safety-related analog signals are delivered by the system and redundantly processed by internal resources of the EDSADC. The results of the redundant processing are transported from the EDSADC to a system volatile memory via the CPU or the DMA and compared by the CPU.	see Assumptions 1,2

2	Redundant safety-related analog signals are delivered by the system and redundantly processed by internal resources of the EVADC and the EDSADC. The results of the redundant processing are transported from the EVADC and the EDSADC to a system volatile memory via the CPU or the DMA and compared by the CPU.	see Assumptions 1,2

Assumption 1: The System Integrator shall avoid the use of adjacent pins and ADC channels belonging to the same group for the acquisition of redundant input signals. These are measures against common cause failures.

Assumption 2: The common cause failure outside the MCU due to faults in analog signals acquisition HW circuitry are detected by the system-level safety measures, e.g. design of diverse and dynamic analog inputs.

Analog acquisition ASIL-D with redundant EVADC channels

For this *FUC*, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EVADC	ESM[SW]:EVADC:PLAUSIBILITY ESM[SW]:EVADC:VAREF_PLAUSIBILITY ESM[SW]:EVADC:DIVERSE_REDUNDANCY ESM[SW]:EVADC:CONFIG_CHECK	
CONVCTRL	ESM[SW]:CONVCTRL:CONFIG_CHECK	

Analog acquisition ASIL-D with redundant EDSADC channels

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EDSADC	ESM[SW]:EDSADC:PLAUSIBILITY ESM[SW]:EDSADC:VAREF_PLAUSIBILITY ESM[SW]:EDSADC:DIVERSE_REDUNDANCY	.
CONVCTRL	ESM[SW]:CONVCTRL:CONFIG_CHECK	

Analog acquisition ASIL-D with one EVADC channel and one EDSADC channel

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EDSADC EVADC	ESM[SW]:EDSADC:PLAUSIBILITY ESM[SW]:EDSADC:VAREF_PLAUSIBILITY ESM[SW]:EDSADC:DIVERSE_REDUNDANCY ESM[SW]:EVADC:PLAUSIBILITY ESM[SW]:EVADC:VAREF_PLAUSIBILITY ESM[SW]:EVADC:DIVERSE_REDUNDANCY ESM[SW]:EVADC:CONFIG_CHECK	
CONVCTRL	ESM[SW]:CONVCTRL:CONFIG_CHECK	

4.3.2.5 Analog Acquisition ASIL B

The Safety Related Function "Analog Acquisition ASIL B" describes the use of *EVADC* and *EDSADC* for performing acquisition of analog signals in ASIL B applications. It relates to the following TLSR:

- TLSR Analog Acquisition - ASIL B

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	Single safety-related analog signal is received on single pin of the MCU and redundantly processed by internal resources of the EVADC. The results of the redundant processing are transported from the EVADC to a system volatile memory via the *CPU* or the *DMA* and compared by the CPU.	see Assumption 1
1	Single safety-related analog signal is delivered by the system and diversely processed by the EVADC and the EDSADC. The results of the diverse processing are transported from the EVADC and the EDSADC to a system volatile memory via the CPU or the DMA and compared by the CPU.	see Assumption 1

Assumption 1: The pattern of the single safety-related analog signal is known to the system integrator.

Single analog acquisition ASIL-B with EVADC channels

For this *FUC*, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes

EVADC	ESM[SW]:EVADC:PLAUSIBILITY ESM[SW]:EVADC:VAREF_PLAUSIBILITY ESM[SW]:EVADC:DIVERSE_REDUNDANCY ESM[SW]:EVADC:CONFIG_CHECK	
CONVCTRL	ESM[SW]:CONVCTRL:CONFIG_CHECK	

Singel analog acquisition ASIL-B with EDSADC channels

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
EDSADC EVADC	ESM[SW]:EDSADC:PLAUSIBILITY ESM[SW]:EDSADC:VAREF_PLAUSIBILITY ESM[SW]:EDSADC:DIVERSE_REDUNDANCY ESM[SW]:EVADC:PLAUSIBILITY ESM[SW]:EVADC:VAREF_PLAUSIBILITY ESM[SW]:EVADC:DIVERSE_REDUNDANCY ESM[SW]:EVADC:CONFIG_CHECK	
CONVCTRL	ESM[SW]:CONVCTRL:CONFIG_CHECK	

4.3.2.6 Digital Acquisition ASIL B/D

The Safety Related Function "Digital Acquisition ASIL B/D" describes the use of *GTM*, *CCU6* and *GPT12* for performing acquisition of digital signals in ASIL B and ASIL D applications. It is related to the following TLSR:

- TLSR Digital Acquisition ASIL B
- TLSR Digital Acquisition ASIL D

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	The *MCU* receives redundant signals and each signal is acquired by redundant GTM timer input resources (TIM). The signal measurement results shall be read and compared by the *CPU*.	See Assumptions 1, 3

1	The MCU receives diverse signals and each signal is acquired by redundant GTM timer input resources (TIM). The signal measurement results shall be read and compared by the CPU.	See Assumptions 1, 2, 3
2	The MCU receives redundant signals and each signal is acquired by the GTM timer input resource (TIM) and the CCU6, respectively. The signal measurement results shall be read and compared by the CPU.	See Assumptions 1, 3
3	The MCU receives redundant signals and each signal is acquired by the CCU6 and the GPT12, respectively. The signal measurement results shall be read and compared by the CPU.	See Assumption 4
4	The MCU receives a single signal which is acquired by both the CCU6 and the GPT12. The signal measurement results shall be read and compared by the CPU.	See Assumption 4

Assumption 1: The plausibility of the redundant signals is checked by the *Application SW* against known pattern (e.g. 0% or 100% PWM duty cycle shall be inferred as an invalid pattern).

Assumption 2: Diversity of digital input signals is introduced at System HW level (e.g. one digital input signal is inverted with respect to the other).

Assumption 3: The system integrator is responsible for avoiding the usage of adjacent pins for independence between redundant safety-related digital signals in order to avoid common cause failures of the ports and package.

Assumption 4: These use cases are foreseen for devices where the GTM is not available.

Digital Acquisition ASIL-D with redundant TIM/TIM channels

For this *FUC*, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
GTM	ESM[SW]:GTM:GTM_TIM_REDUNDANCY ESM[SW]:GTM:TIM_CLOCK_MONITORING	

Digital Acquisition ASIL-D with redundant TIM/TIM channels and HW diversity

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
GTM	ESM[SW]:GTM:GTM_TIM_REDUNDANCY ESM[SW]:GTM:TIM_CLOCK_MONITORING	

Digital Acquisition ASIL-D with redundant CCU6/TIM channels

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
GTM CCU6	ESM[SW]:GTM:GTM_CCU6_REDUNDANCY ESM[SW]:GTM:TIM_CLOCK_MONITORING	

Digital Acquisition ASIL-B with redundant CCU6/GPT12 channels

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
CCU6 GPT12	ESM[SW]:CCU6:CCU6_CAPTURE_MON_BY_GPT12	

4.3.2.7 Digital Actuation ASIL B/D

The Safety Related Function "Digital Actuation - ASIL B/D" describes the use of *GTM*, *CCU6* and *GPT12* for performing actuation of digital signals in ASIL B or ASIL D applications. It is related to the following TLSR:

- TLSR Digital Actuation ASIL D
- TLSR Digital Actuation ASIL B

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	The *CPU* or *DMA* generates an output digital signal request, e.g. PWM period and duty cycle. The GTM output resource (TOM or ATOM) is used to generate the PWM signal which shall be fed back from the external actuator to an MCU input port. Another GTM output resource (TOM or ATOM) is used to generate the internal reference PWM signal to be compared by the IOM against received PWM signal from the external actuator.	
1	The CPU or DMA generates an output digital signal request, e.g. PWM period and duty cycle. The GTM output resource (TOM or ATOM) is used to generate the PWM signal which shall be fed back from the external actuator to an *MCU* input port. The CCU6 output resource is used to generate the internal reference PWM signal to	

	be compared by the *IOM* against received PWM signal from the external actuator.	
2	The CPU or DMA generates an output digital signal request, e.g. PWM period and duty cycle. The GTM output resource (TOM or ATOM) is used to generate the PWM signal which shall be fed back from the external actuator to a GTM input resource (TIM). *Application SW* performs a comparison of the requested PWM output with the sampled feedback PWM signal.	
3	The CPU or DMA generates an output digital signal request, e.g. PWM period and duty cycle. The CCU6 output resource is used to generate the PWM signal which shall be fed back from the CCU6 to the GPT12. Application SW performs a comparison of the requested PWM output with the sampled feedback PWM signal.	This use case is foreseen for devices where the GTM is not available

Digital Actuation ASIL-D/B with redundant *TOM* channels and IOM comparison

For this *FUC*, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
GTM	ESM[SW]:GTM:IOM_ALARM_CHECK ESM[SW]:GTM:TIM_CLOCK_MONITORING	SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM active

Digital Actuation ASIL-D/B with redundant TOM/CCU6 channels and IOM comparison

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
GTM CCU6	ESM[SW]:GTM:IOM_ALARM_CHECK	*RHF* on GTM will be detected by CCU6 and IOM

Digital Actuation ASIL-D/B with redundant TOM/*TIM* channels and Application SW comparison

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
GTM	ESM[SW]:GTM:TOM_TIM_MONITORING ESM[SW]:GTM:TIM_CLOCK_MONITORING	

Digital Actuation ASIL-B with redundant CCU6/GPT12 channels and Application SW comparison

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
CCU6 GPT12	ESM[SW]:CCU6:CCU6_GPT12_MONITORING	

4.3.2.8 Sensor Acquisition ASIL B/D

The Safety Related Function "Sensor Acquisition" describes the use of *SENT* and *PSI5* for performing acquisition of signals from sensors in ASIL D or ASIL B applications. It is related to the following TLSR:

- TLSR Sensor Acquisition ASIL D
- TLSR Sensor Acquisition ASIL B

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	The *MCU* requests data from the redundant external PSI5 sensor devices. There are two modes of operation: • The MCU sends a request and the external PSI5 sensor device performs and responds with an answer. • The MCU sends a request and the external PSI5 sensor device performs and starts a periodic data transmission. The redundant PSI5 signals are independently processed by the PSI5 channels and read by the *CPU* or the *DMA*.	see Assumptions 1,2
1	The MCU requests data from the redundant external SENT sensor devices. There are two modes of operation: • The MCU sends a request and the external SENT sensor device performs and responds with an answer (so called SPC mode). • Once the external SENT sensor device is configured, it starts a periodic data transmission. The redundant SENT signals are independently processed by the SENT channels and read by the CPU or the DMA.	see Assumptions 1,2

Assumption 1: The *Application SW* is responsible for checking the integrity of the input sensor signals and the comparison of redundant sensor signal processing results against each other.

Assumption 2: Common cause failures shall be avoided by selecting non-adjacent pins for the sensors input signals and by selecting independent *GTM* triggers (if used).

Sensor acquisition using SENT

For this *FUC*, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
SENT	ESM[SW]:SENT:CHANNEL_REDUNDANCY	

Sensor acquisition using PSI5

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
PSI5	ESM[SW]:PSI5:CHANNEL_REDUNDANCY	

4.3.2.9 External Communication ASIL D

The Safety Related Function "External Communication ASIL D" describes the scenarios where the *MCU* communicates with external devices in [ASIL-D] applications. It is related to the following TLSR:

- TLSR Inbound External Communication ASIL D
- TLSR Outbound External Communication ASIL D

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	Data is received from external communication devices using one of the following protocols: *HSSL*, *GETH*, *ERAY*, *MCMCAN* or *QSPI*.	See Assumption 1
1	Data is transmitted to external communication devices using one of the following protocols: HSSL, GETH, ERAY, MCMCAN or QSPI.	See Assumption 1

Assumption 1: both *FUC* rely on ESM[SW] only, since no HW mechanisms are implemented for the monitoring of systematic and random hardware failures which may affect the communication.

External communication ASIL-D using GETH

For this protocol, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes

GETH	ESM[SW]:GETH:SAFE_COMMUNICATION	

External communication ASIL-D using ERAY

For this protocol, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
ERAY	ESM[SW]:ERAY:SAFE_COMMUNICATION	

External communication ASIL-D using MCMCAN

For this protocol, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
MCMCAN	ESM[SW]:MCMCAN:SAFE_COMMUNICATION	

External communication ASIL-D using HSSL

For this protocol, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
HSSL	ESM[SW]:HSSL:SAFE_COMMUNICATION	

External communication ASIL-D using QSPI

For this protocol, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
QSPI	ESM[SW]:QSPI:SAFE_COMMUNICATION	

4.3.3 MCU infrastructure functions

4.3.3.1 Avoidance or Detection of Common Cause Failures ASIL D

The Safety Related Function "Avoidance or Detection of Common Cause Failures ASIL D" groups the functional blocks that represents the major contributors to the common cause failure initiators within the MCU, thereby affecting the correct functionality of the safety related functions. It is related to the following TLSR:

- TLSR Avoidance or detection of common cause failures ASIL D

PMS

The *PMS* provides the power infrastructure, generates supply voltages using internal voltage regulators, monitors supply voltages, facilitates power distribution and manages system power modes. The failure modes of the PMS are plausible common cause failure initiators, since it could result in the failure of any safety-related functional block and its safety mechanism or the failure of redundant safety-related functional blocks.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:PMS:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
PMS	ESM[HW]:PMS:* ESM[SW]:PMS:*	
SYS	ESM[HW]:SYS:WATCHDOG_FUNCTION	

CLOCK

The [CLOCK] acts as the clock source for all MCU functions. The failure modes of the [CLOCK] are plausible common cause failure initiators, since it could result in the failure of any safety-related functional block and its safety mechanism or the failure of redundant safety-related functional blocks.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:CLOCK:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
CLOCK	ESM[SW]:CLOCK:*	
SYS	ESM[HW]:SYS:WATCHDOG_FUNCTION ESM[SW]:SYS:SW_SUPERVISION	

RESET

The *RESET* receives reset triggers from internal as well as external sources and provides various types of hierarchical resets to the *MCU*. The failure modes of the RESET are plausible common cause failure initiators, since it could result in the failure of any safety-related functional block and its safety mechanism or the failure of redundant safety-related functional blocks.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:RESET:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
SYS	ESM[HW]:SYS:WATCHDOG_FUNCTION ESM[SW]:SYS:SW_SUPERVISION	

DTS

The *DTS* measures the temperature of MCU. Two instances of the DTS are available across the MCU, namely DTS and *DTSC*. Since too high or too low temperature impacts the overall MCU functionality, thereby acting as a plausible common cause failure initiator, such abnormal temperatures must be detected without failure by the DTS.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:DTS:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
DTS	ESM[SW]:DTS:DTS_RESULT	

DFX

The *DFX* performs TEST function. The DFX is a plausible common cause failure initiator, since an unintentional activation of TEST mode would result in the un-deterministic state of the MCU, thereby leading to failure of safety applications being executed.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:DFX:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
SYS	ESM[HW]:SYS:WATCHDOG_FUNCTION	

PORT

In applications where the *PORT* is used for signal acquisition in a redundant manner to feed the signals to the redundant functional blocks or channels for safety-related functions or for signal actuation in a redundant manner, the failure modes of the PORT could be a plausible common cause failure initiator.

Common cause failures in this functional block are controlled by the following ESM[SW]:

Functional Block	ESM needed	Notes
PORT	ESM[SW]:PORT:REDUNDANCY ESM[SW]:PORT:LOOPBACK	see Assumption 1

Assumption 1: System integrator shall avoid the use of adjacent pins or the same ports where redundant hardware inputs/outputs are used.

DEBUG

The *DEBUG* provides functions for debugging, calibration and rapid prototyping of the MCU. The DEBUG is a plausible common cause failure initiator, since an unintentional activation of debug mode would result in the un-deterministic state of the MCU, thereby leading to failure of safety applications being executed.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:DEBUG:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
SYS	ESM[SW]:SYS:WATCHDOG_FUNCTION	

SCU

The *SCU* provides several system functions and configuration options for the MCU. The Failure modes of the SCU can a plausible common cause failure initiator, since several vital system functions could get affected thereby leading to failure of safety applications being executed.

Common cause failures in this functional block are controlled by a combination of safety mechanisms in HW (marked SM[HW]:SCU:*) and the following ESM[SW]:

Functional Block	ESM needed	Notes
SYS	ESM[SW]:SYS:WATCHDOG_FUNCTION	

4.3.3.2 Safe State Support ASIL D

The Safety Related Function "Safe State Support ASIL D" describes the scenario where the *MCU* shall react to an internal failure and report the failure notification to an external device in ASIL D applications. It is related to the following TLSR:

- TLSR Internal Failure Reporting ASIL D
- TLSR Internal Failure Reporting Time ASIL D
- TLSR External Failure Reporting ASIL D
- TLSR External Failure Reporting Time ASIL D
- TLSR External Failure Reporting Interface State Transition Control ASIL D
- TLSR External Failure Reporting Interface Transition Avoidance ASIL D
- TLSR External Failure Reporting Interface Control by External Signals ASIL D
- TLSR Alternate External Failure Reporting ASIL D

It is composed by the following Functional Use Cases:

FUC number	**FUC description**	**Notes**
0	The internal failure reporting interface enables the MCU to indicate, via the SMU_core, the presence of an internal MCU failure detected by a safety mechanisms. The internal failure reporting interface indicates two states through the SMU alarm flags: • The Fault State: indicates that a given HW/SW safety mechanism has detected a Failure in the MCU (SMU alarm flag set to 1). • The Fault-free State: indicates that no Failure has been detected by any HW/SW safety mechanism (SMU alarm flag set to 0). The SMU_core can be configured to request one of the following reactions to a Failure being detected by a HW/SW safety mechanism: • Interrupt Request to one or multiple CPUs (ISR) • Non Makeable Interrupt (NMI) • CPU Reset Request to one or multiple CPUs • MCU Reset Request	
1	The external failure reporting interface enables the communication of the presence of an internal MCU failure to an external Safe State Controller via SMU_core. It also enables the MCU to receive an error notification from an external device via the Emergency Stop ports and to react to it	

	without the intervention of a CPU. The external failure reporting interface indicates two states through the FSP ErrorPins or the Emergency Stop GPIOs: • The Fault State: indicates that a given HW/SW safety mechanism has detected a Failure in the MCU or that an external device has notified the MCU of a Failure. This State is indicated by setting FSP ErrorPins or the Emergency Stop GPIOs in Fault State • The Fault-free State: indicates that no Failure has been detected by any HW/SW safety mechanism. This State is indicated by setting FSP ErrorPins or the Emergency Stop GPIOs in Fault Free State. When a Failure is detected by a HW/SW safety mechanism, the SMU_core can be configured to requests the ES activation to the SCU or to set the FSP ErrorPins in Fault State	
2	The alternate external failure reporting interface enables the communication of the presence of an MCU common cause failure to an external Safe State Controller via SMU_stdby. The alternate external failure reporting interface indicates two states. • The Fault State: which indicates that a given HW safety mechanism has detected a Failure in the MCU. This State is indicated by setting FSP in high impedance state • The Fault-free State: which indicates that no Failure has been detected by any HW safety mechanism. This State is indicated by setting FSP ErrorPins in Fault Free State. Upon detection of common cause failure (e.g. clock failure, power failure, SMU failure, high or low temperature detection), the SMU_stdby can be configured to set the Fault Signaling Protocol ErrorPins in high impedance state regardless of the port configuration.	

Internal Failure Reporting

This *FUC* is achieved by SM[HW] only. The system integrator shall configure SMU alarm reactions according to its application needs.

External Failure Reporting

For this FUC, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes

SYS	ESM[HW]:SYS:FSP_ERROR_PIN_MONITOR ESM[HW]:SYS:ES_ERROR_PIN_MONITOR ESM[HW]:SYS:SW_ERROR_PIN_MONITOR	At least one of these ESM shall be implemented

Alternate Failure Reporting

For this FUC, the Application SW is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
SYS	ESM[HW]:SYS:FSP_ERROR_PIN_MONITOR ESM[HW]:SYS:ES_ERROR_PIN_MONITOR ESM[HW]:SYS:SW_ERROR_PIN_MONITOR	At least one of these ESM shall be implemented
SMU	ESM[SW]:SMU:ALIVE_ALARM_TEST	

4.3.3.3 Coexistence of HW/SW elements ASIL D

The Safety Related Function "Coexistence of HW/SW elements ASIL D" describes the safety mechanisms to support the coexistence of Hardware/Software elements which enable the system integrator to ensure the freedom from interference between HW/SW safety elements that have different ASILs or no ASILs assigned. It is related to:

- TLSR Coexistence of HW/SW elements ASIL D

Systematic faults and random hardware faults of the MCU hardware-elements and systematic failures of the MCU software-elements can cause interference, thereby violating the co-existence of HW and SW elements with different ASILs within the MCU. In order to fulfil the criteria of co-existence of HW and SW elements, it is the responsibility of the system integrator to use the below listed SMs according to the needs of the safety application.

Master TAG ID based safety mechanisms

The MCU provides the interconnect bus master agent with different master TAG ID. Master TAG ID is used to decide whether requested interconnect access to the MCU configuration address space area or memory address space area from functional block with the interconnect bus master agent function are granted or not.

- SM[HW]:*:CFG_AS_AP
- SM[HW]:*:VM_AS_AP
- SM[HW]:CPU:STI
- SM[HW]:DMA:STI

Access privilege level based safety mechanisms

The MCU provides the CPU and the DMA with different access privilege levels. The functional block configuration registers can be written by the CPU or the DMA only with the pre-determined privilege level.

- SM[HW]:*:SV_AP

- SM[HW]:CPU:SV
- SM[HW]:DMA:SV
- SM[HW]:CPU:UM0
- SM[HW]:CPU:UM1

ENDINIT and SAFETY_ENDINIT based safety mechanisms

The MCU provides the alternative protection mechanisms to access the functional block configuration registers via interconnect bus. These safety mechanisms are used to control the access permission to the configuration registers.

- SM[HW]:*:ENDINIT
- SM[HW]:*:SAFETY_ENDINIT
- SM[HW]:SCU:ENDINIT_WATCHDOG
- SM[HW]:SCU:SAFETY_WATCHDOG

Program and data memory region based safety mechanisms

The MCU provides the application software with the ability to protect the program memory and data memory from unauthorized CPU's access.

- SM[HW]:CPU:CODE_MPU
- SM[HW]:CPU:DATA_MPU

Safety mechanism to detect unintended interrupt request from the HW elements with lower ASIL or no ASIL

Unintended and continuous interrupt request to the IR from the HW elements with lower ASIL or no ASIL could consume too much CPU or DMA performance or it could consume shared resources (e.g. bus infrastructure – SRI, FPI) for too long time. In order to detect unintended interrupt requests, the *Application SW* is responsible to implement the following ESM[SW]:

Functional Block	ESM needed	Notes
IR	ESM[SW]:IR:ISR_MONITOR	

The following picture gives an overview of the HW mechanims implemented in the TC3xx architecture for ensuring Coexistence of HW/SW elements.

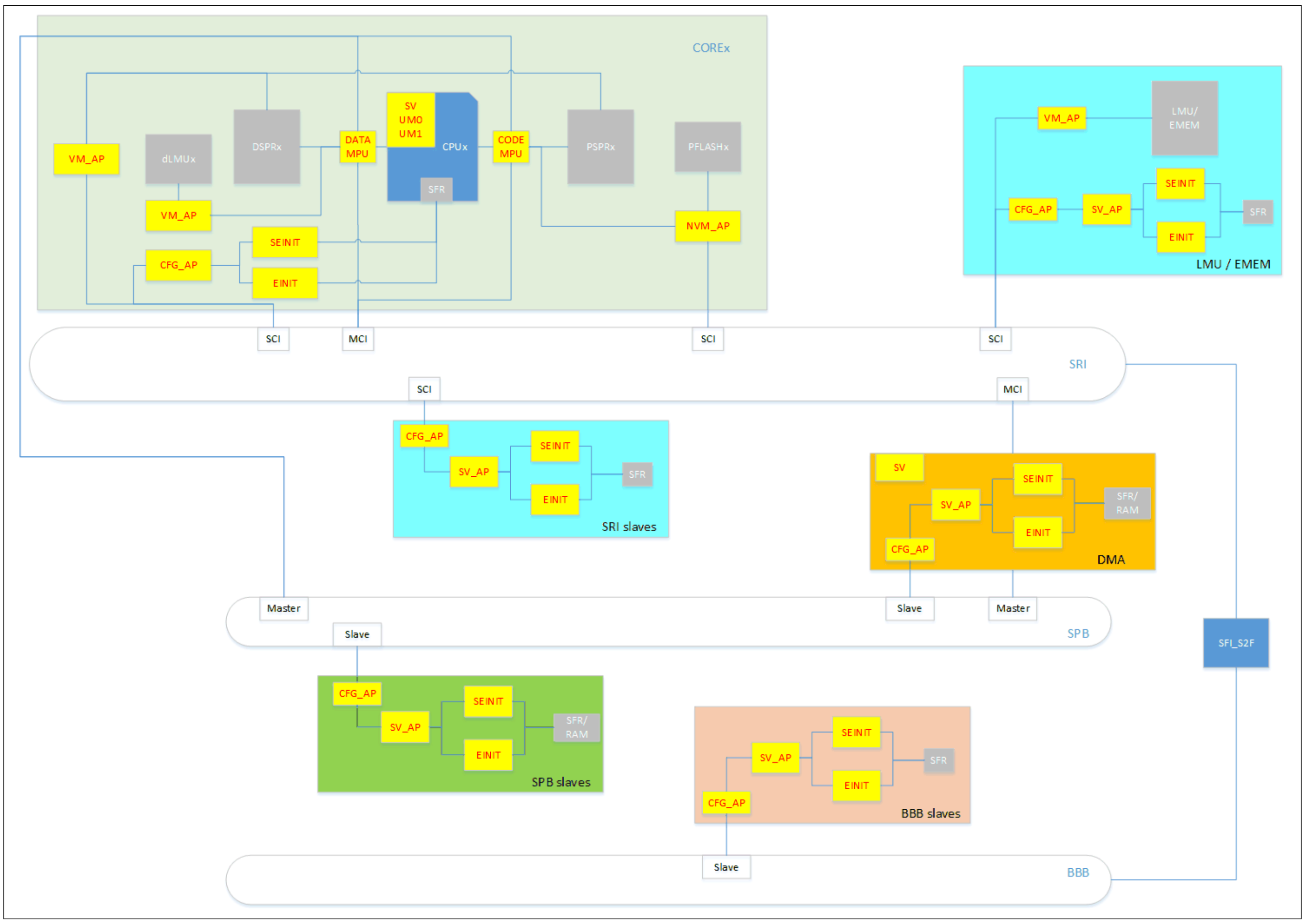
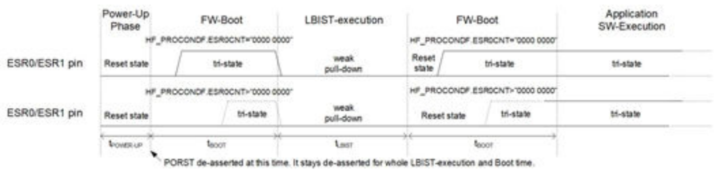

COREx
VM_AP
dLMUx
DSPRx
DATA MPU
SV UM0 UM1
CPUx
SFR
CODE MPU
PSPRx
PFLASHx
VM_AP
SEINIT
CFG_AP
EINIT
NVM_AP
LMU/ EMEM
VM_AP
CFG_AP
SV_AP
SEINIT
EINIT
SFR
LMU / EMEM
SCI
MCI
SCI
SCI
SRI
SCI
MCI
CFG_AP
SV_AP
SEINIT
EINIT
SFR
SRI slaves
SV
SV_AP
SEINIT
EINIT
SFR/ RAM
CFG_AP
DMA
Master
Slave
Master
SPB
Slave
SFI_S2F
CFG_AP
SV_AP
SEINIT
EINIT
SFR/ RAM
SPB slaves
SV_AP
SEINIT
EINIT
SFR
CFG_AP
BBB slaves
Slave
BBB

Figure 11

5 Assumptions of use

5.1 ESM[HW]:MCU:LBIST_MONITOR

Description	System integrator shall verify the completion of the LBIST operation by ensuring that the complete start-up sequence of the *MCU* is finished within the expected time interval defined by the user. This time interval shall include the t_{LBIST} value according to the LBIST configuration selected (in case of default LBIST configuration t_{LBIST} shall not exceed 6ms). The external monitoring is e.g. possible by observing the value of ESR0 or ESR1 pin. Both pins always show a permanent weak pull-down behavior during LBIST-execution is underway. In case ESR0/1 pins are still showing a weak pull-down behavior after $t_{POWER-UP}$ + t_{LBIST} + 2 x t_{BOOT} the chip can be reset through an external PORST.  Power-Up Phase FW-Boot LBIST-execution FW-Boot Application SW-Execution HF_PROCONDF.ESR0CNT="0000 0000" ESR0/ESR1 pin Reset state tri-state weak pull-down Reset state tri-state tri-state HF_PROCONDF.ESR0CNT>"0000 0000" ESR0/ESR1 pin Reset state tri-state weak pull-down Reset state tri-state tri-state $t_{POWER-UP}$ t_{BOOT} t_{LBIST} t_{BOOT} PORST de-asserted at this time. It stays de-asserted for whole LBIST-execution and Boot time. **Figure 12** Other startup monitoring mechanisms like external safety monitoring are implemented to ensure that startup time required by application is maintained.
Notes	During FW execution ESR0 pin will either show a tri-state or a weak pull-down behavior depending on the HF_PROCONDF .ESR0CNT setting. The FW will not influence the behavior of ESR1 pin. As soon LBIST operation has been started, a weak pull-down behavior will be visible for both ESR0 and ESR1. After LBIST operation is finished an internal PORST will be triggered followed by another FW-boot sequence (ESR0 pin will be tri-state or pull-down again depending on the HF_PROCONDF .ESR0CNT setting).
References	Aurix TC3xx User Manual v1.0.0, Chapter 9.3.3 Device Appendix
Recommendations	An external WDT can be used for this purpose as highlighted in ESM[HW]:SYS:WATCHDOG_FUNCTION. After PORST de-assertion, the external power supply waits for a first SPI message from the MCU which will serve the external WDT. If the wait time exceed the allowed time window because of a problem during start-up sequence, the appropriate reaction shall be taken at system level.

5.2 ESM[HW]:PMS:VEXT_VEVRSB_ABS_RATINGS

Description	It is assumed that the external voltage supplies will not stay between the maximum operating conditions and the absolute maximum ratings for longer than the time specified in the device data sheet. Exceeding the operating conditions for longer than the specified time may lead to an increase of the micro-controller failure rate. –This is the reliability issue. –In case micro-controller has a fault due to the exposure to the voltage level exceeding the specified operating conditions, self-test (LBIST, HW BIST) could detect this fault (within LBIST/HWBIST coverage) during start-up.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.2.5 Device Datasheet v1.0, "Absolute Maximum Ratings" chapter
Recommendations	

5.3 ESM[HW]:PMS:VEXT_VEVRSB_OVERVOLTAGE

Description	It is assumed that the system detects and disables the external voltage supplies VEXT and VEVRSB of the MCU in case of an over-voltage condition with the continuous monitoring of the external voltage supplies of the MCU. The Over-voltage can be divided into 2 regions: –Up to absolute maximum rating: Mechanisms are in a separate domain. Exceeding the operating conditions for longer than the specified time may lead to an increase of the micro-controller failure rate. –Above absolute maximum rating: this is the reliability issue. In case internal circuitry is damaged, LBIST and HWBIST will detect it during start-up. –If microcontroller is permanently damaged due to the exposure to the voltage level exceeding the specified maximum supply voltage, self-test (LBIST, HW BIST) will detect this fault during start-up.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.2.5 Device Datasheet v1.0, "Operating Conditions" chapter
Recommendations	

5.4 ESM[HW]:PMS:VX_FILTER

Description	It is assumed that external circuitry provides robustness against spikes for all externally supplied voltages. This refers to the external decoupling capacitors, external regulator output buffer capacitors, external EMI/EMC filter elements that shall attenuate short voltage spikes and high frequency oscillations. Voltage spikes shorter than the voltage monitoring period would not be detected and could lead to damages faults in the respective externally supplied domain. Multiple capacitors are recommended to be used in order to avoid of having single point fault.
Notes	

References	Device Datasheet v1.0, "EVRC SMPS External components" chapter
Recommendations	

5.5 ESM[HW]:SYS:ES_ERROR_PIN_MONITOR

Description	A component independent from the AURIX microcontroller shall monitor the Emergency Stop activation. Emergency Stop requests to PORTS provides a fast and software independent information about the presence of an internal failure of the microcontroller. As an alarm reaction, any combination of *FSP* (Fault Signaling Protocol) pin activation, ES(Emergency Stop) pin activation, NMI request and ISR request can be used depending on system environment. Moreover, a violation of a Safety Goal may happen due to hardware fault of the microcontroller's infrastructure composed by the [CLOCK], *PMS*, *SMU*. In case the SMU_stdby is not used to control ErrorPins (FSP[1:0]) to react on alarm from infrastructure monitors, the *Application SW* shall implement SMU status checks
Notes	In this case, upon appearance of an infrastructure faults, the system integrator is responsible to decide on the appropriate reaction.
References	Aurix TC3xx User Manual v1.0.0, Chapter 9.6
Recommendations	**1.** SMU_stdby hardware alarm path described in ***SM[HW]:PMS:VDDM_MONITOR*** can be replaced by software cyclic read of AG2i_STDBY or EVRSTAT.UVDDM and EVRSTAT.OVDDM registers AND triggering alarms in parallel by SMU_core commands and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** in case the software execution is not reliable due to infrastructure failures. **2.** SMU_stdby hardware alarm path described in ***SM[HW]:CLOCK:ALIVE_MONITOR*** can be replaced by Software Safety Mechanisms to detect wrong peripheral operation (in case of fPLL1, fPLL2 failure) and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** (in case of fPLL0 failure). **3.** In case SMU_core is used to control pin with Emergency Stop feature to react on alarms, check the correct activation by reading the status of Emergency Stop controlled pins.

5.6 ESM[HW]:SYS:FSP_ERROR_PIN_MONITOR

Description	A component independent from the AURIX microcontroller shall monitor the *FSP*. FSP error pin provides a fast and software independent information about the presence of an internal failure of the microcontroller. As an alarm reaction, any combination of FSP(Fault Signaling Protocol) pin activation, ES(Emergency Stop) pin activation, NMI request and ISR request can be used depending on system environment.

	Moreover, a violation of a Safety Goal may happen due to hardware fault of the microcontroller's infrastructure composed by the [CLOCK], *PMS*, *SMU*. In case the SMU_stdby is not used to control ErrorPins (FSP[1:0]) to react on alarm from infrastructure monitors, the *Application SW* shall implement SMU status checks.
Notes	In this case, upon apperence of an infrastructure faults, the system integrator is responsible to decide on the appropriate reaction.
	Aurix TC3xx User Manual v1.0.0, Chapter 15.3.1.8
Recommendations	SMU_stdby hardware alarm path described in ***SM[HW]:SMU:ALIVE_MONITOR*** can be replaced by: • Software cyclic reads of either - AG2i_STDBY registers (SMU Alive Alarm ALM21[16]) OR - AGx registers and plausibility check in software if the configured error reaction (e.g. ErrorPin) was executed as expected • AND triggering alarms in parallel by SMU_core commands and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** in case the software execution is not reliable due to infrastructure failure SMU_stdby hardware alarm path described in ***SM[HW]:PMS:VDDM_MONITOR*** can be replaced by software cyclic read of AG2i_STDBY or EVRSTAT.UVDDM and EVRSTAT.OVDDM registers AND triggering alarms in parallel by SMU_core commands and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** in case the software execution is not reliable due to infrastructure failure SMU_stdby hardware alarm path described in ***SM[HW]:CLOCK:ALIVE_MONITOR*** can be replaced by Software Safety Mechanisms to detect wrong peripheral operation (in case of fPLL1, fPLL2 failure) and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** (in case of fPLL0 failure) In case SMU_core is used to control FSPx pin to react on alarm, check correct activation by reading status of FSP ErroPins (FSP[1:0])

5.7 ESM[HW]:SYS:SW_ERROR_PIN_MONITOR

Description	A component independent from the AURIX microcontroller shall monitor the SW-based External Failure Reporting Interface. The SW-based External Failure Reporting Interface provides a fast information about the presence of an internal failure of the microcontroller. As an alarm reaction, any combination of *FSP* (Fault Signaling Protocol) pin activation, ES (Emergency Stop) pin activation, NMI request and ISR request can be used depending on system environment. Moreover, a violation of a Safety Goal may happen due to hardware fault of the microcontroller's infrastructure composed by the [CLOCK], *PMS*, *SMU*.

	In case the SMU_stdby is not used to control ErrorPins (FSP[1:0]) to react on alarm from infrastructure monitors, the *Application SW* shall implement SMU status checks.
Notes	In this case, upon appearance of an infrastructure faults, the system integrator is responsible to decide on the appropriate reaction.
References	Aurix TC3xx User Manual v1.0.0, Chapter 9.6
Recommendations	**1.** SMU_stdby hardware alarm path described in ***SM[HW]:PMS:VDDM_MONITOR*** can be replaced by software cyclic read of AG2i_STDBY or EVRSTAT.UVDDM and EVRSTAT.OVDDM registers AND triggering alarms in parallel by SMU_core commands and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** in case the software execution is not reliable due to infrastructure failures. **2.** SMU_stdby hardware alarm path described in ***SM[HW]:CLOCK:ALIVE_MONITOR*** can be replaced by Software Safety Mechanisms to detect wrong peripheral operation (in case of fPLL1, fPLL2 failure) and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** (in case of fPLL0 failure). **3.** In case SMU_core is part of the SW-based External Failure Reporting Interface, check correct activation by reading status of the External Failure Reporting Interface.

5.8 ESM[HW]:SYS:WATCHDOG_FUNCTION

Description	An external device with independent reference clock and with the functionality of a time-window watchdog supervises the AURIX MCU. This external device must initiate the transition to the system safe state if a fault which can lead to the violation of system safety goal is detected.
Notes	• This is necessary to control common cause failures initiated by external stress conditions or internal failures of the microcontroller, may lead to a state where the microcontroller cannot signal an internal failure. • Some systems implement external watchdogs supporting program flow monitoring with a question-answer protocol. Although these functions can be taken over by the internal watchdogs, cancelling the need for such a device, complex external watchdogs are still supported by the AURIX microcontroller. In this case a separate window watchdog may not be needed.
References	
Recommendations	

5.9 ESM[SW]:AMU.LMU_DAM:DATA_INTEGRITY

Description	In case the System Integrator wants to use LMU_DAM for ASIL-D applications, the *Application SW* shall monitor the corruption of data stored in the LMU_DAM caused by random hardware faults by using information redundancy.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 8.2.1

Recommendations	The Application SW shall store a block of data in the LMU_DAM with a data checksum accumulated over all the data in the block. The calculation of the checksum should be of a suitably robust implementation (e.g. IEEE 802.3 standard or AUTOSAR CRC32P4 standard). If the Application SW uses the data stored in the LMU_DAM, the Application SW shall first calculate a data checksum for all the data stored in the block. On completion of the calculation the Application SW shall compare the calculated data checksum with the expected data checksum for the stored data. If the Application SW matches the calculated data checksum and the expected data checksum, the Application SW may use the data. If the Application SW does not match the calculated data checksum and the expected data checksum, the Application SW shall signal an alarm to the SMU.

5.10 ESM[SW]:AMU.LMU_DAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, [Application SW] shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:AMU.LMU_DAM:REG_MONITOR_TEST***

5.11 ESM[SW]:CCU6:CCU6_CAPTURE_MON_BY_GPT12

Description	The CCU6 and the GPT12 redundantly process the input signal or signals generated by and received from the system. The application software shall compare the processing result of the CCU6 against the corresponding GPT12 processing result. Case #1: the CCU6 and the GPT12 receives a single signal, the equality of both processing result shall be checked by the application software if the PWM signal is within the expected range. Additionally, The application software shall determine the validity of processing result against known pattern to the system in order to monitor any possible common cause failure due to single signal usage.

	Case #2: the CCU6 and the GPT12 receives redundant signals, the equality of both processing result shall be checked by the application software if the PWM signal is within the expected range. In case the comparison result is outside the allowed tolerance, the application software shall notify the system about the presence of a fault and an appropriate action shall be taken by the system.
Notes	Case #1: The system integrator shall select the proper pin in a way that the pin is shared between the CCU6 and the GPT12. Case #2: The system integrator shall select proper pins in a way that there are sufficient distance between them.
References	
Recommendations	

5.12 ESM[SW]:CCU6:CCU6_GPT12_MONITORING

Description	The GPT12 processes the PWM input signal generated by the CCU6 and fed back to the GPT12. The application software shall compare the processing result of the GPT12 against intended PWM properties resulting from the CCU6 target configuration. In case the monitoring result is outside the allowed tolerance, the application software shall notify the system about the presence of a fault and an appropriate fault reaction shall be taken by the system.
Notes	
References	
Recommendations	

5.13 ESM[SW]:CIF.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, [Application SW] shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code

	implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CIF.RAM:REG_MONITOR_TEST***

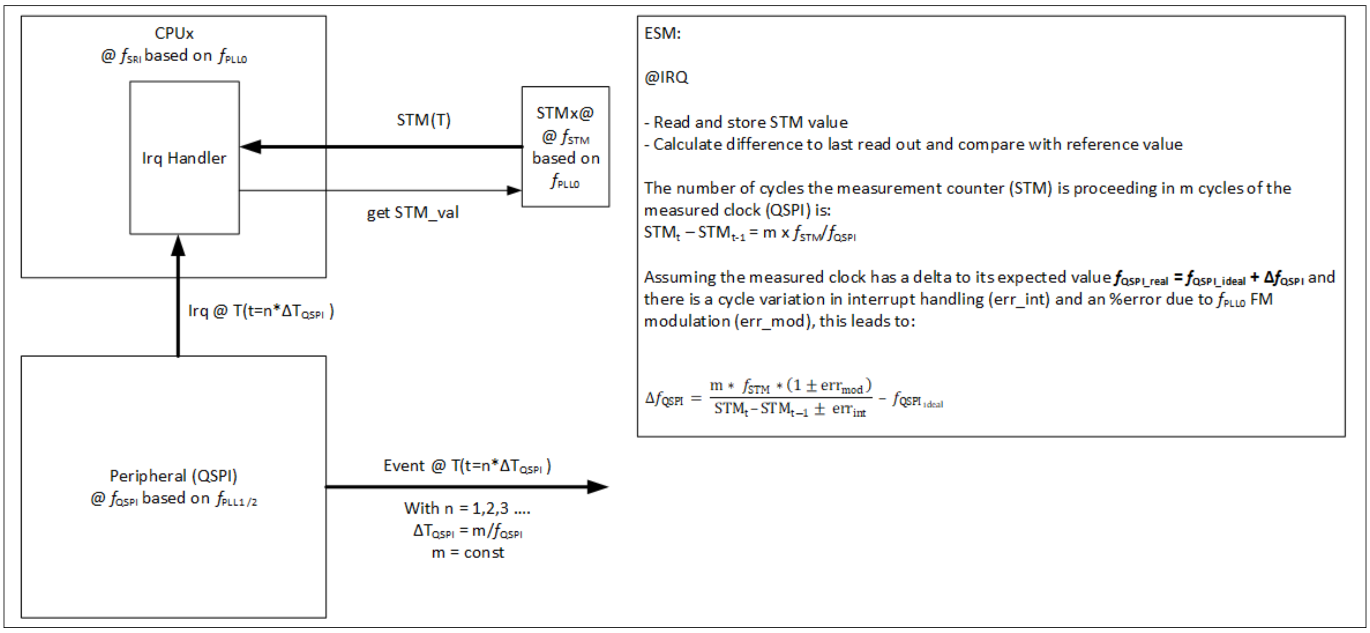
5.14 ESM[SW]:CLOCK:PLAUSIBILITY

Description	The *Application SW* shall evaluate the different clock frequencies provided by PERPLLCON1.K2 and PERPLLCON1.K3 with a time measurement based on SYSPLLCON1.K2 clock frequency. If the Application SW does not match the calculated and expected clock frequency values, the Application SW shall trigger an alarm.
Notes	
References	Aurix TC3xx User Manual v1.0.0, chapter 10.4.2.2
Recommendations	
Related safety mechanism	***SM[HW]:CLOCK:CFG_MONITOR***

he Application SW shall evaluate the different clock frequencies provided by PERPLLCON1.K2 and PERPLLCON1.K3 with a time measurement based on SYSPLLCON1.K2 clock frequency. If the Application SW does not match the calculated and expected clock frequency values, the Application SW shall trigger an alarm.

Example Implementation:

A CPUx, running at a clock based on f_{PLL0}, together with STMx, running at f_{SPB} based on f_{PLL0}, is continuously evaluating timestamps generated by a peripheral running at a clock based on f_{PLL1} or f_{PLL2}, e.g. f_{QSPI}. The related interrupt handler has to read and store the STM value to get the periodic timestamp. As the application knows the expected time interval between two timestamps, it is possible to calculate clock deviations to a certain degree. This depends on the length of the time intervals (the longer the better) and possible variances in interrupt processing. A timeout mechanism needs to be in place to cover a stopped peripheral clock as well.


CPUx
@ f_{SRI} based on f_{PLL0}
Irq Handler
STM(T)
get STM_val
STMx@ @ f_{STM} based on f_{PLL0}
Irq @ T(t=n*ΔT_{QSPI})
Peripheral (QSPI)
@ f_{QSPI} based on $f_{PLL1/2}$
Event @ T(t=n*ΔT_{QSPI})
With n = 1,2,3 ....
ΔT_{QSPI} = m/f_{QSPI}
m = const

ESM:

@IRQ

- Read and store STM value
- Calculate difference to last read out and compare with reference value

The number of cycles the measurement counter (STM) is proceeding in m cycles of the measured clock (QSPI) is:
$STM_t - STM_{t-1} = m \times f_{STM}/f_{QSPI}$

Assuming the measured clock has a delta to its expected value $\boldsymbol{f_{QSPI\_real} = f_{QSPI\_ideal} + \Delta f_{QSPI}}$ and there is a cycle variation in interrupt handling (err_int) and an %error due to f_{PLL0} FM modulation (err_mod), this leads to:

$$\Delta f_{QSPI} = \frac{m * f_{STM} * (1 \pm err_{mod})}{STM_t - STM_{t-1} \pm err_{int}} - f_{QSPI_{ideal}}$$

Figure 13

KDIV PLL0		**KDIV PLL1**		**f QSPInom**	**f QSPIalt**	**f STMnom**	**f STMalt**	**Δ STM_calculated**	**Δ STM_measured**	**Δ STM_diff**
nom	err	nom	err							
001	001	000	000	400,00E+6	400,00E+6	100,00E+6	102,00E+6	5,00E+3	5,43E+3	9%
001	001	000	001	400,00E+6	200,00E+6	100,00E+6	102,00E+6	5,00E+3	10,53E+3	111%
001	001	000	010	400,00E+6	133,33E+6	100,00E+6	102,00E+6	5,00E+3	15,63E+3	213%
001	001	000	100	400,00E+6	80,00E+6	100,00E+6	102,00E+6	5,00E+3	25,83E+3	417%
001	001	001	001	200,00E+6	200,00E+6	100,00E+6	102,00E+6	10,00E+3	10,53E+3	5%
001	001	001	000	200,00E+6	400,00E+6	100,00E+6	102,00E+6	10,00E+3	5,43E+3	-46%
001	001	001	011	200,00E+6	100,00E+6	100,00E+6	102,00E+6	10,00E+3	20,73E+3	107%
001	001	001	101	200,00E+6	66,67E+6	100,00E+6	102,00E+6	10,00E+3	30,93E+3	209%

001	001	010	010	133,33E+6	133,33E+6	100,00E+6	102,00E+6	15,00E+3	15,63E+3	4%
001	001	010	011	133,33E+6	100,00E+6	100,00E+6	102,00E+6	15,00E+3	20,73E+3	38%
001	001	010	000	133,33E+6	400,00E+6	100,00E+6	102,00E+6	15,00E+3	5,43E+3	-64%
001	001	010	110	133,33E+6	57,14E+6	100,00E+6	102,00E+6	15,00E+3	36,03E+3	140%
001	001	011	011	100,00E+6	100,00E+6	100,00E+6	102,00E+6	20,00E+3	20,73E+3	4%
001	001	011	010	100,00E+6	133,33E+6	100,00E+6	102,00E+6	20,00E+3	15,63E+3	-22%
001	001	011	001	100,00E+6	200,00E+6	100,00E+6	102,00E+6	20,00E+3	10,53E+3	-47%
001	001	011	111	100,00E+6	50,00E+6	100,00E+6	102,00E+6	20,00E+3	41,13E+3	106%
001	001	100	100	80,00E+6	80,00E+6	100,00E+6	102,00E+6	25,00E+3	25,83E+3	3%
001	001	100	101	80,00E+6	66,67E+6	100,00E+6	102,00E+6	25,00E+3	30,93E+3	24%
001	001	100	110	80,00E+6	57,14E+6	100,00E+6	102,00E+6	25,00E+3	36,03E+3	44%
001	001	100	000	80,00E+6	400,00E+6	100,00E+6	102,00E+6	25,00E+3	5,43E+3	-78%
001	001	101	101	66,67E+6	66,67E+6	100,00E+6	102,00E+6	30,00E+3	30,93E+3	3%
001	001	101	100	66,67E+6	80,00E+6	100,00E+6	102,00E+6	30,00E+3	25,83E+3	-14%
001	001	101	111	66,67E+6	50,00E+6	100,00E+6	102,00E+6	30,00E+3	41,13E+3	37%
001	001	101	001	66,67E+6	200,00E+6	100,00E+6	102,00E+6	30,00E+3	10,53E+3	-65%
001	001	110	110	57,14E+6	57,14E+6	100,00E+6	102,00E+6	35,00E+3	36,03E+3	3%
001	001	110	111	57,14E+6	50,00E+6	100,00E+6	102,00E+6	35,00E+3	41,13E+3	18%
001	001	110	100	57,14E+6	80,00E+6	100,00E+6	102,00E+6	35,00E+3	25,83E+3	-26%

001	001	110	010	57,14E+6	133,33E+6	100,00E+6	102,00E+6	35,00E+3	15,63E+3	-55%
001	001	111	111	50,00E+6	50,00E+6	100,00E+6	102,00E+6	40,00E+3	41,13E+3	3%
001	001	111	110	50,00E+6	57,14E+6	100,00E+6	102,00E+6	40,00E+3	36,03E+3	-10%
001	001	111	101	50,00E+6	66,67E+6	100,00E+6	102,00E+6	40,00E+3	30,93E+3	-23%
001	001	111	011	50,00E+6	100,00E+6	100,00E+6	102,00E+6	40,00E+3	20,73E+3	-48%
$f_{QSPInom}$	**f_{DCO1}/(KDIV PLL1 nom +1)**									
$f_{QSPIalt}$	**f_{DCO1}/(KDIV PLL1 err +1)**									
f_{STMnom}	**f_{DCO0}/(KDIV PLL0 nom +1)/3**									
f_{STMalt}	**(f_{DCO0}+f_{DCO0}*mod)/(KDIV PLL0 err +1)/3**									
$\Delta_{STM\_calculated}$	**(m/ $f_{QSPInom}$)*f_{STMnom}**									
$\Delta_{STM\_measured}$	**(m/$f_{QSPIalt}$)*f_{STMalt}+Δ_{IRQ}/3**									
Δ_{STM_diff}	**($\Delta_{STM\_measured}$-$\Delta_{STM\_calculated}$)/$\Delta_{STM\_calculated}$**									
m	**20000**	**f_{DCO0} = 600 MHz**								
Δ_{IRQ} =	**1000**	**f_{DCO1} = 400 MHz**								

Assuming a variance of 1000 CPU cycles for interrupt handling, a ratio of f_{SRI}/f_{STM} = 3 and max 2% error due to f_{PLL0} FM modulation, the table above shows an example where KDIV errors can be detected with the ESM. The nominal measurement error (without KDIV flop) must be less than the deviation when there is a KDIV alteration. This is needed to differentiate measurement error and "real" error. The example above shows alterations in PLL1 KDIV, the opposite case with alterations in PLL0 KDIV is similar. The used values for m and Δ_{IRQ} have been arbitrarily chosen as an example.

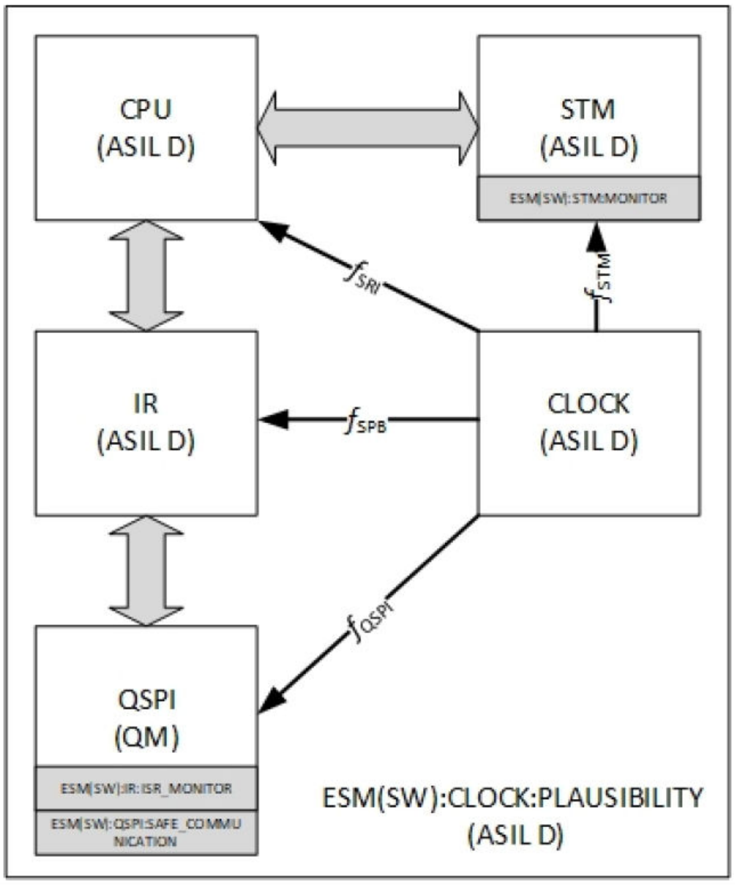


Figure 14

- Safety Mechanism ESM[SW]:STM:MONITOR is used to detect faults in STM which can lead to wrong STM time stamps read from CPU.
- Safety Mechanism ESM[SW]:IR:ISR_MONIOTOR is used to detect faults in interrupt processing which can lead to missing or unintended interrupt events from QSPI.
- Safety Mechanism ESM[SW]:QSPI:SAFE_COMMUNICATION is used to detect faults in QSPI protocol machine which can leas to missing or unintended interrupt events from QSPI.
- Safety Mechanism ESM[SW]:SYS:SW_SUPERVISION is used to detect faults in CPU and memories which can lead to wrong calculation during ESM[SW]:CLOCK:PLAUSIBILITY execution.

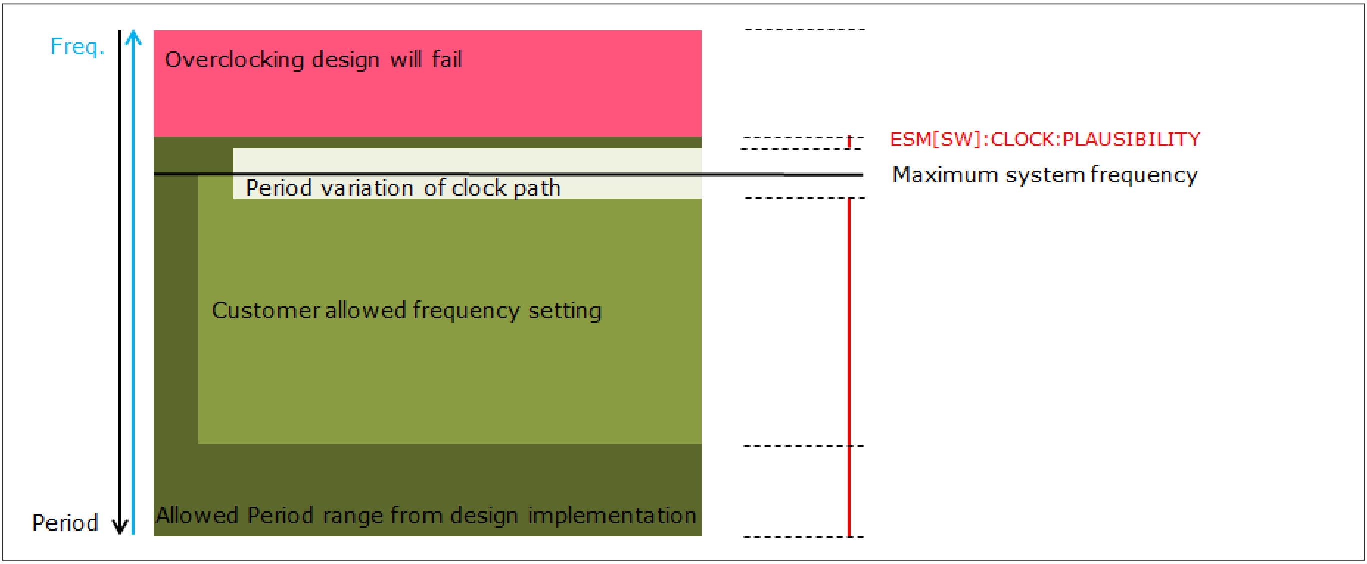

Freq.
Overclocking design will fail
Period variation of clock path
Customer allowed frequency setting
Allowed Period range from design implementation
Period
ESM[SW]:CLOCK:PLAUSIBILITY
Maximum system frequency

Figure 15

5.15 ESM[SW]:CONVCTRL:ALARM_CHECK

Description	The *Application SW* shall activate the fault injection mechanisms at least once during a drive cycle. It shall then check the generation of the associated alarm signal.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 29.2.4
Recommendations	

5.16 ESM[SW]:CONVCTRL:CONFIG_CHECK

Description	The *Application SW* shall verify the write access to safety relevant register PHSCFG by reading the respective register and check the returned value.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 29.2.5
Recommendations	

5.17 ESM[SW]:CPU.DCACHE:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.

Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.DCACHE:REG_MONITOR_TEST***

5.18 ESM[SW]:CPU.DLMU:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.DLMU:REG_MONITOR_TEST***

5.19 ESM[SW]:CPU.DSPR:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, [Application SW] shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time.

	This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.DSPR:REG_MONITOR_TEST***

5.20 ESM[SW]:CPU.DTAG:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.DTAG:REG_MONITOR_TEST***

5.21 ESM[SW]:CPU.PCACHE:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.PCACHE:REG_MONITOR_TEST***

5.22 ESM[SW]:CPU.PSPR:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.PSPR:REG_MONITOR_TEST***

5.23 ESM[SW]:CPU.PTAG:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CPU.PTAG:REG_MONITOR_TEST***

5.24 ESM[SW]:CPU:AP_CHECK

Description

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	N.A.	N.A.
Latent Fault Detection	N.A.	Yes

	<table><tr><th>Detection and Control</th><th>lockstep CPU</th><th>non-lockstep CPU</th></tr><tr><td>Freedom from Interference</td><td>N.A.</td><td>Yes</td></tr></table> The *Application SW* shall perform a verification of correct *CPU* Access Protection configuration after initial SW initialization and after every subsequent configuration change. If the Application SW does not match the configured value to the expected value the Application SW shall trigger an alarm to the *SMU*.
Notes	The following registers should be checked CPUx_SFR_SPROT_ACCEN*, CPUx_LPB_SPROT_ACCEN*
References	Aurix TC3xx User Manual v1.0.0, Chapter 5.3.10 and 5.4.5
Recommendations	

5.25 ESM[SW]:CPU:BUS_MPU_INITCHECK

Description	<table><tr><th>Detection and Control</th><th>lockstep CPU</th><th>non-lockstep CPU</th></tr><tr><td>Single Point Fault Detection</td><td>N.A.</td><td>N.A</td></tr><tr><td>Latent Fault Detection</td><td>N.A.</td><td>Yes</td></tr><tr><td>Freedom from Interference</td><td>N.A.</td><td>Yes</td></tr></table> The *Application SW* shall perform a verification of correct *CPU* Bus MPU configuration after initial SW initialization and after every subsequent configuration change. If the Application SW does not match the configured value to the expected value the Application SW shall trigger an alarm to the *SMU*.
Notes	The following registers should be checked CPUx_SPR_SPROT_RGN*, CPUx_DLMU_SPROT_RGN*, LMUx_RGN* (for devices with Local Memory Units), DAMx_RGN* (for devices with Default Application Memory), EMEM_MPUx_RGN* (for devices with Extension Memory)
References	Aurix TC3xx User Manual v1.0.0, chapter 5.4.7
Recommendations	
Related safety mechanism	***SM[HW]:AMU.LMU_DAM:VM_AS_AP*** ***SM[HW]:CPU.DLMU:VM_AS_AP*** ***SM[HW]:CPU.DSPR:VM_AS_AP*** ***SM[HW]:CPU.PSPR:VM_AS_AP*** ***SM[HW]:EMEM:VM_AS_AP*** ***SM[HW]:LMU:VM_AS_AP***

5.26 ESM[SW]:CPU:CODE_MPU_CHECK

Description	**Detection and Control** / **lockstep CPU** / **non-lockstep CPU**: Single Point Fault Detection: N.A. / N.A; Latent Fault Detection: N.A. / Yes; Freedom from Interference: N.A. / Yes. The *Application SW* shall perform a verification of correct *CPU* Code MPU configuration after initial SW initialization and after every subsequent configuration change. If the Application SW does not match the configured value to the expected value the Application SW shall trigger an alarm to the *SMU*.
Notes	The following registers should be checked CPUx_CPR*, CPUx_CPXE*
References	Aurix TC3xx User Manual v1.0.0, Chapter 5.3.10
Recommendations	
Related safety mechanism	***SM[HW]:CPU:CODE_MPU***

Description table:

Detection and Control	**lockstep CPU**	**non-lockstep CPU**
Single Point Fault Detection	N.A.	N.A
Latent Fault Detection	N.A.	Yes
Freedom from Interference	N.A.	Yes

5.27 ESM[SW]:CPU:DATA_INTEGRITY

Detection and Control	**lockstep CPU**	**non-lockstep CPU**
Single Point Fault Detection	N.A.	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description	In case the System Integrator wants to use a non-lockstepped *CPU* data memory for ASIL-D applications, the *Application SW* shall monitor the corruption of data stored in the CPU.PSPR, CPU.DSPR or CPU.DLMU of non-lockstepped CPUs caused by random hardware faults by using information redundancy.
Notes	This ESM[SW] describes the measures to introduce for monitoring faults on a memory of a non-lockstep CPU in case of ASIL-D applications. For all ASIL-D applications, Application SW shall be executed from a Lockstep CPU only. This requirement includes all checks listed in this ESM[SW].
References	
Recommendations	The Application SW shall store a block of data in the CPU.PSPR, CPU.DSPR or CPU.DLMU of a non-lockstepped CPU with a data checksum accumulated over all the data in the block. The calculation of the checksum should be of a suitably robust implementation (e.g. IEEE 802.3 standard or AUTOSAR CRC32P4 standard). If the Application SW uses the data stored in the CPU.PSPR, CPU.DSPR or CPU.DLMU of a non-lockstepped CPU, the Application SW shall first calculate a data checksum for all the data stored in the block. On completion of the

	calculation the Application SW shall compare the calculated data checksum with the expected data checksum for the stored data. If the Application SW matches the calculated data checksum and the expected data checksum, the Application SW may use the data. If the Application SW does not match the calculated data checksum and the expected data checksum, the Application SW shall signal an alarm to the *SMU*.

5.28 ESM[SW]:CPU:DATA_MPU_CHECK

Description

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	N.A.	N.A
Latent Fault Detection	N.A.	Yes
Freedom from Interference	N.A.	Yes

The *Application SW* shall perform a verification of correct *CPU* Data MPU configuration after initial SW initialization and after every subsequent configuration change. In the event that the configured value does not match the expected value the Application SW shall trigger an alarm to the *SMU*.

Notes	The following registers should be checked CPUx_DPR*, CPUx_DPWE*, CPUx_DPRE*
References	Aurix TC3xx User Manual v1.0.0, Chapter 5.3.10
Recommendations	
Related safety mechanism	***SM[HW]:CPU:DATA_MPU***

5.29 ESM[SW]:CPU:INTERNAL_BUS_MONITOR

Description

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	N.A.	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

The *Application SW* shall detect random hardware faults in the internal buses and fifos of a non-lockstep *CPU*. This monitor should be executed cyclically, at least once in a diagnostic test interval. This Safety Mechanism is a measure to support the single point fault metric.

The Application SW shall :

	• Perform load and store sequences to local PSPR memory to check the FIFO paths through the CPU store buffers, The S_SRI interface, the Internal bus connectivity to the PSPR memory. • Perform Load and store access to the local DLMU memories. **Function** The ESM ESM[SW]:CPU:INTERNAL_BUS_MONITOR is run at start-up and then cyclically in order to detect latent and single CPU internal bus faults. **Areas of coverage** • CPU internal FIFOs (Store buffers, bus interfaces) • CPU internal bus connectivity to SRI Master and Slave interfaces (Read and Write) • CPU internal bus connectivity to DLMU **Requirements** • A 0x40 byte region of PSPR is required aligned on a 16 byte boundary (The same area as used for SBST can be used). • A 0x8 byte region of DLMU is required aligned on a 8 byte boundary. • The local data MPU must allow write access to the local PSPR and DLMU. • The bus MPU must allow PSPR read/write access for the local CPU data master. • Fault reaction should be the same as for the SBST • Interrupts must be disabled for the FIFO and bus test.
Notes	Pseudo Code and an implementation example in Assembler of this routine is provided in Appendix B.
References	
Recommendations	

5.30 ESM[SW]:CPU:SBST

Description

Detection and Control	**lockstep CPU**	**non-lockstep CPU**
Single Point Fault Detection	N.A.	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

For each non-lockstepped *CPU* used for safety-relevant applications, the *Application SW* shall execute the SM[SW]:CPU:SBST every *DTI*. In case the test returns a "fail" result, the non-lockstepped CPU shall be considered not reliable. It is responsibility of the System Integrator to define an appropriate error reaction.

Notes	Details on SM[SW]:CPU:SBST can be found on AURIX_2nd_generation CPU SBST User Manual, Ch. 4
References	AURIX_2nd_generation CPU Software based Self Test User Manual.
Recommendations	
Related safety mechanism	***SM[SW]:CPU:SBST***

5.31 ESM[SW]:CPU:SFR_TEST

Description

Detection and Control	**lockstep CPU**	**non-lockstep CPU**
Single Point Fault Detection	N.A.	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

The *Application SW* shall perform a verification of correct safety critical *CPU* CSFR and SFR register configuration after initial SW initialization, after every subsequent configuration change and cyclically every diagnostic time interval (DTI). In the event that the configured value does not match the expected value the Application SW shall trigger an alarm to the *SMU*.

Notes	The CSFR and SFR registers to be verified will be application dependent and should include all endinit (BTV, BIV, ISP, PMA0, PMA1, PMA2, PCON0, DCON0, SEGEN) and safety_endinit (SMACON, SYSCON, COMPAT, TPS_EXTIM_ENTRY_LVAL, TPS_EXTIM_EXIT_LVAL) protected registers.
References	Aurix TC3xx User Manual v1.0.0, Chapter 5.3.4.22.1
Recommendations	

5.32 ESM[SW]:CPU:SOFTERR_MONITOR

Description

Detection and Control	**lockstep CPU**	**non-lockstep CPU**
Single Point Fault Detection	N.A.	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

The *Application SW* shall implement logical and temporal monitoring mechanisms in order to detect wrong sequences or timing violations of the running programs due to *RHF*.

Notes	This ESM is covered by a combination of safety mechanisms and external safety mechanisms used in the application.
References	

Recommendations	

5.33 ESM[SW]:DMA.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:DMA.RAM:REG_MONITOR_TEST***

5.34 ESM[SW]:DMA:ADDRESS_CRC

Description	After the *DMA* has moved the data, the *Application SW* shall compare the DMA address checksum (stored in DMA channel DMA_SDCRCRc.SDCRC) calculated by the DMA with an expected DMA address checksum (stored in system memory) calculated by the Application SW. If the Application SW does not match the calculated and expected DMA address checksums, the Application SW shall signal ESM[SW]:DMA:SUPERVISION.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 18.3.4.4.6
Recommendations	
Related safety mechanism	***SM[HW]:DMA:ADDRESS_CRC***

5.35 ESM[SW]:DMA:DATA_CRC

Description	After the *DMA* has moved the data, the *Application SW* shall compare the DMA data checksum (stored in DMA channel DMA_RDCRCRc.RDCRC) calculated by the DMA with an expected DMA data checksum appended to the data. If the Application SW does not match the calculated and expected DMA data checksums, the Application SW shall signal ESM[SW]:DMA:SUPERVISION.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 18.3.4.10
Recommendations	

Related safety mechanism	***SM[HW]:DMA:DATA_CRC***

5.36 ESM[SW]:DMA:ERROR_HANDLING

Description	If the *DMA* triggers an error interrupt service request, the *Application SW* shall check the type of DMA error and the number of the DMA channel. If the Application SW determines the DMA error violates the safety goal, the Application SW shall signal ESM[SW]:DMA:SUPERVISION.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 18.6.2
Recommendations	
Related safety mechanism	***SM[HW]:DMA:ERROR_HANDLING*** ***SM[HW]:DMA:STI*** ***SM[HW]:DMA:SV***

5.37 ESM[SW]:DMA:SUPERVISION

Description	The *Application SW* shall configure the *DMA* (e.g. assignment of DMA channels to resource partitions), supervise the DMA (e.g. DMA clock control) and resolve all DMA errors. If the DMA triggers an alarm or the Application SW reports an error (i.e. ESM[SW]:DMA:ADDRESS_CRC, ESM[SW]:DMA:DATA_CRC, ESM[SW]:DMA:TIMESTAMP or ESM[SW]:DMA:ERROR_HANDLING), the Application SW shall trigger the most appropriate reaction, which depends on the ASIL allocation of the Application SW task which signalled the error and other Application SW constraints.
Notes	
References	
Recommendations	The actions taken by ESM[SW]:DMA:SUPERVISION may include repeating the DMA transaction with a redundant DMA channel to check the results, applying a DMA channel reset and restarting the DMA channel, applying a system reset, etc.

5.38 ESM[SW]:DMA:TIMESTAMP

Description	The *Application SW* may perform temporal monitoring of DMA transactions by the use of timestamps written on completion of a *DMA* transaction. A timestamp may be sourced from the *STM* and/or the DMA. <u>Timestamp sourced from STM</u> As soon as the DMA has completed a DMA transaction to move application data, the DMA performs an additional DMA transaction via a linked list operation to copy a timestamp from the STM to a DMA destination address. <u>Timestamp sourced from DMA</u>

	As soon as the DMA has completed a DMA transaction to move application data, the DMA performs an additional DMA write move to append a timestamp to the destination data sequence. (see Reference) Temporal Check After the DMA has written the timestamp, the Application SW shall compare the timestamp written by the DMA with an expected timestamp to determine if the DMA transaction was completed within a system deadline. If the Application SW determines the DMA transaction was completed after a system deadline, the Application SW shall signal ESM[SW]:DMA:SUPERVISION.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 18.3.4.4.6
Recommendations	
Related safety mechanism	***SM[HW]:DMA:TIMESTAMP***

5.39 ESM[SW]:DTS:DTS_RESULT

Description	The *Application SW* shall read temperature measurement results of two *DTS* blocks and compare them to detect single point and latent faults. The comparison is done by periodic readout of DTSSTAT.RESULT and DTSCSTAT.RESULT bits during runtime. The comparison of DTSSTAT.RESULT and DTSCSTAT.RESULT bits can be also done on an alarm generation by Application SW. Both DTS and *DTSC* over and under temperature monitor limits are configured to trigger respective *SMU* alarms at the same thresholds. Incase one of the DTS sensors trigger an over temperature alarm, the temperature results DTSSTAT.RESULT and DTSCSTAT.RESULT of both sensors can be checked against each other for plausibility check.
Notes	Due to the high probability of an unequal temperature distribution at different locations on the die and DTS inaccuracy, there will be a difference between two result values. The acceptable difference shall not exceed 9°C at every moment of time. Otherwise the Application SW shall trigger an alarm.
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.1.4 Device Datasheet 1.0, "Temperature sensor" chapter
Recommendations	

5.40 ESM[SW]:EDSADC:DIVERSE_REDUNDANCY

Description	The System Integrator shall implement redundant independent analog input channels using different A/D converters for each safety-relevant input signal. The *Application SW* shall compare the redundant result values from mission channel and monitor channel against each other and against system-specific limit values (see ESM[SW]:EDSADC:VAREF_PLAUSIBILITY and ESM[SW]:EDSADC:PLAUSIBILITY).

	If the redundant result values do not match within a defined band of uncertainty, the Application SW shall notify the system about the presence of a fault.
Notes	A digital input channel can be used to check the carrier generator signal which delivers a known data sequence. This can be accomplished by using the digital input of an EDSADC channel. Alternatively, the PWM signal can be fed back, e.g. with the port loopback function, to the *GTM TIM* to determine the maximum/minimum of the intended carrier signal. The GTM then triggers an *EVADC* conversion to verify and compare the max/min level of the generated carrier output signal with the intended max/min value. In the case of deviations Application SW shall notify the system about the presence of a fault.
References	
Recommendations	Diversity of result value can be enhanced by using different integration levels on the two channels, resulting in different values for the same result.

5.41 ESM[SW]:EDSADC:PLAUSIBILITY

Description	After a conversion, the *Application SW* shall determine the validity of result values by comparing the redundant result values of mission channel and monitor channel against each other and against system-specific limit values. If the two result values differ more than a defined band of uncertainty (owing to noise or differences in the physical paths), the Application SW shall notify the system about the presence of a fault.
Notes	The error notification and reaction is fully dependent on the Application.
References	
Recommendations	The result values can also be validated against defined upper/lower limit values. If the dynamics of the input signal are known, Application SW can also evaluate the difference to the previous result value(s).

5.42 ESM[SW]:EDSADC:VAREF_PLAUSIBILITY

Description	The System Integrator shall implement a check of the ADC reference voltage (VAREF / VAGND) either by an external monitor or by internally converting a known signal and compare the result with the expected value. This check shall be executed at least once per driving cycle or better cyclically after a predefined time or number of conversions. If the *Application SW* detects unexpected results during these checks, it shall notify the system about the presence of a fault.
Notes	1. The error notification and reaction is fully dependent on the Application. 2. Typically, when executing this safety mechanism, two conversions take place and VAREF comparison is done. Since only a low discharging is applied, an additional resistance of several tens of kOhms which could be caused by an external failure effect does not play that major role. This leads to the test being considered as passed. However, when the application SW is switching to normal operating mode (e.g. 200 conversions every ms), the discharge of the VAREF

	input is significantly higher and a systematical failure will happen on all channels due to the increased additional resistance which could lead to the violation of application safety goal. Therefore, it is recommended that this check is performed under representative application conditions.
References	
Recommendations	

5.43 ESM[SW]:EMEM.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:EMEM.RAM:REG_MONITOR_TEST***

5.44 ESM[SW]:EMEM:DATA_INTEGRITY

Description	In case the System Integrator wants to use EMEM for ASIL-D applications, the *Application SW* shall monitor the corruption of data stored in the EMEM caused by random hardware faults by using information redundancy.
Notes	
References	
Recommendations	The Application SW shall store a block of data in the EMEM with a data checksum accumulated over all the data in the block. The calculation of the checksum should be of a suitably robust implementation (e.g. IEEE 802.3 standard or AUTOSAR CRC32P4 standard). If the Application SW uses the data stored in the *EMEM*, the Application SW shall first calculate a data checksum for all the data stored in the block. On completion of the calculation the Application SW shall compare the calculated data checksum with the expected data checksum for the store data. If the Application SW matches the calculated data checksum and the expected data checksum, the Application SW may use the data. If the Application SW does not match the calculated data checksum and the expected data checksum, the Application SW shall signal an alarm to the *SMU*.

5.45 ESM[SW]:EMEM:READ_CONFIGURATION

Description	After configuration, the *Application SW* shall check the mode of the *EMEM* tiles by reading the EMEM_TILECC, EMEM_TILESTATE and EMEM_SBRCTR registers to detect transient faults and a change in the interface access permissions to the EMEM tiles.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 21.3.3
Recommendations	
Related safety mechanism	***SM[HW]:EMEM:READ_WRITE_MUX*** ***SM[HW]:EMEM:READ_WRITE_SRI***

5.46 ESM[SW]:ERAY.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:ERAY.RAM:REG_MONITOR_TEST***

5.47 ESM[SW]:ERAY:SAFE_COMMUNICATION

Description	The *Application SW* shall implement Safe Communication which detects random HW faults, systematic faults and interference that can lead to message corruption and protocol violation, including temporal and random faults of safety related data transferred over FlexRay interface. Safe Communication can be achieved by implementing a combination of safety measures like information redundancy (CRC & Data-ID), frame counter and timeout monitoring. The safety measures are facilitated by using additional meta-data like CRC, Counters and timestamps as part of the payload data. The [Application Software] has to initiate the Safe Communication safety measures during every communication event (Transmission & Reception). Upon detection of a failure, the Application SW shall trigger the reaction.
Notes	The effectiveness of failure coverage depends on the safety measures used for the Safe Communication protection.

References	AUTOSAR Release 4.3.1 - Specification of SW-C End-to-End Communication Protection Library.
Recommendations	End-to-end safety protocol can be designed and tailored according to ISO26262-6 Annex D and IEC62280/EN50159-1,-2.

5.48 ESM[SW]:EVADC:CONFIG_CHECK

Description	The *Application SW* shall verify the write access to safety relevant registers GxANCFG and GxSYNCTR (x=0..11) by reading the respective register and check the returned value.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 30.11.1, 30.11.3
Recommendations	

5.49 ESM[SW]:EVADC:DIVERSE_REDUNDANCY

Description	The [System Integrator] shall implement redundant independent analog input channels using different A/D converters for each safety-relevant input signal. The *Application SW* shall compare the redundant result values from mission channel and monitor channel against each other and against system-specific limit values (see ESM[SW]:EVADC:PLAUSIBILITY and ESM[SW]:EVADC:VAREF_PLAUSIBILITY). If the redundant result values do not match within a defined band of uncertainty, the Application SW shall notify the system about the presence of a fault.
Notes	The error notification and reaction is fully dependent on the Application.
References	
Recommendations	Diversity of result value can be enhanced by using different accumulation levels on the two channels, resulting in different values for the same result.

5.50 ESM[SW]:EVADC:PLAUSIBILITY

Description	After a conversion, the *Application SW* shall determine the validity of result values by comparing the redundant result values of mission channel and monitor channel against each other and against system-specific limit values. If the two result values differ more than a defined band of uncertainty (owing to noise or differences in the physical paths), the Application SW shall notify the system about the presence of a fault.
Notes	The error notification and reaction is fully dependent on the Application.
References	
Recommendations	The result values can also be validated against defined upper/lower limit values. If the dynamics of the input signal are known, Application SW can also evaluate the difference to the previous result value(s).

5.51 ESM[SW]:EVADC:VAREF_PLAUSIBILITY

Description	The System Integrator shall implement a check of the ADC reference voltage (VAREF / VAGND) either by an external monitor or by internally converting a known signal and compare the result with the expected value. This check shall be executed at least once per driving cycle or better cyclically after a predefined time or number of conversions. If the *Application SW* detects unexpected results during these checks, it shall notify the system about the presence of a fault.
Notes	1. The error notification and reaction is fully dependent on the Application. 2. Typically, when executing this safety mechanism, two conversions take place and VAREF comparison is done. Since only a low discharging is applied, an additional resistance of several tens of kOhms which could be caused by an external failure effect does not play that major role. This leads to the test being considered as passed. However, when the application SW is switching to normal operating mode (e.g. 200 conversions every ms), the discharge of the VAREF input is significantly higher and a systematical failure will happen on all channels due to the increased additional resistance which could lead to the violation of application safety goal. Therefore, it is recommended that this check is performed under representative application conditions.
References	
Recommendations	The reference voltage can be validated either by using an alternate reference voltage supplied via CH0 of a converter group, or by converting an internal bandgap-based voltage.

5.52 ESM[SW]:GETH.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:GETH.RAM:REG_MONITOR_TEST***

5.53 ESM[SW]:GETH:SAFE_COMMUNICATION

Description	The *Application SW* shall implement Safe Communication]which detects random HW faults, systematic faults and interference that can lead to message corruption and protocol violation, including temporal and random faults of safety related data transferred over Ethernet interface. Safe Communication can be achieved by implementing a combination of safety measures like information redundancy (CRC & Data-ID), frame counter and timeout monitoring. The Application SW has to initiate the [Safe Communication] safety measures during every communication event (Transmission & Reception). Upon detection of a failure, the Application SW shall trigger the reaction.
Notes	The effectiveness of failure coverage depends on the safety measures used for the Safe Communication protection.
References	AUTOSAR Release 4.3.1 - Specification of SW-C End-to-End Communication Protection Library.
Recommendations	End-to-end safety protocol can be designed and tailored according to ISO26262-6 Annex D and IEC62280/EN50159-1,-2

5.54 ESM[SW]:GTM.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:GTM.RAM:REG_MONITOR_TEST***

5.55 ESM[SW]:GTM:GTM_CCU6_REDUNDANCY

Description	The *Application SW* shall compare the last input signal of GTM TIM against the corresponding CCU6 capture. In case the comparison is outside the allowed tolerance, the Application SW shall notify the system about the presence of a fault.
Notes	The error notification and reaction is fully dependent on the Application. In alternative, GPT12 can be used instead of CCU6.
References	

Recommendations	

5.56 ESM[SW]:GTM:GTM_TIM_REDUNDANCY

Description	The *Application SW* shall provide means to detect faults, in a scenario, where two TIMs of different clusters are operated, which are having the same input pattern. The result shall be compared by the Application SW. In case Application SW detects a mismatch then Application SW shall trigger a reaction. To ensure, that the configuration has been written properly, the configuration should be read back and compared after writing the complete configuration. In addition to the compare check upfront, a configuration register needs to be checked regularly, to ensure that read works properly. To avoid clocking problems, the ESM[SW]:GTM:TIM_CLOCK_MONITORING shall be used.
Notes	
References	
Recommendations	

5.57 ESM[SW]:GTM:IOM_ALARM_CHECK

Description	Once per driving cycle, the *Application SW* shall test the correctness of *IOM* alarm generation by configuring *GTM* and IOM for triggering an alarm under known conditions. First known condition: Pass test: the IOM shall get no fault: No alarm shall be generated. Second known condition: Fail test: the IOM shall get an erratic pattern: A fault shall be signaled via alarm. In case of fail test, an alarm interrupt shall be raised.
Notes	
References	
Recommendations	
Related safety mechanism	***SM[HW]:GTM:TOM_CCU6_MONITORING_WITH_IOM*** ***SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM***

5.58 ESM[SW]:GTM:TIM_CLOCK_MONITORING

Description	The *Application SW* shall check if the *GTM* is clocked. Therefore the ECLK (derived vom CLS0_CLK and not from CMU_CLKx) signal shall be read back by the *TIM* and evaluated, if the value is within the expected timing. To make sure, that configuration has been written, the complete configuration for this monitoring shall be read back once.
Notes	This check has to be done per safety cluster within the GTM.
References	
Recommendations	

5.59 ESM[SW]:GTM:TOM_TIM_MONITORING

Description	The *Application SW* shall check, if the *GTM* is clocked and if the PWM signal is within the expected range. The necessary initial configuration shall be written as one block and read back in one block and shall be compared to the initial static configuration. To check the clock, the mechanism ESM[SW]:GTM:TIM_CLOCK_MONITORING shall be used. If this check is OK, a second *TIM* (on the same clock base as inside the TIM_CLOCK_MONITORING) shall read back a known PWM signal from a DTM (DTM_OUTx_N) output (*FB PORT*). If DTM is not used, a TOM/ATOM signal can be used. Pass: a) The clock signal and the PWM signal coming out the FB PORT is within the tolerance range +/-1 timer ticks. Fail: The measured clock signal or PWM signal is out of the tolerance range +/-1 timer ticks. In case of a fail, an alarm shall be raised.
Notes	• This check has to be done per safety cluster within the GTM. • To achieve complete freedom of interference from Advanced Routing Unit access to ATOM channels, the ATOM channel shall be taken out of the round robin access scheme of the Advanced Routing Unit. Once this is achieved then ATOM channel is similar to TOM channel.
References	
Recommendations	

5.60 ESM[SW]:HSPDM.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.

5.61 ESM[SW]:HSSL:SAFE_COMMUNICATION

Description	The *Application SW* shall implement Safe Communication which detects random HW faults, systematic faults and interference that can lead to message corruption

	and protocol violation, including temporal and random faults of safety related data transferred over HSSL interface. Safe Communication can be achieved by implementing a combination of safety measures like information redundancy (CRC & Data-ID), frame counter and timeout monitoring (For exa., using timestamps). The safety measures are facilitated by using additional meta-data like CRC, Counters and timestamps as part of the payload data. The Application SW has to initiate the Safe Communication safety measures during every communication event (Transmission & Reception). Upon detection of a failure, the Application SW shall trigger the reaction.
Notes	The effectiveness of failure coverage depends on the safety measures used for the Safe Communication protection.
References	AUTOSAR Release 4.3.1 - Specification of SW-C End-to-End Communication Protection Library.
Recommendations	End-to-end safety protocol can be designed and tailored according to ISO26262-6 Annex D and IEC62280/EN50159-1,-2.

5.62 ESM[SW]:IR:ISR_MONITOR

Description	Monitors Interrupt Service Requests to a given *CPU* or *DMA*. The *Application SW* shall detect missing or unintended service requests for safety-related interrupts. On detection of a missing or unintended safety-related interrupt request, the Application SW shall trigger the most appropriate reaction, depending on the application.
Notes	Faults in the *IR* are covered by other SMs of IR.This ESM shall detect faults in the interrupt sources. The following methods are given as hints to implement the monitoring. On detecting a failure, the application software shall trigger an appropriate reaction. 1. Using redundant interrupt sources: The application may use at least two redundant sources for the interrupt. The faults in the sources of the interrupts are detected via this redundancy. 2. Plausibility check for periodic interrupts: Certain interrupts are expected periodically (e.g. ADC triggering every 1ms). Based on this expected rate (or for example within each FTTI) – the application shall implement periodic checks for the plausibility of such interrupts – i.e. If an expected interrupt is missing, or unintended interrupts occur. NOTE: For interrupts that are expected to be triggered only once during a driving cycle (e.g. at startup or configuration), a periodic check every FTTI shall be implemented to check the corresponding SRC register in the IR to ensure that the interrupt was not triggered unexpectedly later. 3. Application level mechanisms: For non-periodic interrupts – application dependent mechanisms can be implemented. For example, the interrupt source itself can be periodically checked (i.e. the interrupt status register in the peripheral) – or program flow monitoring can be implemented. For communication peripherals – system level mechanisms (e.g. detection of failure external to the MCU) shall be implemented, or other redundancy measures such as CRC or Timestamp. 4. Disabled Interrupts: For interrupts that are not enabled / expected to be triggered: The SRC register shall be checked once every driving cycle for an unexpected trigger.

References	AUTOSAR 4.3.0 Overview of Functional Safety Measures in AUTOSAR, Chapter 2.2 Timing Monitoring
Recommendations	Example: the combination of Alive Supervision (missing interrupt) and Inter-Arrival Time Protection (unintended interrupt). Implementation of redundant source for interrupt generation and comparison of generated interrupt.

5.63 ESM[SW]:LMU.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:LMU.RAM:REG_MONITOR_TEST***

5.64 ESM[SW]:MCMCAN.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:MCMCAN.RAM:REG_MONITOR_TEST***

5.65 ESM[SW]:MCMCAN:SAFE_COMMUNICATION

Description	The *Application SW* shall implement Safe Communication which detects random HW faults, systematic faults and interference that can lead to message corruption and protocol violation, including temporal and random faults of safety related data transferred over CAN interface. Safe Communication can be achieved by implementing a combination of safety measures like information redundancy (CRC & Data-ID), frame counter and timeout monitoring. The safety measures are facilitated by using additional meta-data like CRC, Counters and timestamps as part of the payload data. The Application SW has to initiate the Safe Communication safety measures during every communication event (Transmission & Reception). Upon detection of a failure, the Application SW shall trigger the reaction.
Notes	The effectiveness of failure coverage depends on the safety measures used for the Safe Communication protection.
References	AUTOSAR Release 4.3.1 - Specification of SW-C End-to-End Communication Protection Library.
Recommendations	End-to-end safety protocol can be designed and tailored according to ISO26262-6 Annex D and IEC62280/EN50159-1,-2.

5.66 ESM[SW]:MCU:LBIST_RESULT

Description	The *Application SW* shall check the LBIST results and compare it with the expected value after LBIST execution is finished. Before checking the MISR signature the Application SW shall verify if the previously triggered LBIST-run terminated successfully by checking if LBISTCTRL0.LBISTDONE='1' and RSTSTAT.LBTERM='1'. Afterwards the 32-bit MISR-signature result from LBISTCTRL3.SIGNATURE shall be compared with the expected MISR-result. In this conjunction it shall be regarded that the expected MISR-signature result is depending on the product version, the number of applied LBIST-patterns (LBISTCTRL0.PATTERNS) and the LBIST-config bits in LBISTCTRL1-register (including the BODY-bit). For a successful LBIST-execution it is essential that the default value in LBISTCTRL2.LENGTH field has not been modified by Application SW.
Notes	It shall be regarded that LBIST is a structural test method which is causing a high switching activity and requires stable environmental voltage and temperature conditions. Therefore it might happen that under extreme operation conditions a single LBIST run might fail by showing an unexpected MISR-result. In this case the faulty silicon behavior shall be confirmed by repeating the LBIST execution at least 2 times. If the LBIST function should fail to produce a correct MISR-signature within 3 consecutive runs, the device has to be considered as defect and shall not boot into a safe application.
References	Aurix TC3xx User Manual v1.0.0, Chapter 9.3.3 Device Appendix
Recommendations	After MISR-signature evaluation it is recommended to bring the LBIST-controller to a safe reset-state by setting LBISTCTRL0.LBISTRES='1'.

	This will clear the MISR-signature in LBISTCTRL3-register but is necessary to de-assert the LBIST SMU-alarm ALM8[5] bit.
Related safety mechanism	***SM[HW]:MCU:LBIST***

5.67 ESM[SW]:NVM.PFLASH:INTEGRITY_CHECK

Description	Within Latent Fault Diagnostic Time Interval, the *Application SW* shall compare the NVM.PFLASH checksum of all safety related code with an expected NVM.PFLASH checksum. If the Application SW does not match the calculated and expected NVM.PFLASH checksums, the Application SW shall trigger an alarm.
Notes	This SM is aiming to reduce LFM of PFLASH; Indeed, since ECC can only detect errors during read accesses, all non-read location can potentially accumulate errors (latent faults).
References	
Recommendations	In order to locate the errors present in the PFLASH at startup, IFX recommends to read the entire PFLASH and incidentally compute a CRC over the data. The User will be notified by an SMU alarm that the PFLASH error management (i.e. error buffers) has detected an unexpected high rate of latent faults. Note: Without using this integrity check, undetected latent faults can accumulate into uncorrectable MBEs. A low priority background scan during one driving cycle could be sufficient. Real NVM.PFLASH checksum could be calculated using *DMA* to read NVM.PFLASH and *FCE* to calculate CRC. This would reduce *CPU* load. Expected NVM.PFLASH checksum could be re-used from stored value in ESM[SW]:NVM.PFLASH:UPDATE_CHECK.

5.68 ESM[SW]:NVM.PFLASH:UPDATE_CHECK

Description	After update of NVM.PFLASH content and before starting safety related *Application SW*, Application SW shall compare the NVM.PFLASH checksum of all safety related code with an expected NVM.PFLASH checksum. If the Application SW does not match the calculated and expected NVM.PFLASH checksums, the Application SW shall trigger an alarm.
Notes	This SM is aiming to reduce LFM of PFLASH; Indeed, since ECC can only detect errors during read accesses, all non-read location can potentially accumulate errors (latent faults). Without using this integrity check, undetected latent faults can accumulate into uncorrectable MBEs.
References	
Recommendations	Real NVM.PFLASH checksum could be calculated using *DMA* to read NVM.PFLASH and *FCE* to calculate CRC. This would reduce *CPU* load. Expected NVM.PFLASH checksum could be calculated by [Customer Production Equipment] from data to be written into NVM.PFLASH and stored in *NVM* after NVM.PFLASH update for further use in ESM[SW]:NVM.PFLASH:INTEGRITY_CHECK. In order to locate the errors present in the PFLASH after content update, IFX recommends to read the entire PFLASH and incidentally compute a CRC over the data. The User will be notified by an SMU alarm that the PFLASH error

	management (i.e. error buffers) has detected an unexpected high rate of latent faults.

5.69 ESM[SW]:NVM.PFLASH:WL_FAIL_DETECT

Description	Upon the occurrence of every *SMU* alarm related to a DBE correction, the *Application SW* shall verify the integrity of the PFLASH wordline in which the DBE occurred by reading 10 pages and monitoring the multi bit errors. Background: If a DBE is found, the probability that the complete wordline is broken is higher. Further multi bit fails might be visible on the remaining pages on the same wordline (wordline fail).
Notes	The multi bit errors are monitored and signalled to Application SW via an SMU alarm using SM[HW]:NVM.PFLASH:ERROR_MANAGEMENT. No further action is needed by Application SW.
References	
Recommendations	

5.70 ESM[SW]:PMS:MONBIST_RESULT

Description	The *Application SW* shall check the MONBIST results in MONBISTSTAT register after MONBIST execution is finished.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.2.5.4
Recommendations	
Related safety mechanism	***SM[HW]:PMS:MONBIST***

5.71 ESM[SW]:PORT:LOOPBACK

Description	*Application SW* shall configure and use redundant GPIO communication to detect faults (when receiving information) or allow the receiver to detect faults (when transmitting information). An appropriate reaction on a fault is application dependent.
Notes	The need for this ESM and its implementation is application dependent. This note can only offer application hints. The assumed scenario is that the Application SW drives a safety critical output value (called signal "S1") on pin Px.y. SPF in the Port logic, Pad logic, bond wire, package, the PCB or the receiving component might cause the receiver to detect a value different from S1. Comparison of S1 against a redundant signal S2 can help monitor results within the Fault Tolerant Time Interval ("FTTI") of the application. The type of faults detected depends on the implementation of measures undertaken
References	
Recommendations	Implementation Examples:

	1. Application level plausibility: The control loop of the application may also be used to read a signal S2, enabling detection of wrong values S1 (e.g. motion sensor used to check activation of motor). **2.** Receiver confirmation: the receiver of signal S1 returns on a separate communication channels the received value of S1 (or an inverted version) to a different pin Pa.b. If the receiver pin uses a different port module this redundancy covers also faults in port module "x". **3.** PCB level return path: on the PCB the pin Px.y is connected to receiving pin Pa.b. The Application SW can detect faults on the path to the PCB wire. **4.** Pad internal loopback: the Application SW reads back the value of pin Px.y detecting faults that cause a wrong value on the pad output.

5.72 ESM[SW]:PORT:REDUNDANCY

Description	*Application SW* shall configure and use redundant GPIO communication to detect faults (when receiving information) or allow the receiver to detect faults (when transmitting information). An appropriate reaction on a fault is application dependent.
Notes	The need for this ESM and its implementation is application dependent. This note can only offer application hints. The assumed scenario (A) is that the Application SW drives a safety critical output value (called signal "S1") on pin Px.y to an external receiver. In order to enable the receiver to detect faults on S1 the Application SW drives a redundant signal "S2" on Pa.b carrying the same information (e.g. the same signal as S1 or inverted S1) to the receiver. The assumed scenario (B) is that the Application SW receives a safety critical input value (called signal "S1") on pin Px.y from an external sender. In order to detect faults on S1 it receives on pin Pa.b the same information on signal S2. Cyclically it compares the information on signals S1 and S2. SPF in the Port logic, Pad logic, bond wire, package, the PCB or the external component might cause the receiver to detect a value different from S1. By using a redundant signal S2 as described above such faults can be detected. The type of faults detected depends on the implementation of measures undertaken.
References	
Recommendations	Implementation Examples: **1.** Pin redundancy: the PCB wire transmitting signal S1 is connected to two pins Px.y and Pa.b. If both are located in different Port modules (i.e. "a" unequal to "x") this redundancy covers also faults in Port "x" logic. **2.** End to end redundancy: the transmitter drives signal S2 on different pin, pcb wire to a different pin of the receiver.

5.73 ESM[SW]:PSI5.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:PSI5.RAM:REG_MONITOR_TEST***

5.74 ESM[SW]:PSI5:CHANNEL_REDUNDANCY

Description	The *Application SW* shall use two or more independent *PSI5* channels for acquiring data from one or more sensors. Upon receiving a new value, the results captured by the independent channels shall be compared by the Application SW. In case Application SW detects a mismatch then Application SW shall trigger an appropriate reaction.
Notes	The sensors configuration (number, redundancy, etc.) is application dependent and shall be defined by the System Integrator. The configuration settings over TX can be covered by RX read back and compared with reference value.
References	
Recommendations	

5.75 ESM[SW]:QSPI:SAFE_COMMUNICATION

Description	The *Application SW* shall implement Safe Communication which detects random HW faults, systematic faults and interference that can lead to message corruption and protocol violation, including temporal and random faults of safety related data transferred over SPI interface. Safe Communication can be achieved by implementing a combination of safety measures like information redundancy (CRC & Data-ID), frame counter and timeout monitoring. The safety measures are facilitated by using additional meta-data like CRC, Counters and timestamps as part of the payload data. The Application SW has to initiate the Safe Communication safety measures during every communication event (Transmission & Reception). Upon detection of a failure, the Application SW shall trigger the reaction.

Notes	The effectiveness of failure coverage depends on the safety measures used for the Safe Communication protection.
References	AUTOSAR Release 4.3.1 - Specification of SW-C End-to-End Communication Protection Library
Recommendations	End-to-end safety protocol can be designed and tailored according to ISO26262-6 Annex D and IEC62280/EN50159-1,-2.

5.76 ESM[SW]:RIF:ERROR_HANDLING

Description	If the *RIF* triggers an error interrupt service request, the *Application SW* shall check the type of RIF error. If the Application SW determines the RIF error violates the safety goal, the Application SW shall trigger an alarm.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 22.3.13.2
Recommendations	
Related safety mechanism	***SM[HW]:RIF:ERROR_HANDLING***

5.77 ESM[SW]:RIF:SM_CHECK

Description	The *Application SW* shall monitor the status of the safety mechanism SM[HW]:RIF:SM_CHECK by polling the Safety Mechanism Control Status (INTCON.SMCF = 1_B) and take appropriate measures on detecting that a missing alarm has occurred.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 22.3.4
Recommendations	
Related safety mechanism	***SM[HW]:RIF:CFG_MONITOR*** ***SM[HW]:RIF:SM_CHECK***

5.78 ESM[SW]:SCR.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the SPB clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code

	implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:SCR.RAM:REG_MONITOR_TEST***

5.79 ESM[SW]:SDMMC.RAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:SDMMC.RAM:REG_MONITOR_TEST***

5.80 ESM[SW]:SENT:CHANNEL_REDUNDANCY

Description	The *Application SW* shall use two or more independent *SENT* channels for acquiring data from one or more sensors. Upon receiving a new value, the results captured by the independent channels shall be compared by the Application SW. In case Application SW detects a mismatch then Application SW shall trigger an appropriate reaction.
Notes	The sensors configuration (number, redundancy, etc.) is application dependent and shall be defined by the System Integrator. If two or more redundant sensors are used by two or more redundant SENT channels then the SPC trigger from *GTM* should also be redundant to avoid common cause failure. At least two (set of) sensors are needed to build a redundant SPC functionality. It is assumed that the error detection mechanisms of each redundant SENT channels are being used and the resulting flags in the Interrupt Status Register are being taken into account. When the timestamp feature is being used, the timestamp values shall be checked for plausibility.

	With the objective to detect single point faults in the data path from the SENT SFR to the *CPU* the Application SW shall verify each write to a SENT SFR by reading back the same SFR and check the read data for correctness. Each SENT channel publishes the CRC of the received SENT frame in its Receive Status Register (RSRx.CRC). The Application SW shall read this CRC along with the data and the Status and Communication Nibble (RSRx.SCN). The Application SW shall verify the correctness of the data and SCN using this CRC. Similarly the Application SW shall read and verify the correctness fo the data of the SENT "Slow Channel" using the CRC of the "Slow Channel". To avoid systematical faults it is recommended to build the redundant system with at least one communication channel that is not based on SENT communication, e.g. with an analog sensor and an AD converter.
References	
Recommendations	

5.81 ESM[SW]:SMU:ALIVE_ALARM_TEST

Description	The *Application SW* shall, at least once per driving cycle, test the SMU core alive monitor and its connection to the SMU standby by triggering the alive alarm using SMU_AliveTest() command as described in the user manual.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 15.3.1.2.6
Recommendations	
Related safety mechanism	***SM[HW]:SMU:ALIVE_MONITOR***

5.82 ESM[SW]:SMU:APPLICATION_SW_ALARM

Description	The *Application SW* can set and clear these 16 alarms independently during runtime by using SMU_CMD register.
Notes	This ESM can be used as part of the 'Application SW Error Handler'.
References	Aurix TC3xx User Manual v1.0.0, Chapter 15.3.1.6
Recommendations	

5.83 ESM[SW]:SMU:REG_MONITOR_TEST

Description	The *Application SW* shall execute the REGISTER MONITOR test of all relevant functional blocks by setting the related bit in RMCTL register. Once started, [Application SW] shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading RMEF register.
Notes	• Execution time depends on the FB under test and its clock settings.

	• The corresponding "safety flip-flop uncorrectable error" of the functional blocks under test are activated. Therefore SMU alarms shall be cleared by Application SW once the test is completed.
References	Aurix TC3xx User Manual v1.0.0, chapter 15.3.1.2.4 Device Appendix v1.0.0, SMU chapter.
Recommendations	Test execution time can be measured by polling RMSTS.STS bit related to the functional block under test. The worst case execution time is highly dependant on the system integrator's code implementation and the selected clock settings for the FBs under test. Assuming that the default clock settings are used for all FBs, it is recommended to use 2us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:CLOCK:CFG_MONITOR*** ***SM[HW]:CLOCK:FPI_WRITE_MONITOR*** ***SM[HW]:DMA:REQUEST_MONITOR*** ***SM[HW]:EMEM:FPI_WRITE_MONITOR*** ***SM[HW]:GTM:FPI_WRITE_MONITOR*** ***SM[HW]:IR:FPI_WRITE_MONITOR*** ***SM[HW]:IR:REG_MONITOR*** ***SM[HW]:PMS:REG_MONITOR*** ***SM[HW]:RESET:FF_MONITORING*** ***SM[HW]:RESET:FPI_WRITE_MONITOR*** ***SM[HW]:SCU:FPI_WRITE_MONITOR*** ***SM[HW]:SCU:REG_MONITOR*** ***SM[HW]:SMU:FPI_WRITE_MONITOR*** ***SM[HW]:SMU:REG_MONITOR_TEST*** ***SM[HW]:VMT:FPI_WRITE_MONITOR***

5.84 ESM[SW]:SPU.BUFFER:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time.

	This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:SPU.BUFFER:REG_MONITOR_TEST***

5.85 ESM[SW]:SPU.CONFIG:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, [Application SW] shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:SPU.CONFIG:REG_MONITOR_TEST***

5.86 ESM[SW]:SPU.FFT:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:SPU.FFT:REG_MONITOR_TEST***

5.87 ESM[SW]:SPU:CONTROL_FLOW_SIGNATURE

Description	The *Application SW* shall monitor the *SPU* control flow signature. As soon as the SPU has completed an operation or linked list, the Application SW shall check the control flow signature. As soon as the Application SW has completed the check of the control flow signature, the Application SW shall clear all CRC registers to all zeros (for e = 0-n, SPU_CTRLe_CRC = all 0's). The System integrator shall use as a reference signature the signature generated by running the user's static configuration through the system integrator test environment.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.3
Recommendations	
Related safety mechanism	***SM[HW]:SPU:CONTROL_FLOW_SIGNATURE***

5.88 ESM[SW]:SPU:DATA_INTEGRITY

Description	If the SPU is not configured for full redundancy, the *Application SW* shall make data faults safe or detect data faults.
Notes	Transient faults in the datapath or the control logic of the SPU will be cleared when the SPU is initialised for running the next configuration. Transient faults in the configuration will either be overwritten when a new configuration is loaded or detected by SM[HW]:SPU:ACTIVE_CFG_CHECK. Transient faults will therefore not be able to propagate across measurement cycles without detection.
References	
Recommendations	

5.89 ESM[SW]:SPU:EXECUTION_TIME_CHECK

Description	The execution time for SPU processing is deterministic and repeatable for a given configuration or autonomous set of configurations. However, faults in the SPU control logic can lead to the SPU failing to complete its activities. In such a case the resultant failure mode may not be detected by one of the existing safety mechanisms since some of these rely on the completion of processing before the safety mechanism is activated. Therefore the application software should confirm that the SPU has executed its processing within the expected time window. The time window is dependent upon configuration and therefore the System integrator shall determine the reference time window by running the user's static configuration through the system integrator's test environment.
Notes	This safety mechanism will also cover the corner case on the RIF to SPU interface where a data word is missng on the final Ramp of a measurement cycle causing the SPU to stall waiting for the CRC word (input data stream, one or more words short in the final ramp of a measurement cycle). Missing words in earlier ramps of the measurement cycle will be detected by the CRC check.
References	

Recommendations	

5.90 ESM[SW]:SPU:SBST

Description	The *Application SW* shall execute the SM[SW]:SPU:SBST every *DTI*. In case the test returns a "fail" result, the SPU engine shall be considered not reliable. It is responsibility of the System Integrator to define an appropriate error reaction.
Notes	Details on SM[SW]:SPU:SBST can be found on SPU User Aurix_2nd_generation SPU Software Based Self Test (SBST) User Manual, Ch. 4.
References	AURIX_2nd_generation SPU Software based User Manual.
Recommendations	The SM[SW]:SPU:SBST is intended to operate in a system while interrupts are enabled. The [application SW] shall ensure that the processing of interrupts occurring during execution of SM[SW]:SPU_SBST does not cause the execution time to exceed the DTI. The Application SW shall include temporal monitoring mechanisms to detect livelock or deadlock occurring during the execution of SM[SW]:SPU:SBST. There are three possible ESMs that can be used to implement this. In preference order they are: **1.** ESM[SW]:SPU:EXECUTION_TIME_CHECK. Any SPU fault can be localized to SPU under test. Impact can be restricted to the SPU **2.** ESM[SW]:SYS:SW_SUPERVISION. Any SPU fault will be localised to either the SPU or the CPU running the SBST. Impact will be restricted to the CPU and the SPU **3.** ESM[HW]:SYS:WATCHDOG_FUNCTION. Any SPU Fault will impact availability of the MCU The Application SW shall ensure that the SM[SW]:SPU:SBST is executed using the Supervisor mode of the *CPU*.
Related safety mechanism	***SM[SW]:SPU:SBST***

5.91 ESM[SW]:SPU:SM_CHECK

Description	The *Application SW* shall monitor the status of the safety mechanism check by polling the Safety Mechanism Control Status (SPU_SMSTAT SMCTRLSTS = 1_B).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.2
Recommendations	
Related safety mechanism	***SM[HW]:SPU:ACTIVE_CFG_CHECK*** ***SM[HW]:SPU:BYPASS_CRC*** ***SM[HW]:SPU:EMEM_INTERFACE*** ***SM[HW]:SPU:PARTIAL_REDUNDANCY*** ***SM[HW]:SPU:REDUNDANCY*** ***SM[HW]:SPU:SM_CHECK***

5.92 ESM[SW]:SRI:ERROR_HANDLING

Description	If the *SRI* triggers an error interrupt service request, the *Application SW* shall check the type of SRI error and the stored SRI diagnostic information. The Application SW shall evaluate the type of SRI error and trigger the most appropriate reaction. The type of an SRI error is readable via the registers ERRADDRx (x=0..15), ERRx, PESTAT, TIDSTAT, and PECON.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 4.3.3
Recommendations	
Related safety mechanism	***SM[HW]:SRI:ERROR_HANDLING***

5.93 ESM[SW]:STM:MONITOR

Description	The *Application SW* shall perform a plausibility check of the STM counter value with a different timer/counter, before using it. For example, in a multi-core system with multiple STMs, the counter value shall be compared against another STM timer value, or in a single CPU system, the CPU clock counter (CCNT), or the DMA_TIME register. The check shall be performed in any case before using the counter value. It may be also be performed periodically (e.g. before a smaller width counter runs out). In case the STM counter value read out is not plausible, then the application may try reading the STM again (up to N times, application dependent)- and in case of consistent failure, a safety critical reaction shall be performed (e.g. application reset).
Notes	In case the *STM* is used as a ISR trigger / timer base, then the application software shall additionally perform a temporal plausibility check.
References	
Recommendations	Application Software shall configure the CPU TPS timer (or another STM timer interrupt) as a redundant timer source. This redundant timer source shall be reinitialized within the interrupt service routine of the STM interrupt. In addition, within each STM interrupt service, the plausibility of the counter value shall be checked to ensure the plausibility of the interrupt triggering time. If the redundant timer (e.g. TPS, or another STM timer) expires or if the STM IR occurs and the STM counter under supervision is not within the expected range values, the Application SW shall trigger an appropriate reaction (e.g. application reset).

5.94 ESM[SW]:SYS:MCU_FW_CHECK

Description (Cold Power-on Reset)	Before entering the run mode, the *Application SW* shall evaluate the following conditions in order to prove the *FW* execution terminated correctly. **COLD POWER-ON RESET**

	After a Cold Power-on Reset and LBIST execution the registers and alarms listed in "TC3xx Cold Power-On Reset After LBIST Execution" tables in Appendix A shall be read and their values compared with those shown in the tables in Appendix A. In addition, the user shall consider potential timing related or stuck-at failures related to the ESR0 pin signal handling e.g. system level application to monitor releasing of ESR0. At the end of FW execution after Cold Power-on Reset, during the FW evaluation by Application SW: • In case all registers and *SMU* alarms report the expected values, the Application SW shall: - clear the content of the registers mentioned in the table (except STMEM3..6 and LCLCON0.1), - clear the SMU alarms SMU_AG0..11, - clear the corresponding reset status bits in RSTSTAT register. - proceed further. • If any of the registers do not report the expected values, it shall assumed that the FW has not been executed as expected. In this case an appropriate reaction at system level shall be taken. It is recommended to perform a second attempt within an application-dependent time by triggering a new Cold Power-on Reset. If a mismatch is detected at the second attempt, it shall be assumed that the FW is affected by a permanent fault and the device shall be considered faulty. **STANDBY MODE EXIT** At the exit of standby mode, the Application SW shall perform the same checks as listed in the "TC3xx Cold Power-On Reset After LBIST Execution" tables in Appendix A , considering the exceptions listed in the Notes.
Description (Warm Power-on Reset	**WARM POWER-ON RESET** After a Warm Power-on Reset the registers and alarms listed in "TC3xx Warm Power-On Reset" tables in Appendix A shall be read and their values compared with those shown in these tables. In addition, the user shall consider potential timing related or stuck-at failures related to the ESR0 pin signal handling e.g. system level application to monitor releasing of ESR0. At the end of FW execution after Warm Power-on Reset, during the FW evaluation by Application SW:

	• In case all registers and SMU alarms report the expected values, the Application SW shall: - clear the content of the registers mentioned in the table (except STMEM3..6 and LCLCON0.1), - clear the SMU alarms SMU_AG0..11, - clear the corresponding reset status bits in RSTSTAT register. - proceed further. • If any of the registers do not report the expected values, it shall assumed that the FW has not been executed as expected. In this case an appropriate reaction at system level shall be taken. It is recommended to perform a second attempt within an application-dependent time by triggering a new Warm Power-on Reset. If a mismatch is detected at the second attempt, it shall be assumed that the FW is affected by a permanent fault and the device shall be considered faulty.
Description (System Reset)	**SYSTEM RESET** After a System Reset the registers and alarms listed in "TC3xx System Reset" tables in Appendix A shall be read and their values compared with those shown in these tables. At the end of FW execution after System Reset, during the FW evaluation by Application SW: • In case all registers and SMU alarms report the expected values, the Application SW shall: - clear the content of the registers mentioned in the table (except STMEM3..6 and LCLCON0.1), - clear the SMU alarms SMU_AG0..11, - clear the corresponding reset status bits in RSTSTAT register. - proceed further. • If any of the registers do not report the expected values, it shall assumed that the FW has not been executed as expected. In this case an appropriate reaction at system level shall be taken. It is recommended to perform a second attempt within an application-dependent time by triggering a new System Reset. If a mismatch is detected at the second attempt, it shall be assumed that the FW is affected by a permanent fault and the device shall be considered faulty.
Description (Application Reset)	**APPLICATION RESET**

	After an Application Reset the registers and alarms listed in "TC3xx Application Reset" tables in Appendix A shall be read and their values compared with those shown in these tables. At the end of FW execution after an Application Reset, during the FW evaluation by Application SW: • In case all registers and SMU alarms report the expected values, the Application SW shall: - clear the content of the registers mentioned in the table (except STMEM3..6 and LCLCON0.1), - clear the SMU alarms SMU_AG0..11, - clear the corresponding reset status bits in RSTSTAT register. - proceed further. • If any of the registers do not report the expected values, it shall assumed that the FW has not been executed as expected. In this case an appropriate reaction at system level shall be taken. It is recommended to perform a second attempt within an application-dependent time by triggering a new Application Reset. If a mismatch is detected at the second attempt, it shall be assumed that the FW is affected by a permanent fault and the device shall be considered faulty.
Notes	Notifications in FAULTSTS or ERRINFO registers set during runtime shall be cleared before triggering a new reset (except Cold Power-on Reset).
References	Aurix TC3xx User Manual v1.0.0, Chapter 3.1.2.3 Device Appendix
Recommendations	

5.95 ESM[SW]:SYS:MCU_STARTUP

Description	Before entering the run mode, the *Application SW* shall monitor the corruption of data stored in the following safety relevant registers.
Notes	Safety-relevant registers are to be checked are: • DMU_HP_PROCONPpx (p,x=0..5) • DMU_HF_PROCONPF.RPRO • DMU_HF_PROCONUSR • DMU_HF_PROCONDF • DMU_HF_PROCONRAM • DMU_HF_PROCONDBG (if used) • DMU_SP_PROCONHSM (if used) • DMU_SF_PROCONUSR • DMU_SP_PROCONHSMCBS (if used) • DMU_SP_PROCONHSMCX0/1 (if used)

	• DMU_SP_PROCONHSMCOTP0/1 (if used) • DMU_SP_PROCONHSMCFG (if used) • DMU_HP_PROCONOTPpx (p,x=0..5) • DMU_HP_PROCONWOPpx (p,x=0..5) • DMU_HF_PROCONTP • RIFx_LVDSCON1.RTERM (x=0..1) • STSTAT.HWCFG TC33x\TC32x devices in package variants QFP80 and QFP100 alternatively must use PMSWSTAT.HWCFGEVR for safety check and must not rely on STSTAT.HWCFG Additionally, the user is responsible for selecting all relevant configuration registers for his application and verify their content before entering the run mode.
References	Aurix TC3xx User Manual v1.0.0, Chapter 6.8 Aurix TC3xx User Manual v1.0.0, Chapter 6.5.3.2 Aurix TC3xx User Manual v1.0.0, Chapter 9.1.5.3 Aurix TC3xx User Manual v1.0.0, Chapter 11.2.1.1
Recommendations	The user shall store a data checksum accumulated over all safety relevant registers and application dependent safety registers. The checksum shall be stored in *NVM*. The calculation of the checksum should be of a suitably robust implementation (e.g. IEEE 802.3 standard or AUTOSAR CRC32P4 standard). When the MCU starts up, the Application SW shall first calculate a data checksum for all the data stored in the safety relevant registers and application dependent safety registers. On completion of the calculation the Application SW shall compare the calculated data checksum with the expected data checksum stored in NVM. If the Application SW matches the calculated data checksum and the expected data checksum, the *MCU* has loaded the configuration data successfully. If the Application SW does not match the calculated data checksum and the expected data checksum, the Application SW has detected a corruption of the configuration data and must take an application dependent reaction.

5.96 ESM[SW]:SYS:SW_SUPERVISION

Description	Due to the hardware fault of the microcontroller as well as software faults, a defective program sequence which can lead to the violation of Safety Goal may exist. In order to detect wrong sequences or timing violations of the running programs, the *Application SW* shall implement Program Sequence Monitoring mechanisms for continuous supervision. The MCU WDTs can be used to implement time-window monitoring. In this case, the method can be as follows: • The Application SW shall enable the CPUx WDT by configuring WDTxSR.DS = 0 • The Application SW shall service the CPUx WDT at least once per FTTI
Notes	Upon detection of wrong sequence or timing violation, system integrator is responsible to decide the reaction.

References	ISO 26262 - Part5 Annex.D Table.D.10
Recommendations	According to ISO 26262(2011) Part5 Annex.D Table.D.10, combination of temporal and logical monitoring of software elements are recommended to achieve high diagnostic coverage. ESM[SW]:SYS:SW_SUPERVISION implementation described above only how to targets D2.9.2. In order to achieve D2.9.3, D2.9.4, or D2.9.5 additional software measures will be required.

5.97 ESM[SW]:TRACE.TRAM:REG_MONITOR_TEST

Description	*Application SW* shall execute the REGISTER MONITOR test by starting MCi_ECCS.SFFD. Once started, Application SW shall check that REGISTER_MONITOR test execution time does not exceed expected value. At the end of the test, Application SW shall check self-test results by reading MCi_FAULTSTS.MISCERR.
Notes	Hardware latency for the execution of this test is given by 20*T_spb + 10*T_domain (where Tspb is the *SPB* clock period and T_domain is the clock period which supplies the functional block).
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	Test execution time can be measured by polling MCi_ECCS.SFFD==0. The worst case execution time is highly dependant on the system integrator's code implementation. It is recommended to use 10us as worst case execution time. This recommended value is based on measurements made 2 different code implementations.
Related safety mechanism	***SM[HW]:TRACE.TRAM:REG_MONITOR_TEST***

5.98 ESM[SW]:VMT:MBIST

Description	*MBIST* shall be executed at least once per driving cycle on all SRAM that are used by *Application SW*. The MBIST result shall be read by Application SW in MCx_ECCD and MCx_ETRR registers.
Notes	During the MBIST the SRAM under test can not be accessed, so Application SW shall prevent any attempt of R/W operations on the memory.
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.8
Recommendations	
Related safety mechanism	***SM[HW]:VMT:MBIST***

5.99 SMC[SW]:CLOCK:OSC_MONITOR

Description	To adapt the clock monitors to the chosen *MCU* oscillator frequencies, the *Application SW* shall apply the following settings: Watchdog for Crystal Oscillator:

	1. Deactivate Monitor Alarm in SMU (if necessary) **2.** Define oscillator frequency for monitor (configure OSCCON.OSCVAL) **3.** Reset Monitor (OSCCON.OSCRES= 1) **4.** Activate Monitor Alarm in SMU Watchdog for Backup Oscillator: **1.** Set lower & upper threshold according to f_pll0 frequency (configure CCUCON4.LOTHR and CCUCON4.UPTHR) **2.** Activate Backup clock monitor (CCUCON4.MONEN = 1)
Notes	
References	Aurix TC3xx User Manual v1.0.0, chapter 10.3.1.5
Recommendations	
Related safety mechanism	***SM[HW]:CLOCK:OSC_MONITOR***

5.100 SMC[SW]:DTS:DTS_CFG

Description	The *Application SW* shall configure die temperature limits for the upper and lower thresholds in all *DTS* instances.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.1.4 Device Datasheet 1.0, "Temperature sensor" chapter
Recommendations	
Related safety mechanism	***SM[HW]:DTS:TEMPERATURE_MONITOR***

5.101 SMC[SW]:FCE:CRC_CFG

Description	Before using the *FCE* as a Safety Mechanism, the *Application SW* shall: • configure the FCE channel (incl. type of CRC, interrupt configuration, range of data,...). • shall configure an expected result. • shall initiate the FCE calculation.
Notes	During runtime, if the automatic CRC checksum comparision is not enabled, then the application SW shall compare the CRC value calculated by SM[HW]:FCE:CRC with the expected CRC value. In,case of mismatch, then the corresponding data stream is considered as incorrect.
References	Aurix TC3xx User Manual v1.0.0, Chapter 17.3.1
Recommendations	
Related safety mechanism	***SM[HW]:FCE:CRC***

5.102 SMC[SW]:GTM:GTM_CONFIG_FOR_ATOM

Description	The *Application SW* shall configure the *GTM* to remove ATOM channels out of the round robin access scheme of the Advanced Routing Unit of GTM.
Notes	
References	
Recommendations	
Related safety mechanism	***SM[HW]:GTM:TOM_CCU6_MONITORING_WITH_IOM*** ***SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM***

5.103 SMC[SW]:GTM:GTM_CONFIG_FOR_GTM

Description	*Application SW* shall configure GTM to use two independent TIM in parallel and compare resulting events for plausibility.
Notes	
References	
Recommendations	

5.104 SMC[SW]:GTM:IOM_CONFIG_FOR_GTM

Description	*Application SW* shall configure IOM to feedback GTM:TX to IOM and analyze PWM by independent safety mechanism.
Notes	
References	
Recommendations	
Related safety mechanism	***SM[HW]:GTM:TOM_CCU6_MONITORING_WITH_IOM*** ***SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM***

5.105 SMC[SW]:IR:FFI_CONTROL

Description	The *Application SW* shall configure *IR* Broadcast, Configuration and Control access protection registers: • SRB - Service Request Broadcast Registers (protection via ACCEN_SRBx0, ACCEN_SRBx1) • SRC[31:16] - Service Request Control Register Control information (protection via ACCEN_SRC_TOSx0, ACCEN_SRC_TOSx1) • SRC[15:0] - Service Request Control Register Configuration information (protection via ACCEN_CONFIG0 , ACCEN_CONFIG1) • ECR - Error Capture Registers (protection via ACCEN_CONFIG0 , ACCEN_CONFIG1) For each Broadcast register the IR shall provide a unique write access protection that provides software with the ability to restrict write access to predefined master agent functions.

	For each Interrupt Service Provider, the IR shall provide a unique write access protection, that provides software with the ability to restrict write access to predefined master agent functions, to the *SRN* control information of the Group of SRN that are mapped via SRN configuration to that *ISP*. The IR shall provide software with the ability to restrict write access to SRN configuration information to predefined master agent functions.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 16.8
Recommendations	
Related safety mechanism	***SM[HW]:IR:FFI_CONTROL***

5.106 SMC[SW]:MCU:LBIST_CFG

Description	The *Application SW* shall configure LBIST parameters via LBISTCTRL0, LBISTCTRL1, LBISTCTRL2 and start LBIST execution according to the description provided in the User Manual. After cold power-on reset *FW* can also configure LBIST parameters via UCB registers and start LBIST execution according to the User Manual description (Configuration A).
Notes	It is assumed that LBIST configuration A is used by Application SW.
References	Aurix TC3xx User Manual v1.0.0, Chapter 9.3.3 Device Appendix
Recommendations	
Related safety mechanism	***SM[HW]:MCU:LBIST***

5.107 SMC[SW]:PMS:MONBIST_CFG

Description	The *Application SW* shall configure MONBIST parameters and start MONBIST execution according to the User Manual description.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.2.5.4
Recommendations	
Related safety mechanism	***SM[HW]:PMS:MONBIST***

5.108 SMC[SW]:PMS:MON_REDUNDANCY_CFG

Description	The *Application SW* shall configure threshold parameters for each Vx_MONITOR (Primary voltage monitors) in HSMOVMON, HSMUVMON registers.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.2.5
Recommendations	
Related safety mechanism	***SM[HW]:PMS:MON_REDUNDANCY***

5.109 SMC[SW]:PMS:VX_MONITOR_CFG

Description	The *Application SW* shall configure: -Threshold parameters for each Vx_MONITOR in EVROVMON, EVROVMON2, EVRUVMON, EVRUVMON2 registers, -Enable over- and undervoltage monitoring and direction of voltage crossing in the EVRMONCTRL register, in accordance with application requirements and user manual.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 11.2.2.5.2
Recommendations	
Related safety mechanism	***SM[HW]:PMS:VDDP3_MONITOR*** ***SM[HW]:PMS:VDDPD_MONITOR*** ***SM[HW]:PMS:VDD_MONITOR*** ***SM[HW]:PMS:VEVRSB_MONITOR*** ***SM[HW]:PMS:VEXT_MONITOR***

5.110 SMC[SW]:RIF:CFG_MONITOR

Description	After writing the initial RIF configuration, the *Application SW* shall enable the *RIF* CRC register check (configure RIF_FLM.REGCRCEN = 10_B). When updating the RIF configuration, the Application SW shall • stop the register CRC (configure RIF_FLM.REGCRCEN = 01_B), • update the configuration of the RIF, • write new CRC value in the CRC register, • enable the register CRC engine again, (configure RIF_FLM.REGCRCEN = 10_B), which restarts it from the initial state. The application software has to write the RIF_FLM.REGCRCEN bit field at the end of the configuration update. The CRC calculated by the software has to assume that this bit field has a value of “10B = enabled”.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 22.3.15
Recommendations	
Related safety mechanism	***SM[HW]:RIF:CFG_MONITOR***

5.111 SMC[SW]:RIF:CONTROL_MONITOR

Description	The *Application SW* shall enable the internal *RIF* lockstep (configure RIF_SFCON.LOCKIEN = 10_B).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 22.3.15

Recommendations	
Related safety mechanism	***SM[HW]:RIF:CONTROL_MONITOR***

5.112 SMC[SW]:RIF:ERROR_HANDLING

Description	The *Application SW* shall enable the *RIF* to perform a *CRC* integrity check of radar data (configure RIF_FLM.CRCEN = 1_B). The Application SW shall enable the CRC error interrupt service requests (for n = 0-3, configure RIF_INTCON.CRCEn = 1_B).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 22.3.15
Recommendations	
Related safety mechanism	***SM[HW]:RIF:ERROR_HANDLING***

5.113 SMC[SW]:RIF:SPU_INTERFACE

Description	The *Application SW* shall enable the checking of *SPU* interface CRC checksums (configure RIF_SFCON.SPUCRCEN = 10).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 22.3.8.2
Recommendations	
Related safety mechanism	***SM[HW]:RIF:SPU_INTERFACE***

5.114 SMC[SW]:SCU:ERU_CONFIG

Description	The Application Software shall configure *ERU* parameters and start external request processing by the ERU according to User Manual description.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 9.5
Recommendations	
Related safety mechanism	***SM[HW]:SCU.ERU:EXT_ALARM_GENERATION***

5.115 SMC[SW]:SMU:CONFIG

Description	*Application SW* shall configure the *SMU* in Core domain and SMU in Standby domain so that their can assert configured reaction in response of incoming Alarm signals.
Notes	
References	Aurix TC3xx User Manual v1.0.0, chapter 15.3.1.5 and chapter 15.3.2.4
Recommendations	
Related safety mechanism	***SM[HW]:SMU:ALIVE_MONITOR***

	SM[HW]:SMU:CCF_MONITOR

5.116 SMC[SW]:SPU:BYPASS_CRC

Description	The *Application SW* shall enable the Bypass data CRC check (SPU_SMCTRL.BPCRC = 10_B).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.4
Recommendations	
Related safety mechanism	***SM[HW]:SPU:BYPASS_CRC***

5.117 SMC[SW]:SPU:CONTROL_FLOW_SIGNATURE

Description	The *Application SW* shall (before starting a processing operation with **1.** initialise the SPU to a known state by performing a kernel reset **a.** Write 1_B to SPU_KRST0.RST **b.** Write 1_B to SPU_KRST1.RST **c.** Poll SPU_KRST0.RSTSTAT until it reads 1_B **d.** Write 1_B to KRSTCLR.CLR **2.** enable the computation of a control flow signature (SPU_SMCTRL.CTRLCRCEN = 10_B).
Notes	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.3
Recommendations	
Related safety mechanism	***SM[HW]:SPU:CONTROL_FLOW_SIGNATURE***

5.118 SMC[SW]:SPU:EMEM_TILE

Description	The *Application SW* shall enable *SPU* access address errors (SPU_SMCTRL.RMTAERR = 10_B).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.6
Recommendations	
Related safety mechanism	***SM[HW]:SPU:EMEM_TILE***

5.119 SMC[SW]:SPU:INIT_CONFIGURATION

Description	The *Application SW* shall store the expected *SPU* configuration checksum in SPU_REGCRC.CRC.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.2
Recommendations	

Related safety mechanism	***SM[HW]:SPU:ACTIVE_CFG_CHECK***

5.120 SMC[SW]:SPU:REDUNDANCY

Description	The *Application SW* shall configure the SPU for redundant operation. Redundant Operation: • The Application SW shall enable all lockstep comparators (SPU_CTRL.LSENx = 10_B). • The Application SW shall disable SPU1 errors (SPU_CTRL.ERRDIS = 10_B). • The Application SW shall mirror SPU0 EMEM read data to SPU1 (SPU_CTRL.MS = 10_B). Partial Redundant Operation: • The Application SW shall enable the lockstep comparators 0 and 1 (for x = 0-1, SPU_CTRL.LSENx = 10_B). • The Application SW shall disable the lockstep comparators 2 and 3 (for x = 2-3, SPU_CTRL.LSENx = 01_B). • The Application SW shall enable SPU1 errors (SPU_CTRL.ERRDIS = 01_B). • The Application SW shall not mirror SPU0 EMEM read data to SPU1 (SPU_CTRL.MS = 01_B). No Redundant Operation: • The Application SW shall disable all lockstep comparators (SPU_CTRL.LSENx = 01_B). • The Application SW shall enable SPU1 errors (SPU_CTRL.ERRDIS = 01_B). • The Application SW shall not mirror SPU0 EMEM read data to SPU1 (SPU_CTRL.MS = 01_B).
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 19.6.1.5
Recommendations	
Related safety mechanism	***SM[HW]:SPU:PARTIAL_REDUNDANCY*** ***SM[HW]:SPU:REDUNDANCY***

5.121 SMC[SW]:VMT:MBIST

Description	*Application SW* shall configure the *MTU* for performing *MBIST* by setting MCx_CONFIG0, MCx_CONFIG1 and MCx_MCONTROL registers, where x identifies the SRAM instance.
Notes	
References	Aurix TC3xx User Manual v1.0.0, Chapter 13.3.5.1
Recommendations	
Related safety mechanism	***SM[HW]:VMT:MBIST***

6 Safety Mechanisms

6.1 SM[HW]:AGBT:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.2 SM[HW]:AGBT:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.3 SM[HW]:AMU.LMU_DAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.4 SM[HW]:AMU.LMU_DAM:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.5 SM[HW]:AMU.LMU_DAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.6 SM[HW]:AMU.LMU_DAM:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.7 SM[HW]:AMU.LMU_DAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[0] - Single bit error correction
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.8 SM[HW]:AMU.LMU_DAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.9 SM[HW]:AMU.LMU_DAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.10 SM[HW]:AMU.LMU_DAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:AMU.LMU_DAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.11 SM[HW]:AMU.LMU_DAM:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.12 SM[HW]:AMU.LMU_DAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[2] - Miscellaneous error detection

6.13 SM[HW]:AMU.LMU_DAM:SRI_ADDRESS_MAP_MONITORING

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.14 SM[HW]:AMU.LMU_DAM:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[0] - (LMU_DAM) SRI Address Phase Error
SMU Alarm	ALM11[1] - (LMU_DAM) SRI Write Data Phase Error

6.15 SM[HW]:AMU.LMU_DAM:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.16 SM[HW]:AMU.LMU_DAM:VM_AS_AP

Description

If a master access the memory via the SRI-Bus or the local CPU access the memory, the safety mechanism checks if an individual master tag identifier is enabled to access a region of volatile memory. In case the individual master tag is disabled for accesses an alarm will be raised.

Init Conditions

Runtime conditions

ESM[SW]:CPU:BUS_MPU_INITCHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[15] - MPU violation

6.17 SM[HW]:ASCLIN:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.18 SM[HW]:ASCLIN:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.19 SM[HW]:ASCLIN:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.20 SM[HW]:ASCLIN:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.21 SM[HW]:CCU6:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.22 SM[HW]:CCU6:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.23 SM[HW]:CCU6:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.24 SM[HW]:CCU6:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.25 SM[HW]:CIF.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.26 SM[HW]:CIF.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.27 SM[HW]:CIF.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.
- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.28 SM[HW]:CIF.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[6] - Single bit error correction

6.29 SM[HW]:CIF.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.30 SM[HW]:CIF.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.31 SM[HW]:CIF.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CIF.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.32 SM[HW]:CIF.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[8] - Miscellaneous error detection

6.33 SM[HW]:CIF:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.34 SM[HW]:CIF:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.35 SM[HW]:CIF:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.36 SM[HW]:CIF:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.37 SM[HW]:CLOCK:ALIVE_MONITOR

Description

Monitor the activity of the following clocks

Clock alive monitor for fPLL0, fPLL1, fPLL2:
- reference signal generates a measurement window of 512 cycles of fBACK -> ~5,12µs
- input signal divided by 60
- alarm if frequency of input signal is below ~5,8MHz

Clock alive monitor for fSPB (only visible to HSM):
- reference signal generates a measurement window of 512 cycles of fBACK -> ~5,12µs
- input signal divided by 10
- alarm if frequency of input signal is below ~0,9MHz

Clock alive monitor for fBACK (depending on fPLL0 configuration):
- reference signal generates a measurement window of 512 cycles of fPLL0 -> ~1,7µs
- input signal divided by 60
- alarm if frequency of input signal is below ~17MHz

Note: In this case, upon apperence of an infrastructure faults, the system integrator is responsible to decide on the appropriate reaction.
As a recommendation:
SMU_stdby hardware alarm path described in ***SM[HW]:CLOCK:ALIVE_MONITOR*** can be replaced by Software Safety Mechanisms to detect wrong peripheral operation (in case of fPLL1, fPLL2 failure) and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** (in case of fPLL0 failure).

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[2] - Back-up clock alive alarm
SMU Alarm	ALM21[15] - PLLx/fSPB Alive Alarm (provided on fBACK clock with x = 0..2)

6.38 SM[HW]:CLOCK:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error

6.39 SM[HW]:CLOCK:CFG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

ESM[SW]:CLOCK:PLAUSIBILITY

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[4] - Safety flip-flop uncorrectable error detected
SMU Alarm	ALM6[5] - Safety flip-flop uncorrectable error detected
SMU Alarm	ALM6[8] - PLL Safety flip-flop uncorrectable error detected
SMU Alarm	ALM6[24] - Safety flip-flop correctable error detected
SMU Alarm	ALM6[25] - Safety flip-flop uncorrectable error detected
SMU Alarm	ALM10[20] - CCU Safety flip-flop correctable error detected.
SMU Alarm	ALM10[21] - Miscellaneous Safety flip-flop uncorrectable error detected
SMU Alarm	ALM21[7] - Safety flip-flop uncorrectable error detected

6.40 SM[HW]:CLOCK:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[4] - SCU Safety FF uncorrectable error detected
SMU Alarm	ALM6[8] : SYS_PLL/PER_PLL SFF uncorrectable error detected

6.41 SM[HW]:CLOCK:OSC_MONITOR

Description

The Watchdog function monitors the frequency of the PLL inputs. - Backup Clock (fBACK) - XTAL Oscillator (fOSC)

The backup clock (fBACK) monitor simply counts the number of backup clock cycles within a window of 512 f_{PLL0} clock cycles. If the number of counted cycles exceeds a configurable window size, an alarm is generated. This check is continuously repeated. The configured window is used to adapt to the selected frequency of the monitoring clock, i.e. f_{PLL0}.

The oscillator watchdog (OWD) is used to monitor the frequency of the f_{OSC} clock, by comparing it to the frequency of the internal oscillator f_{BACK}.

<u>Frequency too high:</u>

fosc_high=f_{osc}/(1-0.3)*(1+0.3)+3/Window_size, where window_size = Ncounts*Ndiv/(f_{BACK}*(1-0.3)), Ncounts=25, Ndiv=40, f_{BACK}=100 MHz

<u>Frequency too low:</u>

fosc_low=f_{osc}/(1+0.3)*(1-0.3)-3/Window_size, where window_size = Ncounts*Ndiv/(f_{BACK}*(1+0.3)), Ncounts=25, Ndiv=40, f_{BACK}=100 MHz

Init Conditions

SMC[SW]:CLOCK:OSC_MONITOR

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[0] - OSC clock frequency out of range
SMU Alarm	ALM8[1] - Back-up clock out-of-range alarm

6.42 SM[HW]:CLOCK:PLL_GLITCH_FILTER

Description

The PLL has a DCO which provides a minimum low and high phase for each clock cycle, so PLL intrinsic no glitches can propagate from PLL input to PLL output.

After PLL output further divider and multiplexors are in each clock path. Synchronous digital design is clocked on positive edge, duty cycle has very big margin.

Glitches can be caused by unintended asynchronous switching between clocks. Changes in these settings are there for applied with a special “glitch free multiplexer” design.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N/A	N.A.

6.43 SM[HW]:CLOCK:PLL_LOSS_OF_LOCK_DETECTION

Description

The PLL has a lock detection that supervises the VCO part of the PLL in order to differentiate between stable and unstable VCO circuit behavior. The PLL may become unlocked, caused by a break of the crystal/ceramic resonator or the external clock line. In case of loss-of-lock, the Clock Control Unit (CCU) switches to the back-up clock, this is called the Clock Emergency Behavior.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[3] - System PLL DCO loss of lock event
SMU Alarm	ALM8[4] - Peripheral PLL DCO loss of lock event

6.44 SM[HW]:CLOCK:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.45 SM[HW]:CONVCTRL:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.46 SM[HW]:CONVCTRL:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.47 SM[HW]:CONVCTRL:PHASE_SYNC_ERR

Description

The Safety Mechanism supervises the operation of the Phase Synchronizer by monitoring the parity bit of its prescaler(PHSCFG.PHSDIV) and the counter value.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[12] - Phase Synchronizer Error

6.48 SM[HW]:CONVCTRL:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.49 SM[HW]:CONVCTRL:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.50 SM[HW]:CPU.DCACHE:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.51 SM[HW]:CPU.DCACHE:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.52 SM[HW]:CPU.DCACHE:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[9] - (x=0..5) Single bit error correction
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.53 SM[HW]:CPU.DCACHE:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.54 SM[HW]:CPU.DCACHE:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.55 SM[HW]:CPU.DCACHE:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.DCACHE:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.56 SM[HW]:CPU.DCACHE:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[11] - (x=0..5) Miscellaneous error detection

6.57 SM[HW]:CPU.DLMU:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.58 SM[HW]:CPU.DLMU:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.59 SM[HW]:CPU.DLMU:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[9] - (x=0..5) Single bit error correction
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.60 SM[HW]:CPU.DLMU:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.61 SM[HW]:CPU.DLMU:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.62 SM[HW]:CPU.DLMU:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.DLMU:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.63 SM[HW]:CPU.DLMU:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[11] - (x=0..5) Miscellaneous error detection

6.64 SM[HW]:CPU.DLMU:VM_AS_AP

Description

If a master access the memory via the SRI-Bus or the local CPU access the memory, the safety mechanism checks if an individual master tag identifier is enabled to access a region of volatile memory. In case the individual master tag is disabled for accesses an alarm will be raised.

Init Conditions

Runtime conditions

ESM[SW]:CPU:BUS_MPU_INITCHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[1] - CPUx (x=0..5) Access Protection violation

6.65 SM[HW]:CPU.DSPR:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.66 SM[HW]:CPU.DSPR:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.67 SM[HW]:CPU.DSPR:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[9] - (x=0..5) Single bit error correction
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.68 SM[HW]:CPU.DSPR:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.69 SM[HW]:CPU.DSPR:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[10] - (x=0..5) Uncorrectable critical error detection

6.70 SM[HW]:CPU.DSPR:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.DSPR:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.71 SM[HW]:CPU.DSPR:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[11] - (x=0..5) Miscellaneous error detection

6.72 SM[HW]:CPU.DSPR:VM_AS_AP

Description

If a master access the memory via the SRI-Bus or the local CPU access the memory, the safety mechanism checks if an individual master tag identifier is enabled to access a region of volatile memory. This check does not cover load or store operations of the local CPU. In case the individual master tag is disabled for accesses an alarm will be raised.

Init Conditions

Runtime conditions

ESM[SW]:CPU:BUS_MPU_INITCHECK

Tests

Fault identification interfaces

VD Assignment	VA_Rationale
SMU Alarm	ALMx[1] - CPUx (x=0..5) Access Protection violation

6.73 SM[HW]:CPU.DTAG:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[13] - (x=0..5) Uncorrectable critical error detection

6.74 SM[HW]:CPU.DTAG:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[13] - (x=0..5) Uncorrectable critical error detection

6.75 SM[HW]:CPU.DTAG:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU. [ALL0] and [ALL1] errors can be detected. In case [ALL0] is detected a critical uncorrectable error alarm is forwarded to the SMU. In case [ALL1] is detected a critical uncorrectable error alarm as well as a correctable error alarm are forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[12] - (x=0..5) Single bit error correction
SMU Alarm	ALMx[13] - (x=0..5) Uncorrectable critical error detection

6.76 SM[HW]:CPU.DTAG:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[13] - (x=0..5) Uncorrectable critical error detection

6.77 SM[HW]:CPU.DTAG:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[13] - (x=0..5) Uncorrectable critical error detection

6.78 SM[HW]:CPU.DTAG:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.DTAG:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.79 SM[HW]:CPU.DTAG:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[14] - (x=0..5) DTAG Miscellaneous error detection

6.80 SM[HW]:CPU.PCACHE:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.81 SM[HW]:CPU.PCACHE:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.82 SM[HW]:CPU.PCACHE:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[6] - (x=0..5) Single bit error correction
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.83 SM[HW]:CPU.PCACHE:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.84 SM[HW]:CPU.PCACHE:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.85 SM[HW]:CPU.PCACHE:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.PCACHE:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.86 SM[HW]:CPU.PCACHE:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[8] - (x=0..5) Miscellaneous error detection

6.87 SM[HW]:CPU.PSPR:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.88 SM[HW]:CPU.PSPR:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.89 SM[HW]:CPU.PSPR:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[6] - (x=0..5) Single bit error correction
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.90 SM[HW]:CPU.PSPR:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.91 SM[HW]:CPU.PSPR:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[7] - (x=0..5) Uncorrectable critical error detection

6.92 SM[HW]:CPU.PSPR:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.PSPR:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.93 SM[HW]:CPU.PSPR:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[8] - (x=0..5) Miscellaneous error detection

6.94 SM[HW]:CPU.PSPR:VM_AS_AP

Description

If a master access the memory via the SRI-Bus or the local CPU access the memory, the safety mechanism checks if an individual master tag identifier is enabled to access a region of volatile memory. This check does not include fetch operations by the local CPU. In case the individual master tag is disabled for accesses an alarm will be raised.

Init Conditions

Runtime conditions

ESM[SW]:CPU:BUS_MPU_INITCHECK

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[1] - CPUx (x=0..5) Access Protection violation

6.95 SM[HW]:CPU.PTAG:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[4] - (x=0..5) Uncorrectable critical error detection

6.96 SM[HW]:CPU.PTAG:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[4] - (x=0..5) Uncorrectable critical error detection

6.97 SM[HW]:CPU.PTAG:ERROR_DETECTION

Description

The SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[4] - (x=0..5) Uncorrectable critical error detection

6.98 SM[HW]:CPU.PTAG:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[4] - (x=0..5) Uncorrectable critical error detection

6.99 SM[HW]:CPU.PTAG:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[4] - (x=0..5) Uncorrectable critical error detection

6.100 SM[HW]:CPU.PTAG:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:CPU.PTAG:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.101 SM[HW]:CPU.PTAG:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALMx[5] - (x=0..5) Miscellaneous error detection

6.102 SM[HW]:CPU:CFG_AS_AP

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Description

The safety mechanism checks if an individual master tag identifier is enabled to access a CPU CSFR or SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[1] - CPUx (x=0..5) Access Protection violation
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.103 SM[HW]:CPU:CODE_MPU

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Description

The safety mechanism protects user-specified memory ranges from unauthorized fetch accesses by the software running on the CPU. A protection range is a contiguous part of the address space defined by a lower and upper address boundary for which access permissions are specified. A protection set is a complete set of protection region registers. The safety mechanism implements 6 protection sets. Each protection implements 10 protection ranges. The protection sets define instruction fetch permissions. The safety mechanism manages the changing of the protection set on a task context switch. Each task is assigned code execution ranges, for application code and shared code (libraries).

Init Conditions

Runtime conditions

ESM[SW]:CPU:CODE_MPU_CHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
CPU Trap Class 1	MPX, TIN 4

6.104 SM[HW]:CPU:CONTROL_REDUNDANCY

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description

A lockstep detects permanent and transient failures of the CPU:PFLASH read path by the way of hardware redundancy, using two independent CPU:PFLASH read path instances called the master CPU:PFLASH read path and the checker CPU:PFLASH read path. The two CPU:PFLASH read path instances operate in a lockstep manner: They use the same input data, execute the same operation, and all the functional outputs of the master CPU:PFLASH read path and the checker CPU:PFLASH read path are compared on a cycle by cycle basis using a hardware comparator. Alarm generation from CPU:PFLASH lockstep is combined with CPU:TRICORE lockstep.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[0] (CPUx) - (x=0..5) Lockstep Comparator Error ALMx[2] (CPUx) - (x is Non Lockstep CPU) PFLASH Read Path Monitor

6.105 SM[HW]:CPU:CONTROL_REDUNDANCY_SCC

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	Yes	Yes
Freedom from Interference	N.A.	N.A.

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[0] (CPUx) - (x=0..5) Lockstep Comparator Error ALMx[2] (CPUx) - (x is Non Lockstep CPU) PFLASH Read Path Monitor

6.106 SM[HW]:CPU:CRC

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description

The *CPU* data checksum is a cyclic redundancy checksum calculated from data stored in the CPU. The calculation of the CPU data checksum within the CPU shall fulfil the IEEE 802.3 standard. *Application SW* may compare the calculated CPU data checksum with an expected CPU data checksum stored in system memory. If the Application SW fails to match the expected and calculated CPU data checksums, the Application SW has detected a fault in the data.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	

6.107 SM[HW]:CPU:DATA_MPU

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Description

The safety mechanism protects user-specified memory ranges from unauthorized read or write accesses by the software running on the CPU. A protection range is a contiguous part of the address space defined by a lower and upper address boundary for which access permissions are specified. A protection set is a complete set of protection region registers. The safety mechanism implements 6 protection sets. Each protection implements 18 protection ranges. The protection sets define read and write access fetch permissions. The safety mechanism manages changing the protection set on a task context switch. Each task is assigned data ranges for local data, constants (read only), global data and local stack.

Init Conditions

Runtime conditions

ESM[SW]:CPU:DATA_MPU_CHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
CPU Trap Class 1	MPR, TIN 2
CPU Trap Class 1	MPW, TIN 3
CPU Trap Class 1	MPP, TIN 5
CPU Trap Class 1	MPN, TIN 6

6.108 SM[HW]:CPU:ENDINIT

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.

Detection and Control	lockstep CPU	non-lockstep CPU
Freedom from Interference	Yes	Yes

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of CPU CSFRs and SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these CPU CSFRs and SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write a ENDINIT protected CPU CSFR or SFR without deactivating the protection is discarded and the register value is not changed. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.109 SM[HW]:CPU:SAFETY_ENDINIT

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of CPU CSFRs and SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these CPU CSFRs and SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such CPU CSFRs and SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected CPU CSFR or SFR without deactivating the protection is discarded and the register value is not changed. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.110 SM[HW]:CPU:SRI_ADDRESS_MAP_MONITORING

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.111 SM[HW]:CPU:SRI_TRANSACTION_INTEGRITY

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag

SMU Alarm	for x = 0 to 5, ALMx[22] - (CPUx PBUS) SRI Address Phase Error
	for x = 0 to 5, ALMx[23] - (CPUx DBUS) SRI Address Phase Error

SMU Alarm	for x = 0 to 5, ALMx[22] - (CPUx PBUS) SRI Write Data Phase Error
	for x = 0 to 5, ALMx[23] - (CPUx DBUS) SRI Write Data Phase Error

SMU Alarm	for x = 0 to 5, ALMx[22] - (CPUx PBUS) SRI Read Data Phase Error
	for x = 0 to 5, ALMx[23] - (CPUx DBUS) SRI Read Data Phase Error

6.112 SM[HW]:CPU:STARTUP_PROTECTION

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

After start-up, the *SSW* automatically set SCU_STCON.STP=1_B for preventing unintentional changes from *Application SW* of start-up protected *SFR*, which define the *MCU* system characteristic. This is a safety mechanism for failure reduction only and does not report any *SMU* alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.113 SM[HW]:CPU:STI

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

If the CPU is executing a safety task, the CPU may use the Safety Task Identifier (STI) to select an alternative master tag identifier when load/store instruction accesses an MCU resource. In each CPU, the usage of the STI is controlled by the STI flag PSW.S. The Program Status Word (PSW) is part of any task context so that a task switch automatically determines the bus master tag identifier associated with that task. The STI is used in the context of MCU Access Protection (AP).

Note: The STI does not apply to local DSPR but applies to PSPR accesses as loads and stores to PSPR are routed via the SRI and the MCU AP is present.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[20] - SPB Bus Error Event
CPU Trap Class 4	DSE, TIN 2

6.114 SM[HW]:CPU:SV

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

If a task is to perform read and write accesses to system registers and all peripheral modules, the tasks should be configured with CPU privilege level supervisor mode access protection. Tasks configure with supervisor mode may disable interrupts. Supervisor mode is configured by the I/O privilege bit field PSW.IO.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.115 SM[HW]:CPU:SV_AP

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[1] - CPUx (x=0..5) Access Protection violation
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.116 SM[HW]:CPU:SWAP_CONFIGURATION_PROTECTION

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description

The safety mechanism checks for unintentional SWAP configuration change. For an unintentional change an alarm is generated for the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[2] (CPUx) - (x=0..5)

6.117 SM[HW]:CPU:TPS

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

The Temporal Protection System (TPS) guards against task runtime overrun. For each CPU, TPS consists of three independent decrementing counters (TPS_TIMERn). If a counter decrements to zero, the TPS generates a trap. An operating system may use SM[HW]:CPU:TPS to detect budget overruns in the system.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
CPU Trap Class 4	TAE, TIN 7

6.118 SM[HW]:CPU:TPS_EXCEPTION_TIME_MONITOR

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

The CPU exception handling safety mechanism checks that an exception request is taken by an exception handler. An incoming exception request sets a timer. As soon as the exception request enters the exception handler, the CPU turns off the timer. If the timer reaches the maximum allowable time period, the CPU triggers an alarm. Software may set the maximum allowable time period by configuring the CPU TPS_EXTIM registers.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[24] (CPUx) - (x=0..5) CPU exception (interrupt/trap)

6.119 SM[HW]:CPU:TRICORE_LOCKSTEP

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	N.A.
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Note: CPU0 shall be used in lockstep mode only. All ohters lockstep *CPU* can be configured as non-lockstep if required by the application.

Description

A lockstep detects permanent and transient failures of the TriCore CPU by the way of hardware redundancy, using two independent TriCore CPU instances called the master TriCore CPU and the checker TriCore CPU. The two TriCore CPUs operate in a lockstep manner: They use the same input data, execute the same operation, and all the functional outputs of the master TriCore CPU and the checker TriCore CPU are compared on a cycle by cycle basis using a hardware comparator. The lockstep operation has no effect on the software execution. Additional measures are implemented in hardware to mitigate common cause faults between the redundant TriCore CPU instances.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[0] (CPUx) - (x=0..3) Lockstep Comparator Error
SMU Alarm	ALM8[18] - LockStep DualRail Error

6.120 SM[HW]:CPU:TRICORE_LOCKSTEP_SCC

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	N.A.
Latent Fault Detection	Yes	N.A.
Freedom from Interference	N.A.	N.A.

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[0] (CPUx) - (x=0..3) Lockstep Comparator Error

6.121 SM[HW]:CPU:UM0

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

If a task is not to access peripherals, the task should be configured with CPU privilege level user mode 0 access protection. Tasks configured for user mode 0 cannot enable or disable interrupts.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
CPU Trap Class 1	PRIV, TIN 1
CPU Trap Class 1	MPP, TIN 5

6.122 SM[HW]:CPU:UM1

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	Yes	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	Yes	Yes

Description

If a task is to access common, unprotected peripherals, the task should be configured with CPU privilege level user mode 1 access protection. Typical tasks are a read or write access to serial port, a read access to timer, and most I/O status registers. Tasks configured with user mode 1 access protection may enable and disable interrupts (default behaviour). This ability may be removed if desired by configuration of the system control register SYSCON appropriately.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
CPU Trap Class 1	PRIV, TIN 1

6.123 SM[HW]:DEBUG:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.124 SM[HW]:DEBUG:CFG_MONITOR

Description

The safety mechanism detects a transient fault in global debug enable register which is considered as a common cause initiator. On failure detection the safety mechanism corrects the failure using majority voting scheme.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[23] - Safety FF for global debug enable register uncorrectable error detected

6.125 SM[HW]:DEBUG:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.126 SM[HW]:DEBUG:FFI_CONTROL

Description

Local debug control signals are gated with the global debug enable control signal to prevent unintended debug enable.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	**SM Flag**
No Alarm and no Status	N.A.

6.127 SM[HW]:DEBUG:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	**SM Flag**
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.128 SM[HW]:DEBUG:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.129 SM[HW]:DFX:FFI_CONTROL

Description

DFX-FB controls several signals which allow a direct overruling of basic system functionality (involving Clocks, Reset and Port-logic) for test purpose only. During safe application it has to be ensured that these signals always stay at their safe value to ensure freedom-from-interference. At the source side within the DfX-block it is ensured that all test-related overruling signals remain at their inactive (safe) state as long the chip is not operating in a specific test-mode. Nevertheless due to the fact that the test-related overruling signals are widely spread across the chip-area additional blocking mechanisms are also implemented on the destination side for safety-critical test-overruling signals (failure rate reduction). The blocking is realized through simple AND-gating which involves the test-override signal and combines it with a general test-enable signal. This ensures that the resulting test control override signal always remains at its inactive state as long the chip is not operating in a specific test-mode.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.130 SM[HW]:DFX:FFI_MONITOR

Description

DFX controls several signals meant for production test purpose only which allow a direct overruling of basic system functionality. This mechanism monitors that during runtime test signals are not activated. The mechanism is detecting and signaling via an SMU alarm the following events:

1. Test-Mode Alarm: detects failures which lead MCU to enter Test Mode or enable safety-critical overruling signals. 2. LBIST-Alarm: detects failures which lead MCU to enter LBIST Mode or indicate that the LFM of the central LBIST-controller is not in the safe idle-state (*). 3. Pad-Heating Alarm: Detects failures which lead MCU to enable a high current consumption mode in the supply pads (only used during production test). 4. Clock-Monitoring Alarm: Detects failures which lead MCU to override safety-critical settings in CLOCK-FB (this might result in a disturbance of the system clock).

(*) Note: In case of an erroneous LBIST-entry the chip-internal safety-mechanisms can no longer work, because the system is transferred into a structural scan-test-mode. Therefore the LBIST-alarm function can only successfully indicate fails which bring the LBIST-control logic to an unsafe state **without** starting LBIST-operation. In case LBIST is erroneously started this can only be indicated by an ESM (e.g. ESM[HW]:SYS:WATCHDOG_FUNCTION or ESM[HW]:MCU:LBIST_MONITOR).

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[5] - LBIST Alarm
SMU Alarm	ALM8[21] - LBIST Test Mode Alarm
SMU Alarm	ALM8[20] - Pad Heating Alarm
SMU Alarm	ALM21[15] - Clock Alive Monitor

6.131 SM[HW]:DFX:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.132 SM[HW]:DMA.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.133 SM[HW]:DMA.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.134 SM[HW]:DMA.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM6[19] - Single bit error correction
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.135 SM[HW]:DMA.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.136 SM[HW]:DMA.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.137 SM[HW]:DMA.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:DMA.RAM:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.138 SM[HW]:DMA.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM6[21] - Miscellaneous error detection

6.139 SM[HW]:DMA:ADDRESS_CRC

Description

The *DMA* address checksum is a cyclic redundancy checksum calculated for DMA source addresses and DMA destination addresses. The calculation of the DMA address checksum within the DMA shall fulfil the IEEE 802.3 standard. The DMA address checksum is stored in the DMA channel DMA_SDCRCRc register. On completion of a DMA transaction, ESM[SW]:DMA:ADDRESS_CRC may compare the calculated DMA address checksum with an expected DMA address checksum stored in system memory. If ESM[SW]:DMA:ADDRESS_CRC fails to match the expected and calculated checksums, software has detected a fault on the DMA address generation of the DMA source addresses and DMA destination addresses. Depending on the DMA channel TCS configuration, the DMA address checksum is accumulated over a DMA transaction or over all DMA transactions of a DMA linked list.

SM[HW]:DMA:ADDRESS_CRC is not supported when the DMA channel is configured for Conditional Linked List.

Init Conditions

Runtime conditions

ESM[SW]:DMA:ADDRESS_CRC

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Interrupt Service Request	

6.140 SM[HW]:DMA:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.141 SM[HW]:DMA:DATA_CRC

Description

The *DMA* data checksum is a cyclic redundancy checksum calculated for data moved during a DMA transaction. The calculation of the DMA data checksum within the DMA shall fulfil the IEEE 802.3 standard. The DMA data checksum is stored in the DMA channel DMA_RDCRCRc register. On completion of a DMA transaction, ESM[SW]:DMA:DATA_CRC may compare the calculated DMA data checksum with an expected DMA data checksum stored in system memory. If ESM[SW]:DMA:DATA_CRC fails to match the expected and calculated checksums, software has detected a fault in the DMA move data. Depending on the DMA channel TCS configuration, the DMA data checksum is accumulated over a DMA transaction or over all DMA transactions of a DMA linked list.

Init Conditions

Runtime conditions

ESM[SW]:DMA:DATA_CRC

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Interrupt Service Request	

6.142 SM[HW]:DMA:DATA_EDC

Description

During a DMA move, the DMA move data is protected with ECC. The *DMA* detects permanent and transient faults on internal DMA move data. If the DMA performs a DMA move from an SRI source module to an SRI destination module, the DMA maintains the integrity of the SRI bus protocol as the DMA move data is routed through the DMA. If the DMA performs a DMA read move to an FPI source module, the DMA generates an equivalent SRI data phase checksum to protect the DMA move data as it is routed through the DMA. If the DMA detects a fault, the DMA triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[23] - DMA SRI ECC Error

6.143 SM[HW]:DMA:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.144 SM[HW]:DMA:ERROR_HANDLING

Description

The *DMA* may detect the following types of DMA errors:

- Transaction Request Lost (TRL)
- Source Error (SER)
- Destination Error (DER)
- DMA RAM Error (RAMER)
- DMA Linked List Error (DLLER)
- Safe Linked List Error (SLLER)

If the DMA detects a DMA error, the DMA may trigger an interrupt service request. The DMA stores the number of the DMA channel and the type of DMA error to provide diagnostic information.

Init Conditions

Runtime conditions

ESM[SW]:DMA:ERROR_HANDLING

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Interrupt Service Request	

6.145 SM[HW]:DMA:REQUEST_MONITOR

Description

The *DMA* detects permanent and transient faults in DMA requests. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[6] - Safety FF uncorrectable error detected

6.146 SM[HW]:DMA:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.147 SM[HW]:DMA:SINGLE_TRANSACTION

Description

The *DMA* may be configured to perform a single occurrence of a DMA transaction. If the DMA detects a subsequent DMA hardware request for the same DMA transaction, the DMA signals a transaction request lost error.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	DMA Flag
Interrupt Service Request	DMA channel TSRc.TRL

6.148 SM[HW]:DMA:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU	ALM8[23] - (DMA) SRI Read Data Phase Error

6.149 SM[HW]:DMA:STI

Description

If the *DMA* is servicing a safe DMA request, the DMA may use an alternative master tag identifier when performing DMA read moves or DMA write moves to MCU resources. The DMA provides software with the capability to assign a DMA channel to a resource partition. Each resource partition supports a unique on chip bus master tag identifier which allows the access protection system to distinguish between the resource partitions when performing write accesses to registers or read and writes accesses to volatile memory.

Init Conditions

Runtime conditions

Tests

ESM[SW]:DMA:ERROR_HANDLING

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - XBAR0 bus error event
SMU Alarm	ALM7[19] - XBAR2 bus error event
SMU Alarm	ALM7[20] - SPB bus error event

6.150 SM[HW]:DMA:SV

Description

The *DMA* provides software with the ability to configure a resource partition to perform DMA write moves for User Mode or Supervisor Mode. The *Application SW* may only write to some registers in supervisor mode.

Init Conditions

Runtime conditions

Tests

ESM[SW]:DMA:ERROR_HANDLING

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - XBAR0 bus error event
SMU Alarm	ALM7[19] - XBAR2 bus error event
SMU Alarm	ALM7[20] - SPB bus error event

6.151 SM[HW]:DMA:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.152 SM[HW]:DMA:TIMESTAMP

Description

DMA Timestamp**:** The *DMA* supports the generation of a 32-bit time stamp from a continuously running timer. If the DMA channel TCS is configured to enable the appendage of a DMA timestamp, the DMA performs an additional DMA write move to append the current 32-bit DMA counter value to the destination data sequence.

STM Timestamp: The DMA may be configured to perform a DMA linked list operation to copy a timestamp from the *STM* to the destination. If a DMA channel TCS is configured to execute a DMA linked list operation then on completion of the data movement the DMA performs a DMA transaction to copy the current 64-bit STM counter value to the destination data sequence.

A DMA timestamp or STM timestamp may be used by the application software to detect failures of the DMA function such as "Corruption of Message", "Insertion of Message", "Loss of Message" and "Incorrect Delivery Time". For repetitive data movement operations, the safety mechanism may be used to monitor the freshness of move data.

SM[HW]:DMA:TIMESTAMP is not supported when the DMA channel is configured for DMA Double Source Buffering Software Switch Only, DMA Double Destination Buffering Software Switch Only, Conditional Linked List and Pattern Detection.

Init Conditions

Runtime conditions

ESM[SW]:DMA:TIMESTAMP

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Interrupt Service Request	

6.153 SM[HW]:DTS:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.154 SM[HW]:DTS:TEMPERATURE_MONITOR

Description

Detects whether the DIE temperature is within the specified limits, and sets temperature underflow/overflow alarms.
Note : Both instances of DTS provide independent temperature underflow/overflow alarms.

Init Conditions

SMC[SW]:DTS:DTS_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[31] / ALM9[0] - Temperature overflow
SMU Alarm	ALM8[30] / ALM9[1] - Temperature underflow

6.155 SM[HW]:EBU:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.156 SM[HW]:EBU:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.157 SM[HW]:EBU:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.158 SM[HW]:EBU:SRI_ADDRESS_MAP_MONITORING

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.159 SM[HW]:EBU:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address phase of an SRI transaction, the SRI interface triggers an alarm.

The EBU implementation of this safety mechanism does not indicate errors in the data phase(s) of a write transaction

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.160 SM[HW]:EBU:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.161 SM[HW]:EDSADC:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.162 SM[HW]:EDSADC:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.163 SM[HW]:EDSADC:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.164 SM[HW]:EDSADC:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.165 SM[HW]:EMEM.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[4] - Uncorrectable critical error detection

6.166 SM[HW]:EMEM.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU. In addition the ECC logic is capable of detecting errors in the address decoding and a critical uncorrectable error alarm is forwarded to the SMU during a SRAM read operation.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[3] - Single bit error correction
SMU Alarm	ALM7[4] - Uncorrectable critical error detection

6.167 SM[HW]:EMEM.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[4] - Uncorrectable critical error detection

6.168 SM[HW]:EMEM.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[4] - Uncorrectable critical error detection

6.169 SM[HW]:EMEM.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:EMEM.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.170 SM[HW]:EMEM.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[5] - Miscellaneous error detection

6.171 SM[HW]:EMEM:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[15] - Bus-level MPU error
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.172 SM[HW]:EMEM:CONTROL_REDUNDANCY

Description

EMEM module control redundancy detects permanent and transient failures of the EMEM SRI control logic by the way of hardware redundancy, using two independent EMEM SRI control logic instances called the master EMEM SRI control logic and the checker EMEM SRI control logic. The two EMEM SRI control logic instances operate in a redundant manner: they use the same input data, execute the same operation, and all the functional outputs of the master and the checker are compared on a cycle by cycle basis using a hardware comparator. The redundant operation has no effect on the EMEM resource access time.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[12] - EMEM comparator error
SMU Alarm	ALM7[13] - EMEM control error

6.173 SM[HW]:EMEM:CONTROL_REDUNDANCY_SCC

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[12] - EMEM comparator error

6.174 SM[HW]:EMEM:ECC_MONITOR

Description

EMEM SRAM have a monitor mechanism that controls the *ECC* output. In case the ECC performs a wrong correction upon an *SBE* event, the ECC monitor forwards an alarm to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[16] (Read error)

6.175 SM[HW]:EMEM:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.176 SM[HW]:EMEM:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure is reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[3] - Safety FF uncorrectable error detected

6.177 SM[HW]:EMEM:READ_WRITE_MUX

Description

The *EMEM* RAM may be accessed through a MUX via the following functional interfaces:

- SRI interface: connection to the Software Execution Platform and Internal MCU Communications MCU functions.
- SEP interface: connection to the ADAS Processing MCU function.
- FPI interface: connection to the CIF for legacy camera applications.
- AGBT/MCDS interface: connection to debug system.

If the EMEM detects an unauthorised access to the EMEM RAM, the EMEM triggers an alarm.

Init Conditions

Runtime conditions

ESM[SW]:EMEM:READ_CONFIGURATION

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[20] - unauthorised access to EMEM RAM

6.178 SM[HW]:EMEM:READ_WRITE_SEP

Description

The *SPU* accesses the *EMEM* via the SPU EMEM Protocol (SEP) bus. The EMEM monitors the integrity of SEP read and write accesses to volatile memory. If the EMEM detects an integrity error in the EMEM SEP path, the EMEM triggers an alarm to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM9[21]

6.179 SM[HW]:EMEM:READ_WRITE_SRI

Description

The *CPU* and *DMA* access the *EMEM* via the *SRI*. The EMEM monitors the integrity of SRI read and write accesses to volatile memory. If the EMEM detects an integrity error in the EMEM SRI data path, the EMEM triggers an alarm to the *SMU*.

Init Conditions

Runtime conditions

ESM[SW]:EMEM:READ_CONFIGURATION

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[14]

6.180 SM[HW]:EMEM:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.181 SM[HW]:EMEM:SPU_MONITOR

Description

The *EMEM* shall monitor the *SPU* configuration. If two functional SPU instances are configured in a redundant mode and the EMEM detects a fault in the SPU redundant mode configuration, the EMEM triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM9[22] - SPU configuration error

6.182 SM[HW]:EMEM:SRI_ADDRESS_MAP_MONITORING

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event

6.183 SM[HW]:EMEM:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[0] - (EMEM) SRI Address Phase Error
SMU Alarm	ALM11[1] - (EMEM) SRI Write Data Phase Error

6.184 SM[HW]:EMEM:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.185 SM[HW]:EMEM:VM_AS_AP

Description

If a master access the memory via the SRI-Bus or the local CPU access the memory, the safety mechanism checks if an individual master tag identifier is enabled to access a region of volatile memory. In case the individual master tag is disabled for accesses an alarm will be raised.

Init Conditions

Runtime conditions

ESM[SW]:CPU:BUS_MPU_INITCHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[15] - Bus-level MPU error

6.186 SM[HW]:ERAY.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[14] - Uncorrectable critical error detection

6.187 SM[HW]:ERAY.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[14] - Uncorrectable critical error detection

6.188 SM[HW]:ERAY.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.

- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[14] - Uncorrectable critical error detection

6.189 SM[HW]:ERAY.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[13] - Single bit error correction

6.190 SM[HW]:ERAY.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[14] - Uncorrectable critical error detection

6.191 SM[HW]:ERAY.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[14] - Uncorrectable critical error detection

6.192 SM[HW]:ERAY.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:ERAY.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.193 SM[HW]:ERAY.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[15] - Miscellaneous error detection

6.194 SM[HW]:ERAY:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.195 SM[HW]:ERAY:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.196 SM[HW]:ERAY:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.197 SM[HW]:ERAY:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.198 SM[HW]:EVADC:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.199 SM[HW]:EVADC:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.200 SM[HW]:EVADC:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.201 SM[HW]:EVADC:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.202 SM[HW]:FCE:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.203 SM[HW]:FCE:CRC

Description

The *FCE* is a hardware acceleration engine that implements concurrent CRC computation on the following polynomials:

- IEEE 802.3 ethernet CRC32
- Autosar CRC32P4
- CCITT CRC16
- SAE J1850 CRC8

The safety mechanism detects fault in stream of data provided by other FBs like *CPU* or *DMA*. The incorrect input data is detected by calculating the checksum of the input data based on the configured CRC polynomial and comparing it with the configured expected checksum value. Incase of a mismatch, an interrupt is triggered.

Init Conditions

SMC[SW]:FCE:CRC_CFG

Runtime conditions

SMC[SW]:FCE:CRC_CFG

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Interrupt Service Request	-

6.204 SM[HW]:FCE:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.205 SM[HW]:FCE:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.206 SM[HW]:FCE:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.207 SM[HW]:FPI:ADDRESS_MAP_MONITORING

Description

The FPI interconnect provides the capability to check whether a transaction address is outside the address ranges of the connected slave agents. On detection of a transaction address that is outside the address ranges of the connected slave agents, the safety mechanism signals an [Alarm] to *SMU*.

A slave agent provides the capability to check whether the type of access to the address is defined for access (read, write) in the Slave Agent address range. If the type of access is not defined for access, the safety mechanism signals an [Alarm] to SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.208 SM[HW]:FPI:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an [Alarm] signaled to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.209 SM[HW]:FPI:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.210 SM[HW]:FPI:ERROR_HANDLING

Description

If the *FPI* detects an FPI protocol error, the FPI signals an [Alarm] to the *SMU*

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB bus error event
SMU Alarm	ALM7[21] - BBB bus error event

6.211 SM[HW]:FPI:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.212 SM[HW]:FPI:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.213 SM[HW]:FPI:TRANSACTION_INTEGRITY

Description

If the *FPI* interconnect detects an integrity error during the address or data phases of an FPI transaction, the FPI interconnect signals an [Alarm] to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB bus error
SMU Alarm	ALM7[21] - BBB bus error

6.214 SM[HW]:GETH.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.215 SM[HW]:GETH.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.216 SM[HW]:GETH.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.

- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.217 SM[HW]:GETH.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[19] - Single bit error correction

6.218 SM[HW]:GETH.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.219 SM[HW]:GETH.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.220 SM[HW]:GETH.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:GETH.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.221 SM[HW]:GETH.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[21] - Miscellaneous error detection

6.222 SM[HW]:GETH:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.223 SM[HW]:GETH:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.224 SM[HW]:GETH:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.225 SM[HW]:GETH:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU	ALM11[9] - (SFI_SPB) SRI Read Data Phase Error

6.226 SM[HW]:GETH:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.227 SM[HW]:GPT12:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.228 SM[HW]:GPT12:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.229 SM[HW]:GPT12:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.230 SM[HW]:GPT12:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.231 SM[HW]:GTM.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[11] - Uncorrectable critical error detection

6.232 SM[HW]:GTM.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[11] - Uncorrectable critical error detection

6.233 SM[HW]:GTM.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.
- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[11] - Uncorrectable critical error detection

6.234 SM[HW]:GTM.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[10] - Single bit error correction

6.235 SM[HW]:GTM.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[11] - Uncorrectable critical error detection

6.236 SM[HW]:GTM.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[11] - Uncorrectable critical error detection

6.237 SM[HW]:GTM.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:GTM.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.238 SM[HW]:GTM.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[12] - Miscellaneous error detection

6.239 SM[HW]:GTM:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.240 SM[HW]:GTM:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.241 SM[HW]:GTM:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[1] - Safety FF uncorrectable error detected

6.242 SM[HW]:GTM:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.243 SM[HW]:GTM:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.244 SM[HW]:GTM:TOM_CCU6_MONITORING_WITH_IOM

Description

By using the IOM as safety mechanism and the *CCU6* as redundant path, the *GTM* is capable of detecting faults on TX PWM, which is generated by the GTM itself.

Init Conditions

SMC[SW]:IOM_CONFIG_FOR_GTM SMC[SW]:GTM:GTM_CONFIG_FOR ATOM

Runtime conditions

Tests

ESM[SW]:GTM:IOM_ALARM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[25] - Pin Mismatch Indication

6.245 SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM

Description

By using the IOM as safety mechanism and another TOM within the *GTM* as redundant path, the GTM is capable of detecting faults on TX PWM, which is generated by the GTM itself.
To make sure, that the GTM is clocked, it is recommended to use ESM[SW]:GTM:TIM_CLOCK_MONITORING in addition.To ensure, that the configuration has been written properly, the configuration should be read back and compared after writing the complete configuration. In addition to the compare check upfront, a configuration register needs to be checked regularly, to ensure that read works properly.

This is valid only when a redundant TOM/ATOM channel is available in the product

Init Conditions

SMC[SW]:IOM_CONFIG_FOR_GTM SMC[SW]:GTM:GTM_CONFIG_FOR ATOM

Runtime conditions

Tests

ESM[SW]:GTM:IOM_ALARM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[25] - Pin Mismatch Indication

6.246 SM[HW]:HSPDM.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.247 SM[HW]:HSPDM.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.248 SM[HW]:HSPDM.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.

- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.249 SM[HW]:HSPDM.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[6]- Single bit error correction

6.250 SM[HW]:HSPDM.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.251 SM[HW]:HSPDM.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.252 SM[HW]:HSPDM.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.253 SM[HW]:HSPDM.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[8] - Miscellaneous error detection

6.254 SM[HW]:HSPDM:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.255 SM[HW]:HSPDM:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.256 SM[HW]:HSPDM:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.257 SM[HW]:HSPDM:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.258 SM[HW]:HSSL:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.259 SM[HW]:HSSL:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.260 SM[HW]:HSSL:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.261 SM[HW]:HSSL:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU	ALM11[10] - (HSSL0) SRI Read Data Phase Error
SMU	ALM11[11] - (HSSL1) SRI Read Data Phase Error

6.262 SM[HW]:HSSL:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.263 SM[HW]:I2C:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.264 SM[HW]:I2C:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.265 SM[HW]:I2C:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.266 SM[HW]:I2C:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.267 SM[HW]:IR:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM10[22] - IR ACCEN Error Event

6.268 SM[HW]:IR:CFG_MONITOR

Description

This safety mechanism detects failures in *IR* configuration registers (especially invalid TOS). A failure is detected if a Service Request for a non existing ISP is pending. Detected failures are reported via an Alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[22] - EDC Configuration & Data Path Error

6.269 SM[HW]:IR:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.270 SM[HW]:IR:FFI_CONTROL

Description

The safety mechanism detects and prevents interferences to the following *IR* control fields:
Service Request Broadcast (SRB) Registers protection via ACCEN_SRBx0, ACCEN_SRBx1
Service Request Control (SRC) Register

- SRC[31:16] protection via ACCEN_SRC_TOSx0, ACCEN_SRC_TOSx1
- SRC[15:0] protection via ACCEN_CONFIG0 , ACCEN_CONFIG1

ECR - Error Capture (ECR) Registers protection via ACCEN_CONFIG0, ACCEN_CONFIG1
On detection of an access protection violation the IR signals an alarm to the *SMU*.

Init Conditions

SMC[SW]:IR:FFI_CONTROL

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM10[22] - IR_R_ACC_EN

6.271 SM[HW]:IR:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[2] - Safety FF uncorrectable error detected

6.272 SM[HW]:IR:ISP_MONITOR

Description

The *IR* implements an *EDC* providing an end-to-end protection of *SRN* configuration.

- End-to-end means: from SRN configuration, through the Interrupt Control Unit *ICU*, through Interrupt Service Provider *ISP*, back to the ICUwith ISP acknowledge, up the point in IR where the acknowledged Service Request information is used to clear the related SRN.

- SRN clear means: a Service Request Node is changed to the state ´No Service Request Pending´.

- SRN configuration information means: mapping to ISP configuration, Service Request Priority Number *SRPN*, Service Request Node Enable and unique SRN Index Number.

Detected failures are reported via an Alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[22] - EDC Configuration & Data Path Error

6.273 SM[HW]:IR:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[2] - IR Safety FF uncorrectable error detected
SMU Alarm	ALM10[21] - Miscellaneous Safety FF uncorrectable error detected

6.274 SM[HW]:IR:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.275 SM[HW]:IR:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers are defined with an access privilege level that restricts the modification to master functions with supervisor privilege level.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.276 SM[HW]:LMU.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.277 SM[HW]:LMU.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.278 SM[HW]:LMU.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[0] - Single bit error correction
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.279 SM[HW]:LMU.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.280 SM[HW]:LMU.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.281 SM[HW]:LMU.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:LMU.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.282 SM[HW]:LMU.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[2] - Miscellaneous error detection

6.283 SM[HW]:LMU:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[15] - Bus-level MPU error

6.284 SM[HW]:LMU:CONTROL_REDUNDANCY

Description

The LMU control redundancy replicates the logic used to control the transfer of data between the *SRI* slave interface and the SSH wrapper around the RAM. The control redundancy detects permanent and transient failures of the control logic by the way of hardware redundancy, using two independent control logic instances called the master and the checker. The two instances of the control logic operate in a redundant manner: they use the same input signals and execute the same operations. All the functional outputs of the master control logic and the checker control logic are compared on a cycle by cycle basis using a hardware comparator. The redundant operation has no effect on the execution of read or write accesses. Additional measures are implemented in hardware to mitigate common cause faults between the redundant control logic instances.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[12] - LMU comparator error
SMU Alarm	ALM7[13] - LMU control error

6.285 SM[HW]:LMU:CONTROL_REDUNDANCY_SCC

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[12] - LMU comparator error

6.286 SM[HW]:LMU:ECC_MONITOR

Description

LMU SRAM has a monitor mechanism that controls the *ECC* output. In case the ECC performs a wrong correction the ECC monitor forwards an alarm to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[16] - EDC read error

6.287 SM[HW]:LMU:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation of the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.288 SM[HW]:LMU:READ_WRITE_ECC

Description

This is an ECC check of the data transferred between the *SRI* slave interface and the SSH wrapper around the RAM. This mechanism cannot correct errors but will detect permanent or transient errors affecting the read or write data.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[14] - ECC error

6.289 SM[HW]:LMU:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.290 SM[HW]:LMU:SRI_ADDRESS_MAP_MONITORING

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.291 SM[HW]:LMU:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[0] - (LMU) SRI Address Phase Error
SMU Alarm	ALM11[1] - (LMU) SRI Write Data Phase Error

6.292 SM[HW]:LMU:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event

6.293 SM[HW]:LMU:VM_AS_AP

Description

If a master access the memory via the SRI-Bus or the local CPU access the memory, the safety mechanism checks if an individual master tag identifier is enabled to access a region of volatile memory. In case the individual master tag is disabled for accesses an alarm will be raised.

Init Conditions

Runtime conditions

ESM[SW]:CPU:BUS_MPU_INITCHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[15] - Bus-level MPU error

6.294 SM[HW]:MCMCAN.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[17] - Uncorrectable critical error detection

6.295 SM[HW]:MCMCAN.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[17] - Uncorrectable critical error detection

6.296 SM[HW]:MCMCAN.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.
- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[17] - Uncorrectable critical error detection

6.297 SM[HW]:MCMCAN.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[16] - Single bit error correction

6.298 SM[HW]:MCMCAN.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[17] - Uncorrectable critical error detection

6.299 SM[HW]:MCMCAN.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[17] - Uncorrectable critical error detection

6.300 SM[HW]:MCMCAN.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:MCMCAN.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.301 SM[HW]:MCMCAN.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[18] - Miscellaneous error detection

6.302 SM[HW]:MCMCAN:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.303 SM[HW]:MCMCAN:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.304 SM[HW]:MCMCAN:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.305 SM[HW]:MCMCAN:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.306 SM[HW]:MCU:LBIST

Description

LBIST is a safety mechanism which executes structural test and checks the *MCU* digital logic to detect Hard Errors: latent and multi-point faults.

Init Conditions

SMC[SW]:MCU:LBIST_CFG

Runtime conditions

ESM[SW]:MCU:LBIST_RESULT

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.307 SM[HW]:MSC:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	**SM Flag**
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.308 SM[HW]:MSC:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	**SM Flag**
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.309 SM[HW]:MSC:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.310 SM[HW]:MSC:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.311 SM[HW]:NVM.FSIRAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - FSI SRAM Uncorrectable critical error

6.312 SM[HW]:NVM.FSIRAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - FSI SRAM Uncorrectable critical error

6.313 SM[HW]:NVM.FSIRAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[0] - FSI SRAM Correctable critical error
SMU Alarm	ALM7[1] - FSI SRAM Uncorrectable critical error

6.314 SM[HW]:NVM.FSIRAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - FSI SRAM Uncorrectable critical error

6.315 SM[HW]:NVM.FSIRAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[1] - FSI SRAM Uncorrectable critical error

6.316 SM[HW]:NVM.FSIRAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.317 SM[HW]:NVM.FSIRAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[2] - FSI SRAM Miscellaneous critical error

6.318 SM[HW]:NVM.PFLASH:AS_AP

Description

If a local or remote master performs a read access to the a local *PFLASH*, the safety mechanism checks if an individual master tag identifier is enabled to read a region of PFLASH.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[1] - CPUx (x=0..5)

6.319 SM[HW]:NVM.PFLASH:CONTROL_REDUNDANCY

Description

The control redundancy logic detects permanent and transient failures of the NVM.PFLASH read path control logic by the way of hardware redundancy, using two independent NVM.PFLASH read path control logic instances. The two NVM.PFLASH read path instances operate in a lockstep manner: The instances use the same input data, execute the same operation, and all the functional outputs of the two instances are compared on a cycle by cycle basis using a hardware comparator. Alarm generation is independent to CPU:PFLASH alarm.

Note : This safety mechanism aims to cover faults that can happen on the NVM side of the PFLASH. Faults occuring on the CPU side of the PFLASH will be covered by SM[HW]:CPU:CONTROL_REDUNDANCY

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[2] - PFLASHx Read Path Error (x=0..5)

6.320 SM[HW]:NVM.PFLASH:CONTROL_REDUNDANCY_SCC

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[2] - PFLASHx Read Path Error (x=0..5)

6.321 SM[HW]:NVM.PFLASH:ECC_ONLINE_CHECKER

Description

The NVM.PFLASH *ECC* has the ability to detect and correct errors (DECTED). The correction logic has the ability to corrupt data by performing a wrong correction. To mitigate this issue the NVM.PFLASH ECC online monitor continuously monitors the NVM.PFLASH ECC correction logic and checks that the corrected word and the ECC bits belong to the valid code word vector space. If an error is detected a SMU alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[28] - PFLASH ECC Error

6.322 SM[HW]:NVM.PFLASH:EDC_COMPARATOR

Description

The NVM.PFLASH *ECC* has the ability to detect and correct errors (DECTED). The detection logic itself could be corrupted and could lead to erroneous information about the data errors. A redundant detection logic and comparator is making sure that such faults are visible. If an error is detected a SMU alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[29] - PFLASH EDC Error

6.323 SM[HW]:NVM.PFLASH:ERROR_CORRECTION

Description

The NVM.PFLASH stores data with Error Correcting Codes *ECC* in order to protect against data corruption. Data in NVM.PFLASH uses an ECC code with *DEC-TED* capabilities. Each block of 256 data bits is accompanied with a set of ECC bits. No initialization required. The ECC logic is operational after reset.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[22] - PFLASH Single Bit Correctable Error
SMU Alarm	ALM7[23] - PFLASH Double Bit Correctable Error

6.324 SM[HW]:NVM.PFLASH:ERROR_MANAGEMENT

Description

The safety mechanism monitors the ECC errors found in a Program_NVM bank. The safety mechanism enables the application to count the number and the type of individual events as well as to reconstruct the system address of the fail. The same address is stored only once for each type of fault. The safety mechanism enables the application to track if the *ECC* error was a correctable error (stored in the single bit and double bit address error buffers) or if it was an uncorrectable data or address error (stored in the multi bit address error buffer).

If the error threshold in one of the following buffers is reached, the safety mechanism triggers an alarm to the *SMU*:

- Single Bit error Address Buffer (SBAB)
- Double Bit error Address Buffer (DBAB)
- Multi Bit error Address Buffer (MBAB)
- all Zero Bit error Address Buffer (ZBAB)

The threshold of the double bit error buffer is halved in case the single bit error buffer is full.

Note : the zero bit error address buffer is a debugging feature for empty locations, an all zero error is also counted as multi bit error.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[24] - PFLASH Single Bit Address Buffer Full
SMU Alarm	ALM7[25] - PFLASH Double Bit Address Buffer Full
SMU Alarm	ALM7[26] - PFLASH Multiple Bit Address Buffer Full
SMU Alarm	ALM7[27] - PFLASH Zero Bit Address Buffer Full

6.325 SM[HW]:NVM.PFLASH:FLASHCON_MONITOR

Description

The *NVM* configuration register CPUx_FLASHCON2 possess redundant configuration bits to be set by the application software to configure the PFlash NVM. The FLASHCON_MONITOR detects illegal values from incorrect software or random hardware faults from this register to the PFlash NVM and correct unwanted illegal values (00b,11b becomes 01b, considered as "safe setting").

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[30] - FLASHCON Configuration Error

6.326 SM[HW]:NVM.PFLASH:INCORRECT_OPERATION_PROTECTION

Description

The *NVM* continuously monitors each bank of NVM.PFLASH for an unintended NVM operation (for example, erase, program, etc.). If the NVM detects an unintended NVM operation on any bank of NVM.PFLASH, the NVM triggers an alarm to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[2] - PFLASHx Read Path Error (x=0..5)

6.327 SM[HW]:NVM.PFLASH:WAIT_STATE_PROTECTION

Description

The safety mechanism checks for unintentional changes in the wait state configuration. The wait state configuration is protected by an ECC mechanism. For an unintentional change an alarm is generated for the SMU.

Note:

The NVM continuously monitors the wait state configuration which controls the number of wait states during a read to NVM.PFLASH. Incorrect wait state leads to a corrupted read.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALMx[2] - PFLASHx Read Path Error (x=0..5)

6.328 SM[HW]:NVM:ARRAY_POINTER

Description

The safety mechanism is detecting non-expected flash bank selection blocking or disturbing the NVM reads. The safety mechanism helps detect UCODE errors (like executing an NVM operation on the wrong bank). If a non-expected NVM bank is selected an alarm is generated for the SMU.

Note : the NVM store locally the configuration “flash bank under system access and available for read” or “flash bank available for NVM operation”.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[31] - Flash Stored Configuration Error

6.329 SM[HW]:NVM:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event

6.330 SM[HW]:NVM:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event

6.331 SM[HW]:NVM:MISR

Description

The safety mechanism checks for read path settings stored in register banks (SFR read settings, bitline redundancy and wordline redundancy). The settings are checked cyclic and prevent the faults from becoming latent. If an incorrect setting is detected for a read operation an alarm is generated for the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[31] - Flash Stored Configuration Error

6.332 SM[HW]:NVM:REDINVFF

Description

The safety mechanism checks for errors in quasi-static PFLASH register bits potentially stored for a long period (read settings and valid bits from the BABRECORD registers).

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[31] - Flash Stored Configuration Error

6.333 SM[HW]:NVM:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event

6.334 SM[HW]:NVM:SRI_ADDRESS_MAP_MONITORING

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event

6.335 SM[HW]:NVM:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[4] - (NVM) SRI Address Phase Error

Alarm Interface	SM Flag
SMU Alarm	ALM11[5] - (NVM) SRI Write Data Phase Error

6.336 SM[HW]:NVM:STARTUP_PROTECTION

Description

MCU Startup Protection. Monitors STARTUP PROTECTION property on interconnect accesses to the functional block configuration address space (TriCore segment F). The configuration addresses for which this protection applies is defined at specification time.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event

6.337 SM[HW]:NVM:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event

6.338 SM[HW]:PMS:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.339 SM[HW]:PMS:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.340 SM[HW]:PMS:MONBIST

Description

After reset release, MONBIST for the secondary monitors and alarm generation path may be carried out by *Application SW*. Secondary Monitor BIST ensures a higher latent fault coverage for the secondary monitors and the associated alarm and Error Pin (*FSP*) fault logic routed to *SMU*.

Init Conditions

SMC[SW]:PMS:MONBIST_CFG

Runtime conditions

ESM[SW]:PMS:MONBIST_RESULT

Tests

Fault identification interfaces

Alarm Interface	**SM Flag**
Application SW Error Handler	N.A.

6.341 SM[HW]:PMS:MON_REDUNDANCY

Description

The tracking *ADC* in Primary monitor of *PMS* also works as a redundant counter part for the SAR ADC used in secondary monitor, hence helping in increasing the latent fault coverage of PMS block.

Init Conditions

SMC[SW]:PMS:MON_REDUNDANCY_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	**SM Flag**
Application SW Error Handler	N.A.

6.342 SM[HW]:PMS:PBIST

Description

A Power Built-In-Self-Test (PBIST) at start-up allows the testing of supply levels, power functions and voltage monitors before cold PORST reset release.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.343 SM[HW]:PMS:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[5] - PMS Safety FF uncorrectable error detected
SMU Alarm	ALM10[21] - Miscellaneous Safety FF uncorrectable error detected

6.344 SM[HW]:PMS:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.345 SM[HW]:PMS:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.346 SM[HW]:PMS:VDDM_MONITOR

Description

Detects whether VDDM (analog supply) is within expected range. Detection of over and under voltage, results in voltage monitor alarm.

Note: In this case, upon apperence of an infrastructure faults, the system integrator is responsible to decide on the appropriate reaction.
As a recommendation:

SMU_stdby hardware alarm path described in ***SM[HW]:PMS:VDDM_MONITOR*** can be replaced by software cyclic read of AG2i_STDBY or EVRSTAT.UVC and EVRSTAT.OVC registers AND triggering alarms in parallel by SMU_core commands and ***ESM[HW]:SYS:WATCHDOG_FUNCTION*** in case the software execution is not reliable due to infrastructure failure

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM20[7] - VDDM Over-voltage Alarm
SMU Alarm	ALM20[13] - VDDM Under-voltage Alarm
SMU Alarm	ALM9[5] - Under-voltage Alarm
SMU Alarm	ALM9[3] - Over-voltage Alarm

6.347 SM[HW]:PMS:VDDP3_MONITOR

Description

Detects whether VDDP3 (3.3V supply generated by EVR or from external) is within expected range. Detection of over and under voltage, results in voltage monitor alarm.

Init Conditions

SMC[SW]:PMS:Vx_MONITOR_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM20[6] - VDDP3 Over-voltage Alarm
SMU Alarm	ALM9[3] - Over-voltage Alarm
SMU Alarm	ALM9[15] - Short to low/high Alarm
SMU Alarm	ALM9[5] - Under-voltage Alarm
SMU Alarm	ALM20[12] - VDDP3 Under-voltage Alarm

6.348 SM[HW]:PMS:VDDPD_MONITOR

Description

Detects whether VDDPD Pre-Reg supply voltage is within expected range. Detection of over and under voltage results in voltage monitor alarm.

Init Conditions

SMC[SW]:PMS:Vx_MONITOR_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[3] - Over-voltage Alarm
SMU Alarm	ALM9[5] - Under-voltage Alarm
SMU Alarm	ALM20[5] - VDDPD Over-voltage Alarm
SMU Alarm	ALM20[11] - VDDPD Under-voltage Alarm

6.349 SM[HW]:PMS:VDD_MONITOR

Description

Detects whether VDD (1.3V supply generated by EVR or from external) is within expected range. Detection of over and under voltage, results in voltage monitor alarm.

Init Conditions

SMC[SW]:PMS:Vx_MONITOR_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM20[4] - VDD Over-voltage Alarm
SMU Alarm	ALM9[3] - Over-voltage Alarm
SMU Alarm	ALM20[10] - VDD Under-voltage Alarm
SMU Alarm	ALM9[5] - Under-voltage Alarm
SMU Alarm	ALM9[15] - Short to low/high Alarm

6.350 SM[HW]:PMS:VEVRSB_MONITOR

Description

Detects whether VEVRSB (standby domain supply) is within expected range. Detection of over and under voltage, results in voltage monitor alarm.

Init Conditions

SMC[SW]:PMS:Vx_MONITOR_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM20[9] - VEVRSB Over-voltage Alarm
SMU Alarm	ALM9[3] - Over-voltage Alarm
SMU Alarm	ALM20[15] - VEVRSB Under-voltage Alarm
SMU Alarm	ALM9[5] - Under-voltage Alarm

6.351 SM[HW]:PMS:VEXT_MONITOR

Description

Detects whether VEXT (3.3 V / 5V external supply) is within valid ranges, which can be configured by registers. Detection of over and under voltage, results in voltage monitor alarm.

Init Conditions

SMC[SW]:PMS:Vx_MONITOR_CFG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM20[8] - VEXT Over-voltage Alarm
SMU Alarm	ALM9[3] - Over-voltage Alarm
SMU Alarm	ALM20[14] - VEXT Under-voltage Alarm
SMU Alarm	ALM9[5] - Under-voltage Alarm

6.352 SM[HW]:PORT:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.353 SM[HW]:PORT:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.354 SM[HW]:PORT:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.355 SM[HW]:PORT:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.356 SM[HW]:PSI5.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.357 SM[HW]:PSI5.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.358 SM[HW]:PSI5.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.
- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.359 SM[HW]:PSI5.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[19] - Single bit error correction

6.360 SM[HW]:PSI5.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.361 SM[HW]:PSI5.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.362 SM[HW]:PSI5.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:PSI5.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.363 SM[HW]:PSI5.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[21] - Miscellaneous error detection

6.364 SM[HW]:PSI5:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.365 SM[HW]:PSI5:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.366 SM[HW]:PSI5:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.367 SM[HW]:PSI5:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.368 SM[HW]:PSI5S:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.369 SM[HW]:PSI5S:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.370 SM[HW]:PSI5S:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.371 SM[HW]:PSI5S:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.372 SM[HW]:QSPI:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.373 SM[HW]:QSPI:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.374 SM[HW]:QSPI:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.375 SM[HW]:QSPI:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.376 SM[HW]:RESET:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.377 SM[HW]:RESET:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.378 SM[HW]:RESET:FF_MONITORING

Description

RESET detects permanent and transient faults in configuration registers as well as in flip flops (FF) controling reset request processing and reset execution.
RESET sends alarm to SMU if a fault was detected.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[4] - Safety FF uncorrectable error detected

6.379 SM[HW]:RESET:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[4] - SCU Safety FF uncorrectable error detected

6.380 SM[HW]:RESET:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.381 SM[HW]:RIF:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.382 SM[HW]:RIF:CFG_MONITOR

Description

The *RIF* configuration checksum is a cyclic redundancy checksum calculated from data stored in the RIF configuration registers. The calculation of the RIF configuration checksum shall fulfil the IEEE 802.3 standard. The RIF periodically parses the RIF configuration registers to calculate a RIF configuration checksum. The RIF compares the calculated RIF configuration checksum with an expected RIF configuration checksum. If the RIF does not match the calculated and expected RIF configuration checksums, the RIF triggers an alarm.

Init Conditions

SMC[SW]:RIF:CFG_MONITOR

Runtime conditions

Tests

ESM[SW]:RIF:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[30] - RIF0 Safety Alarm
SMU Alarm	ALM9[31] - RIF1 Safety Alarm

6.383 SM[HW]:RIF:CONTROL_MONITOR

Description

The *RIF* duplicates the state machine and datapath. The RIF continuously monitors the outputs of the duplicated logic. If the RIF detects an output mismatch, the RIF triggers an alarm.

Init Conditions

SMC[SW]:RIF:CONTROL_MONITOR

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[30] - RIF0 Safety Alarm
SMU Alarm	ALM9[31] - RIF1 Safety Alarm

6.384 SM[HW]:RIF:CONTROL_MONITOR_SCC

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[30] - RIF0 Safety Alarm
SMU Alarm	ALM9[31] - RIF1 Safety Alarm

6.385 SM[HW]:RIF:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.386 SM[HW]:RIF:ERROR_HANDLING

Description

The *MMIC* data checksum is a cyclic redundancy checksum calculated and appended to data transmitted from the MMIC to the *RIF*. The calculation of the MMIC data checksum fulfils the IEEE 802.3 standard. If the MMIC transmits radar data to the RIF, the MMIC shall append the radar data with a MMIC data checksum. As long as the MMIC is transmitting radar data to the RIF, the RIF shall calculate a MMIC data checksum for the received radar data. As soon as the RIF receives the appended MMIC data checksum, the RIF compares the calculated MMIC data checksum to the appended MMIC data checksum. If the RIF does not match the calculated and appended MMIC data checksums, the RIF triggers an interrupt.

Init Conditions

SMC[SW]:RIF:ERROR_HANDLING

Runtime conditions

ESM[SW]:RIF:ERROR_HANDLING

Tests

ESM[SW]:RIF:ERROR_HANDLING

Fault identification interfaces

Alarm Interface	SM Flag
Interrupt Service Request	N.A.

6.387 SM[HW]:RIF:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.388 SM[HW]:RIF:SM_CHECK

Description

The SM[HW]:RIF:SM_CHECK safety mechanism monitors the integrity of other *RIF* safety mechanisms. Any detected failure in the safety mechanisms will cause a status bit to be set in INTCON register. The safety mechanism operates by maintaining a duplicate copy of all the monitored logic ("active" logic). The duplicate copy has the logic states of the registers inverted. The outputs of the "active" logic and the duplicate copy are compared with a hardware comparator

An alarm is always generated if the "active" copy of the logic detects an error. This safety mechanism will not modify that behaviour. The status bit in INTCON is set if the state of the duplicate logic differs from the state of the "active" logic. The expanded truth table is as follows:

1. there is an error and both hardware monitoring have the same result ->
   a. RIF SMU Alarm,
   b. bit is not set INTCON.SMCF
2. there is an error but both hardware monitoring have a different result ->
   a. RIF SMU Alarm +
   b. bit set in INTCON.SMCF
3. there is no error and both hardware monitoring have the same result ->
   a. no Alarm,
   b. bit is not set INTCON.SMCF
4. there is no error but both hardware monitoring have a different result ->
   a. no alarm,
   b. only INTCON.SMCF bit set

The safety mechanism is used to monitor the following RIF safety mechanisms:

- SM[HW]:RIF:CONTROL_MONITOR
- SM[HW]:RIF:SPU_INTERFACE
- SM[HW]:RIF:CFG_MONITOR

Init Conditions

Runtime conditions

ESM[SW]:RIF:SM_CHECK

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Application SW Error Handler	N.A.

6.389 SM[HW]:RIF:SPU_INTERFACE

Description

The *RIF SPU* interface checksum is a cyclic redundancy checksum calculated and appended to data transmitted from the RIF to the SPU. The calculation of the RIF SPU interface checksum shall fulfil the IEEE 802.3 standard. The RIF transmits a variable length data stream of 32-bit data packets to the SPU. The RIF appends each data stream with a RIF SPU interface checksum. As long as the SPU receives an incoming data stream from the RIF, the SPU calculates a RIF SPU interface checksum for the incoming data stream. As soon as the SPU has received a complete data stream from the RIF, the SPU compares the calculated RIF SPU interface checksum and the appended RIF SPU checksum. If the SPU does not match the calculated and appended RIF SPU interface checksums, the SPU triggers an alarm.

Init Conditions

SMC[SW]:RIF:SPU_INTERFACE

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[28] - SPU0 Safety Alarm
SMU Alarm	ALM9[29] - SPU1 Safety Alarm

6.390 SM[HW]:RIF:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.391 SM[HW]:SCR.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.392 SM[HW]:SCR.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.393 SM[HW]:SCR.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.
- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.394 SM[HW]:SCR.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[19] - Single bit error correction

6.395 SM[HW]:SCR.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.396 SM[HW]:SCR.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.397 SM[HW]:SCR.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SCR.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.398 SM[HW]:SCR.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[21] - Miscellaneous error detection

6.399 SM[HW]:SCU.ERU:EXT_ALARM_GENERATION

Description

The failure detected by external safety mechanisms can be reported by the *ERU* pattern detection function based on external requests generated as inputs to ERU. In case of failure detection by ERU logic, it reports them as Alarms to SMU.

Init Conditions

SMC[SW]:SCU:ERU_CONFIG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[6] - External Request Unit Alarm 0
SMU Alarm	ALM8[7] - External Request Unit Alarm 1
SMU Alarm	ALM8[8] - External Request Unit Alarm 2
SMU Alarm	ALM8[9] - External Request Unit Alarm 3
SMU Alarm	ALM8[26] - External Request Unit Alarm 4
SMU Alarm	ALM8[27] - External Request Unit Alarm 5
SMU Alarm	ALM8[28] - External Request Unit Alarm 6
SMU Alarm	ALM8[29] - External Request Unit Alarm 7

6.400 SM[HW]:SCU:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.401 SM[HW]:SCU:EMERGENCY_STOP

Description

This safety mechanism forces selected output ports to be immediately set into a defined state. The EMERGENCY_STOP mechanism can be used as a fast reaction on detected failures modes without the intervention of application SW.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[19] - External Emergency Stop Signal Event

6.402 SM[HW]:SCU:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.403 SM[HW]:SCU:ENDINIT_WATCHDOG

Description

SCU End of Initialization (*ENDINIT*) Timeout Protection. If the ENDINIT is not terminated by *Application SW* within a limited time window, a Watchdog alarm is issued and forwarded to the *SMU*. The duration of the ENDINIT protection deactivation is controlled and monitored by the watch timers.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[10] - CPU0 Watchdog time-out
SMU Alarm	ALM8[11] - CPU1 Watchdog time-out
SMU Alarm	ALM8[12] - CPU2 Watchdog time-out
SMU Alarm	ALM8[13] - CPU3 Watchdog time-out
SMU Alarm	ALM8[14] - CPU4 Watchdog time-out
SMU Alarm	ALM8[15] - CPU5 Watchdog time-out
SMU Alarm	ALM8[17] - All Watchdogs watchdog time-out. This alarm is a logical OR over all watchdog time-out alarms.

6.404 SM[HW]:SCU:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[4] - Safety FF uncorrectable error detected

6.405 SM[HW]:SCU:LOCKSTEP_MONITOR

Description

The safety mechanism detects transient faults in sequential logic of the CPU lockstep control. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[18] - LockStep DualRail Error

6.406 SM[HW]:SCU:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[4] - SCU Safety FF uncorrectable error detected
SMU Alarm	ALM10[20] - CCU Safety FF correctable error detected
SMU Alarm	ALM10[21] - Miscellaneous Safety FF uncorrectable error detected

6.407 SM[HW]:SCU:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.408 SM[HW]:SCU:SAFETY_WATCHDOG

Description

SCU End of Safety Initialization ([SAFETY_ENDINIT]) Timeout Protection. If the [SAFETY_ENDINIT] is not terminated by *Application SW* within a limited time window, a Safety Watchdog alarm is issued and forwarded to the *SMU*. The duration of the [SAFETY_ENDINIT] protection deactivation is controlled and monitored by the watch timers.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM8[16] - Safety Watchdog time-out

Alarm Interface	SM Flag
SMU Alarm	ALM8[17] - All Watchdogs watchdog time-out. This alarm is a logical OR over all watchdog time-out alarms.

6.409 SM[HW]:SCU:STARTUP_PROTECTION

Description

MCU Startup Protection. Monitors STARTUP PROTECTION property on interconnect accesses to the functional block configuration address space (TriCore segment F). The configuration addresses for which this protection applies is defined at specification time.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.410 SM[HW]:SCU:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.411 SM[HW]:SDMMC.RAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.412 SM[HW]:SDMMC.RAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.413 SM[HW]:SDMMC.RAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.
- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.414 SM[HW]:SDMMC.RAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[19] - Single bit error correction

6.415 SM[HW]:SDMMC.RAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.416 SM[HW]:SDMMC.RAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[20] - Uncorrectable critical error detection

6.417 SM[HW]:SDMMC.RAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SDMMC.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.418 SM[HW]:SDMMC.RAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[21] - Miscellaneous error detection

6.419 SM[HW]:SDMMC:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.420 SM[HW]:SDMMC:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.421 SM[HW]:SDMMC:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.422 SM[HW]:SDMMC:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.423 SM[HW]:SENT:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.424 SM[HW]:SENT:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.425 SM[HW]:SENT:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.426 SM[HW]:SENT:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.427 SM[HW]:SMU:ALIVE_MONITOR

Description

Redundant *SMU* in Standby domain monitors activity of the SMU in the Core domain by checking its Alive Alarm. In case the SMU in Standby domain by means of the ***SM[HW]:SMU:CCF_MONITOR*** detects an incoming Alive Alarm, it asserts a configured reaction. The reaction, in case it is configured, is the ErrorPins (FSP[1:0]) activation.

Note: Please refer to the note in ***ESM[HW]:SYS:FSP_ERROR_PIN_MONITOR*** if the SMU_stdby is not used to control ErrorPins (FSP[1:0]) to react on the SMU_core Alive Alarm.

Init Conditions

SMC[SW]:SMU:CONFIG

Runtime conditions

Tests

ESM[SW]:SMU:ALIVE_ALARM_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM21[16] - SMU Alive Monitor Alarm

6.428 SM[HW]:SMU:CCF_MONITOR

Description

Redundant *SMU* in Standby domain is implemented as diverse HW redundancy counterpart of the SMU in the Core domain. It is capable to activate FSP[0:1] Error pins to the fault state in presence of Alarms generated by voltage or clock monitors or by the ***SM[HW]:SMU:ALIVE_MONITOR***.

Init Conditions

SMC[SW]:SMU:CONFIG

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM20[x] - SMU_Stdby Alarms (x=3..15, 31)
SMU Alarm	ALM21[x] - SMU_Stdby Alarms (x=0..5, 7..16)

6.429 SM[HW]:SMU:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB bus error

6.430 SM[HW]:SMU:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.431 SM[HW]:SMU:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[7] - Safety FF uncorrectable error detected

6.432 SM[HW]:SMU:FSP_MONITOR

Description

The *FSP* error pin monitor detects assertion of the FSP output. In case the FSP activation is detected, the monitor reports it via an *SMU* Alarm.

ESM[SW]:SMU:APPLICATION_SW_ALARM can also be used to trigger FSP output.

In order to detect the missing activation of FSP, please refer to the note section of ***ESM[HW]:SYS:FSP_ERROR_PIN_MONITOR***.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM10[18] - FSP activation

6.433 SM[HW]:SMU:LOCK

Description

The SMU configuration is locked. A key has to be written to the SMU_KEY register to enable an SMU configuration. A permanent lock might be used to disable any further configuration.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
No Alarm and no Status	

6.434 SM[HW]:SMU:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[7] - SMU_core Safety FF uncorrectable error detected
SMU Alarm	ALM10[21] - Miscellaneous Safety FF uncorrectable error detected

6.435 SM[HW]:SMU:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs. Detected failures are reported via *SMU* status registers.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.436 SM[HW]:SMU:RT

Description

The *SMU* implements two Recovery Timers, RT0 and RT1, to monitor the response time of a *RESET*, *NMI* or interrupt triggered by an alarm. If the RT is not serviced before it times out, an alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM10[16] - RT0 timeout
SMU Alarm	ALM10[17] - RT1 timeout

6.437 SM[HW]:SMU:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.438 SM[HW]:SMU:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.439 SM[HW]:SPU.BUFFER:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.440 SM[HW]:SPU.BUFFER:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.441 SM[HW]:SPU.BUFFER:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[6] - Single bit error correction
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.442 SM[HW]:SPU.BUFFER:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.443 SM[HW]:SPU.BUFFER:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.444 SM[HW]:SPU.BUFFER:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SPU.BUFFER:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
A pplication SW Error Handler	N.A.

6.445 SM[HW]:SPU.BUFFER:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[8] - Miscellaneous error detection

6.446 SM[HW]:SPU.CONFIG:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.447 SM[HW]:SPU.CONFIG:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- An fault has been detected in one of the registers in the SSH leading to an error in the SRAM access,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.448 SM[HW]:SPU.CONFIG:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[6] - Single bit error correction
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.449 SM[HW]:SPU.CONFIG:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.450 SM[HW]:SPU.CONFIG:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.451 SM[HW]:SPU.CONFIG:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SPU.CONFIG:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
Ap plication SW Error Handler	N.A.

6.452 SM[HW]:SPU.CONFIG:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[8] - Miscellaneous error detection

6.453 SM[HW]:SPU.FFT:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.454 SM[HW]:SPU.FFT:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.455 SM[HW]:SPU.FFT:ERROR_CORRECTION

Description

The SRAM implements a *SEC DED ECC* logic. This mechanism corrects *SBE* and detects SBE as well as *DBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*. In case an DBE is detected or ECE is disabled and an SBE is detected a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[6] - Single bit error correction
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.456 SM[HW]:SPU.FFT:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.457 SM[HW]:SPU.FFT:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[7] - Uncorrectable critical error detection

6.458 SM[HW]:SPU.FFT:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SPU.FFT:REG_MONITOR_TEST

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
Application SW Error Handler	N.A.

6.459 SM[HW]:SPU.FFT:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Monitored Function Mode	Safety Mechanism Mode
SMU Alarm	ALM7[8] - Miscellaneous error detection

6.460 SM[HW]:SPU.SPUCORE:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.461 SM[HW]:SPU.SPUCORE:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.462 SM[HW]:SPU:ACTIVE_CFG_CHECK

Description

The *SPU* configuration checksum is a cyclic redundancy checksum calculated for SPU configuration data. The calculation of the SPU configuration checksum within the SPU fulfils the IEEE 802.3 standard. The SPU periodically checks the integrity of configuration data and calculates a SPU configuration checksum. On completion of the calculation, the SPU compares the calculated SPU configuration checksum with an expected SPU configuration checksum stored in the SPU. If the SPU does not match the calculated and expected SPU configuration checksums, the SPU triggers an alarm.

Init Conditions

SMC[SW]:SPU:INIT_CONFIGURATION

Runtime conditions

Tests

ESM[SW]:SPU:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[28] - SPU0 Safety Alarm
SMU Alarm	ALM9[29] - SPU1 Safety Alarm

6.463 SM[HW]:SPU:BYPASS_CRC

Description

If the *SPU* bypass function is enabled, the SPU safety mechanism monitors the integrity of *RIF* data stored in the *EMEM* by performing a CRC check. If the SPU safety mechanism fails to match a calculated and appended checksum, the SPU safety mechanism triggers an alarm.

Init Conditions

SMC[SW]:SPU:BYPASS_CRC

Runtime conditions

Tests

ESM[SW]:SPU:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[28] - SPU0 Safety Alarm
SMU Alarm	ALM9[29] - SPU1 Safety Alarm

6.464 SM[HW]:SPU:CONTROL_FLOW_SIGNATURE

Description

The safety mechanism computes a signature that enables the *Application SW* to detect any failure in the control operations of the *SPU* within a measurement cycle.

Init Conditions

SMC[SW]:SPU:CONTROL_FLOW_SIGNATURE

Runtime conditions

ESM[SW]:SPU:CONTROL_FLOW_SIGNATURE

Tests

ESM[SW]:SPU:CONTROL_FLOW_SIGNATURE

Fault identification interfaces

Alarm Interface	SM Flag
Application SW Error Handler	N.A.

6.465 SM[HW]:SPU:EMEM_INTERFACE

Description

If the *SPU* performs a read or write access to *EMEM*, the safety mechanism checks the integrity of the access. If the safety mechanism detects an integrity error, the safety mechanism triggers an alarm.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SPU:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[28] - SPU0 Safety Alarm
SMU Alarm	ALM9[29] - SPU1 Safety Alarm

6.466 SM[HW]:SPU:EMEM_TILE

Description

The safety mechanism checks that an *SPU* access to the addressed *EMEM* tile is enabled. If the SPU access to the addressed EMEM tile is not enabled, the safety mechanism triggers an alarm.

Init Conditions

SMC[SW]:SPU:EMEM_TILE

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[28] - SPU0 Safety Alarm
SMU Alarm	ALM9[29] - SPU1 Safety Alarm

6.467 SM[HW]:SPU:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.468 SM[HW]:SPU:PARTIAL_REDUNDANCY

Description

The safety mechanism monitors *SPU* random hardware faults e.g. in a dual-SPU 8-antennae configuration. The safety mechanism uses the two functional SPU instances in a redundant mode to compare control information of functional outputs only. The safety mechanism does not monitor the data outputs. If the safety mechanism detects a difference in the SPU0/1 control outputs, the safety mechanism triggers an alarm.

Init Conditions

SMC[SW]:SPU:REDUNDANCY

Runtime conditions

Tests

ESM[SW]:SPU:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[23] - Lockstep Comparator Alarm

6.469 SM[HW]:SPU:REDUNDANCY

Description

The safety mechanism monitors *SPU* random hardware faults e.g. in a dual-SPU 4-antennae configuration. The safety mechanism uses the two functional SPU instances in a redundant mode to compare control and data outputs. Any failure in the comparison of SPU0/1 outputs forwards an alarm to the *SMU*. If the safety mechanism detects a difference in the SPU0/1 outputs, the safety mechanism triggers an alarm.

Init Conditions

SMC[SW]:SPU:REDUNDANCY

Runtime conditions

Tests

ESM[SW]:SPU:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[23] - Lockstep Comparator Alarm

6.470 SM[HW]:SPU:REDUNDANCY_SCC

Description

A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. A special unit detects that the fault has been propagated to the correct comparator node. If such is not the case either the comparator has a defect or a real fault took place and a lockstep alarm is generated.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM9[23] - SPU Lockstep comparator alarm

6.471 SM[HW]:SPU:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.472 SM[HW]:SPU:SM_CHECK

Description

The safety mechanism monitors the integrity of other *SPU* safety mechanisms. Any detected failure in the safety mechanisms will cause a status bit to be set in SMSTAT register. The safety mechanism operates by maintaining a duplicate copy of all the monitored logic ("active" logic). The duplicate copy has the logic states of the registers inverted.

An alarm is generated if the "active" copy of the logic detects an error

The status bit in SMSTAT is set if the state of the duplicate logic differs from the state of the "active" logic.
The expanded truth table is as follows:

1. there is an error and both hardware monitoring have the same result -> SPU SMU Alarm, bit is not set SMSTAT.SMCTRLSTS
2. there is an error but both hardware monitoring have a different result -> SPU SMU Alarm + bit set in SMSTAT.SMCTRLSTS
3. there is no error and both hardware monitoring have the same result -> no Alarm, bit is not set SMSTAT.SMCTRLSTS
4. there is no error but both hardware monitoring have a different result -> no alarm, only SMSTAT. SMCTRLSTS bit set

The safety mechanism is used to monitor the following SPU safety mechanisms:
SM[HW]:SPU:ACTIVE_CFG_CHECK, SM[HW]:SPU:BYPASS_CRC, SM[HW]:SPU:CONTROL_FLOW_SIGNATURE, SM[HW]:RIF:SPU_INTERFACE.

Init Conditions

Runtime conditions

ESM[SW]:SPU:SM_CHECK

Tests

ESM[SW]:SPU:SM_CHECK

Fault identification interfaces

Alarm Interface	SM Flag
Application SW Error Handler	N.A.

6.473 SM[HW]:SRI:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event

6.474 SM[HW]:SRI:ERROR_HANDLING

Description

If the *SRI* detects an SRI protocol error or an SRI transaction ID error, the SRI triggers an alarm and an interrupt service request. The SRI stores the address and control information of the SRI transaction. For test, the *Application SW* may trigger the alarm (set SRI_PECONx.SETPE = 1).

After reset, the reporting of SRI protocol errors (SRI_PECONx.PEEN) and SRI transaction ID errors (SRI_TIDEN.ENMCI or SRI_TIDEN.ENSCI) is enabled. The Application SW may disable the reporting.

Init Conditions

Runtime conditions

ESM[SW]:SRI:ERROR_HANDLING

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event
Interrupt Service Request	N.A.

6.475 SM[HW]:SRI:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event

6.476 SM[HW]:SRI:SRI_ADDRESS_MAP_MONITORING

Description

A slave bus agent provides the capability to decode if an incoming bus transaction address belongs to the agent address space. If the slave agent does not match incoming address with the slaves address space, the safety mechanism generates an alarm to *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event

6.477 SM[HW]:SRI:SRI_TRANSACTION_INTEGRITY

Description

If the *SRI* interface detects an integrity error during the address or data phases of an SRI transaction, the SRI interface triggers an alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[2] - (Xbar_SRI0, Xbar_SRI1, Xbar_SRI2) SRI Address Phase Error
SMU Alarm	ALM11[3] - (Xbar_SRI0, Xbar_SRI1, Xbar_SRI2) SRI Write Data Phase Error
SMU Alarm	ALM11[6] - (SFI_BBB) SRI Address Phase Error
SMU Alarm	ALM11[7] - (SFI_BBB) SRI Write Data Phase Error
SMU Alarm	ALM11[9] - (SFI_SPB) SRI Read Data Phase Error
SMU Alarm	ALM11[13] - (CPU SOTA, SRI SOTA) SOTA Swap Error
SMU Alarm	ALM11[4] - (DMU) EDC Address Phase Error
SMU Alarm	ALM11[5] - (DMU) EDC Write Phase Error
SMU Alarm	ALM11[8] - (LMU) EDC Read Phase Error
SMU Alarm	ALM11[10] - (HSSL0) EDC Read Phase Error
SMU Alarm	ALM11[11] - (HSSL1) EDC Read Phase Error
SMU Alarm	ALM11[12] - (CONVERTER) Phase Synchronizer Error

6.478 SM[HW]:SRI:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[17] - SRI0 Bus Error Event
SMU Alarm	ALM7[18] - SRI1 Bus Error Event
SMU Alarm	ALM7[19] - SRI2 Bus Error Event

6.479 SM[HW]:SRI:SWAP_CONFIGURATION_PROTECTION

Description

The safety mechanism checks for unintentional SWAP configuration change. For an unintentional change an alarm is generated for the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM11[13] - SRI0 SOTA swap error
SMU Alarm	ALM11[13] - SRI1 SOTA swap error
SMU Alarm	ALM11[13] - SRI2 SOTA swap error

6.480 SM[HW]:STM:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.481 SM[HW]:STM:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.482 SM[HW]:STM:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.483 SM[HW]:STM:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.484 SM[HW]:TRACE.TRAM:ADDRESS

Description

This safety mechanism monitors the address generation logic. It detects failures due to wrong address generation or decoding during both read and write operations. This event is detected as address error by ECCD.UCERR and an uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.485 SM[HW]:TRACE.TRAM:CONTROL

Description

This safety mechanism detects several errors in the surrounding logic of the SRAM:

- SSH has been enabled. Functional access to SRAM is disabled,
- Auto data initialization or partial erase function has been triggered. Part or whole of the SRAM may be overwritten,
- Unexpected triggering of MBIST FSM, or Test access to SRAM,
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational error by FAULTSTS.OPERR and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.486 SM[HW]:TRACE.TRAM:DATA

Description

Each SRAM implements a *DED ECC* logic. This mechanism detects *SBE* and *DBE* during read operations. In case an DBE is detected or ECE is disabled an SBE is detected a critical uncorrectable error alarm is forwarded to the *SMU*.

Additionally these errors in the surrounding logic of the SRAM can be covered:

- Auto-data-init or Partial-erase has been triggered. Part or whole of the SRAM may be overwritten.

- A error has been detected in aregister by safety Flip-Flops in one of the registers in the SSH leading to potential data corruption.
- Unexpected triggering of Test features in the data path, leading to potential data corruption.

These events are detected as critical operational errors by FAULTSTS.OPERR and a a critical uncorrectable error alarm is forwarded to the SMU.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.487 SM[HW]:TRACE.TRAM:ERROR_CORRECTION

Description

The SRAM implements a *SEC ECC* logic. This mechanism detects and correct *SBE* during read operations. In case an SBE is corrected a correctable error alarm is be forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[0] - Single bit error correction

6.488 SM[HW]:TRACE.TRAM:ERROR_MANAGEMENT

Description

Each SRAM has 5 registers (ETRR0..4) that store the memory address in case a correctable or uncorrectable error is detected in a memory location. ETRR0 always contains the address of the last error detected. For each ETRR register, the error type is stored in the corresponding ERRINFO register. In case an error is detected while all ETRR registers are full, an overflow error is generated by ECCD.EOV and a critical uncorrectable error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.489 SM[HW]:TRACE.TRAM:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[1] - Uncorrectable critical error detection

6.490 SM[HW]:TRACE.TRAM:REG_MONITOR_TEST

Description

The safety mechanism implements a self-test mechanism for the SFFs used in functional blocks and allows to detect Latent Faults in SFFs.

Init Conditions

Runtime conditions

Tests

ESM[SW]:TRACE.RAM:REG_MONITOR_TEST

Fault identification interfaces

Alarm Interface	SM Flag
A pplication SW Error Handler	N.A.

6.491 SM[HW]:TRACE.TRAM:SM_CONTROL

Description

This safety mechanism monitors the status of the alarms and other safety mechanisms and detects the following events:

- Any of the alarm notifications of UCENE or CENE got disabled.
- Any of the safety mechanisms got disabled (ECE, TRE, SBE, DBE, ADDRE, OVFE, OPENE).

A detection of these events is detected by FAULTSTS.MISCERR and a miscellaneous error alarm is forwarded to the *SMU*.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[2] - Miscellaneous error detection

6.492 SM[HW]:TRACE:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.493 SM[HW]:TRACE:FFI_CONTROL

Description

Local trace control signals are gated with the global trace enable control signal to reduce single- and multi-point failure rates.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
No Alarm and no Status	N.A.

6.494 SM[HW]:TRACE:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.495 SM[HW]:TRACE:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[21] - BBB Bus Error Event

6.496 SM[HW]:VMT:CFG_AS_AP

Description

The safety mechanism checks if an individual master tag identifier is enabled to access an SFR in the address space Segment F. With this safety mechanism all the registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause an Alarm.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - Built-in SPB Error Detection

6.497 SM[HW]:VMT:ENDINIT

Description

MCU End of Initialization (*ENDINIT*) Protection. There are a number of SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such registers are protected by the ENDINIT mechanism. Any attempt to write an ENDINIT protected SFR without deactivation the protection is discarded, the SFR is not changed, and an alarm is generated. When the ENDINIT protection is deactivated, all the ENDINIT-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.498 SM[HW]:VMT:FPI_WRITE_MONITOR

Description

FPI slave interface safety mechanism detects unintended insertion of data update to configuration register. The detected failure are reported to *SMU* via Alarm interface.

Init Conditions

Runtime conditions

Tests

ESM[SW]:SMU:REG_MONITOR_TEST

Fault identification interfaces

VD Assignment	VA_Rationale
SMU Alarm	ALM6[0] - MTU Safety FF uncorrectable error detected

6.499 SM[HW]:VMT:MBIST

Description

MBIST is a safety mechanism which detects structural problems and checks SRAM to detect permanent random hardware faults in memory cells and memory controller.If MBIST is not run permanent random fault could remain latent in memory cells and memory controller.

Init Conditions

SMC[SW]:VMT:MBIST

Runtime conditions

ESM[SW]:VMT:MBIST

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	Single bit error correction: ALMx[6], ALMx[9], ALMx[12] - (x=0..5) ALM6[10], ALM6[13], ALM6[16], ALM6[19], ALM7[0], ALM7[3], ALM7[6
SMU Alarm	Uncorrectable critical error detection ALMx[4], ALMx[7], ALMx[10], ALMx[13] - (x=0..5) ALM6[11], ALM6[14], ALM6[17], ALM6[20], ALM7[1], ALM7[4], ALM7[7]

6.500 SM[HW]:VMT:REG_MONITOR

Description

The safety mechanism detects transient faults in sequential logic in the related functional block. On failure detection the safety mechanism reports the event to *SMU* via an internal alarm.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM6[0] - MTU Safety FF uncorrectable error detected
SMU Alarm	ALM10[20] - CCU Safety FF correctable error detected
SMU Alarm	ALM10[21] - Miscellaneous Safety FF uncorrectable error detected

6.501 SM[HW]:VMT:SAFETY_ENDINIT

Description

MCU End of Safe Initialization (Safety ENDINIT) Protection. There are a number of *MCU* SFRs that are usually programmed only once during the initialization sequence of the system or application. Modification of these SFRs during a normal application run can have a severe impact on the overall operation of the FB. Such SFRs are protected by the [Safety ENDINIT] mechanism. Any attempt to write a [Safety ENDINIT] protected SFR without deactivating the protection is discarded, the SFR is not changed, and an alarm is generated. When the [Safety ENDINIT] protection is deactivated, all the [Safety ENDINIT]-protected registers can be written to.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.502 SM[HW]:VMT:SV_AP

Description

Supervisor Privilege Level Monitoring. If a slave interface receives a write access, the safety mechanism checks if the write access has supervisor privilege. Critical module registers implement an access permission that limits the accesses to software executing in supervisor mode or to *DMA* channels with the supervisor privilege level only.

Note: Alarms or Interrupts from Safety Mechanisms which control Software Failures (e.g. access protection, MPU) shall not be ignored, but treated according to application needs. If application is not intended to violate access protection, a reasonable Alarm or Interrupt reaction could be application reset to control the software causing the violation. One violation per driving cycle shall be assumed as normal behaviour (caused by soft

errors or unintended access protection violations). Alternatively, the application could execute cyclic (once per DTI) software checks to verify that the configured safety mechanisms are effective and functional.

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
SMU Alarm	ALM7[20] - SPB Bus Error Event

6.503 SM[SW]:CPU:SBST

Detection and Control	lockstep CPU	non-lockstep CPU
Single Point Fault Detection	N.A.	Yes
Latent Fault Detection	N.A.	N.A.
Freedom from Interference	N.A.	N.A.

Description

The *CPU* Software Based Self Test (SBST) detects random hardware faults in a non-lockstep CPU. SBST should be executed cyclically, at least once in a diagnostic test interval. SBST is a non-destructive test designed to preserve the state of the CPU, therefore SBST can be used cyclically during runtime. SBST is a measure against permanent faults to support the single point fault metric.

Init Conditions

Runtime conditions

ESM[SW]:CPU:SBST

Tests

Fault identification interfaces

Alarm Interface	SM Flag
N.A.	N.A.

6.504 SM[SW]:FW:MCU_STARTUP_PREOS_FW

Description

The target of here defined SM is to detect *MCU* configuration being wrong due to random HW failures during device boot phase. For this purpose, two FW components are diversely implemented, namely:

1) Start-up Software *SSW* – installing device registers according to configuration data from *NVM* Configuration sector
2) Checker Software *CHSW* – executing checks of the devices registers installed by SSW against reference data from NVM Configuration sector

Init Conditions

Runtime conditions

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Application SW Error Handler	

6.505 SM[SW]:SPU:SBST

Description

The *SPU* Software Based Self Test (SBST) detects random hardware faults in a SPU. SBST should be executed cyclically, at least once in a diagnostic test interval. SBST is a measure against permanent faults to support the single point fault metric.

Init Conditions

Runtime conditions

ESM[SW]:SPU:SBST

Tests

Fault identification interfaces

Alarm Interface	SM Flag
Application SW Error Handler	N.A.

7 Appendix A

7.1 Appendix A: Alarms tables after Reset

TC39B - COLD POWER-ON RESET AFTER LBIST EXECUTION

	Cold Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x2490	
SMU_AG4	0x2490	
SMU_AG5	0x2490	
SMU_AG6	0x124800	
SMU_AG7	0x92	bit 0 shall be masked to 0 (see errata MTU TC.H015) bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC39B appx	
SCU_STMEM4	see TC39B appx	
SCU_STMEM5	see TC39B appx	
SCU_STMEM6	see TC39B appx	
SCU_LCLCON0	0x80018001	
SCU_LCLCON1	0x80018001	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby: SSH (4, 9) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH (77,78) ECCD = 0x0

FAULTSTS	0x1	after Cold PORST: SSH (40, 77, 78) FAULTSTS = 0x9 SSH (44, 47) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) after Standby: SSH (40) FAULTSTS = 0x9 SSH (77,78) FAULTSTS = 0x0 SSH (44, 47) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B)
ERRINFO[0]	0x0	

TC39B - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
ASMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x2490	
SMU_AG4	0x2490	
SMU_AG5	0x2490	
SMU_AG6	0x124800	
SMU_AG7	0x92	bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC39B appx	
SCU_STMEM4	see TC39B appx	
SCU_STMEM5	see TC39B appx	
SCU_STMEM6	see TC39B appx	
SCU_LCLCON0	0x80018001	
SCU_LCLCON1	0x80018001	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0X9 SSH(44,47) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC39B - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x2490	
SMU_AG4	0x2490	
SMU_AG5	0x2490	
SMU_AG6	0x124800	
SMU_AG7	0x92	
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC39B appx	
SCU_STMEM4	see TC39B appx	
SCU_STMEM5	see TC39B appx	
SCU_STMEM6	see TC39B appx	
SCU_LCLCON0	0x80018001	

SCU_LCLCON1	0x80018001	

For all RAM:

	System Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD = 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0x9 SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC39B- APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x400
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC39B appx
SCU_STMEM4	see TC39B appx
SCU_STMEM5	see TC39B appx
SCU_STMEM6	see TC39B appx

For the following RAM (SSH=0, 34, 35):

	Application Reset Value
ECCD	0x5
FAULTSTS	0x1
ERRINFO[0]	0x0

TC38A - COLD POWER-ON RESET AFTER LBIST EXECUTION

	Cold Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x2490	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	bit 0 shall be masked to 0 (see errata MTU TC.H015)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC38A appx	
SCU_STMEM4	see TC38A appx	
SCU_STMEM5	see TC38A appx	
SCU_STMEM6	see TC38A appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby: SSH (4, 9) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	after cold PORST: SSH(40,77,78) FAULTSTS = 0x9 after Standby: SSH(40) FAULTSTS = 0x9 SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC38A - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x2490	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC38A appx	
SCU_STMEM4	see TC38A appx	
SCU_STMEM5	see TC38A appx	
SCU_STMEM6	see TC38A appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS= 0x9 SSH(77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC38A - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x2490	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC38A appx	
SCU_STMEM4	see TC38A appx	
SCU_STMEM5	see TC38A appx	
SCU_STMEM6	see TC38A appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all RAM:

	System Reset Value	
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH (40) FAULTSTS = 0x9 SSH (77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC38A - APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x400
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0

SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC38A appx
SCU_STMEM4	see TC38A appx
SCU_STMEM5	see TC38A appx
SCU_STMEM6	see TC38A appx

For the following RAM (SSH=0, 34, 35):

SSH instance	Application Reset Value
ECCD	0x5
FAULTSTS	0x1
ERRINFO[0]	0x0

TC35A - COLD POWER-ON RESET AFTER LBIST EXECUTION

	Cold Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124000	
SMU_AG7	0x92	bit 0 shall be masked to 0 (see errata MTU TC.H015) bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC35A appx	
SCU_STMEM4	see TC35A appx	
SCU_STMEM5	see TC35A appx	
SCU_STMEM6	see TC35A appx	

SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby: SSH (4, 9) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH (77,78) ECCD = 0x0
FAULTSTS	0x1	after Cold PORST: SSH (40, 77, 78) FAULTSTS = 0x9 SSH (44) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) after Standby: SSH (40) FAULTSTS = 0x9 SSH (77,78) FAULTSTS = 0x0 SSH (44) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B)
ERRINFO[0]	0x0	

TC35A - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124000	
SMU_AG7	0x92	bit 14 shall be masked to 0 (see errata BROM TC.H011)

SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC35A appx	
SCU_STMEM4	see TC35A appx	
SCU_STMEM5	see TC35A appx	
SCU_STMEM6	see TC35A appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0X9 SSH(44) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC35A - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	

SMU_AG6	0x124000	
SMU_AG7	0x92	
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC35A appx	
SCU_STMEM4	see TC35A appx	
SCU_STMEM5	see TC35A appx	
SCU_STMEM6	see TC35A appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all RAM:

	System Reset Value	
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH (40) FAULTSTS = 0x9 SSH (77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC35A- APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x400
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC35A appx
SCU_STMEM4	see TC35A appx

SCU_STMEM5	see TC35A appx
SCU_STMEM6	see TC35A appx

For the following RAM (SSH=0, 34, 35):

	Application Reset Value
ECCD	0x5
FAULTSTS	0x1
ERRINFO[0]	0x0

TC37xEXT - COLD POWER-ON RESET AFTER LBIST EXECUTION

	Cold Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x92	bit 0 shall be masked to 0 (see errata MTU TC.H015) bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC37xEXT appx	
SCU_STMEM4	see TC37xEXT appx	
SCU_STMEM5	see TC37xEXT appx	
SCU_STMEM6	see TC37xEXT appx	
SCU_LCLCON0	0x80018001	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby: SSH (4, 9) FAULTSTS = 0x1

ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH (77,78) ECCD = 0x0
FAULTSTS	0x1	after Cold PORST: SSH (40, 77, 78) FAULTSTS = 0x9 SSH (44, 46) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) after Standby: SSH (40) FAULTSTS = 0x9 SSH (77,78) FAULTSTS = 0x0 SSH (44, 46) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B)
ERRINFO[0]	0x0	

TC37xEXT - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x92	bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC37xEXT appx	
SCU_STMEM4	see TC37xEXT appx	

SCU_STMEM5	see TC37xEXT appx	
SCU_STMEM6	see TC37xEXT appx	
SCU_LCLCON0	0x80018001	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0X9 SSH(44, 46) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC37xEXT - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x92	
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	

SCU_STMEM3	see TC37xEXT appx	
SCU_STMEM4	see TC37xEXT appx	
SCU_STMEM5	see TC37xEXT appx	
SCU_STMEM6	see TC37xEXT appx	
SCU_LCLCON0	0x80018001	
SCU_LCLCON1	0x80010000	

For all RAM:

	System Reset Value	
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH (40) FAULTSTS = 0x9 SSH (77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC37xEXT- APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x400
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC37xEXT appx
SCU_STMEM4	see TC37xEXT appx
SCU_STMEM5	see TC37xEXT appx
SCU_STMEM6	see TC37xEXT appx

For the following RAM (SSH=0, 34, 35):

	Application Reset Value
ECCD	0x5
FAULTSTS	0x1

ERRINFO[0]	0x0

TC37A - COLD POWER-ON RESET AFTER LBIST EXECUTION

SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	bit 0 shall be masked to 0 (see errata MTU TC.H015)
SMU_AG8	0x100020	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC37x appx	
SCU_STMEM4	see TC37x appx	
SCU_STMEM5	see TC37x appx	
SCU_STMEM6	see TC37x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby:
		SSH (4, 9) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH (77,78) ECCD = 0x0

FAULTSTS	0x1	after Cold PORST: SSH (40, 77, 78) FAULTSTS = 0x9 after Standby: SSH (40) FAULTSTS = 0x9 SSH (77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC37A - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	
SMU_AG8	0x100000	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC37x appx	
SCU_STMEM4	see TC37x appx	
SCU_STMEM5	see TC37x appx	
SCU_STMEM6	see TC37x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0

FAULTSTS	0x1	SSH(40) FAULTSTS = 0X9 SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC37A - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x2490	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	
SMU_AG8	0x100000	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC37x appx	
SCU_STMEM4	see TC37x appx	
SCU_STMEM5	see TC37x appx	
SCU_STMEM6	see TC37x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all RAM:

	System Reset Value	Note
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH (40) FAULTSTS = 0x9 SSH (77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC37A- APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400

SMU_AG1	0x400
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x100000
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC37x appx
SCU_STMEM4	see TC37x appx
SCU_STMEM5	see TC37x appx
SCU_STMEM6	see TC37x appx

For all RAM:

	Application Reset Value	Note
ECCD	0x0	SSH(0, 34, 35) ECCD = 0x5
FAULTSTS	0x0	SSH (0, 34, 35) FAULTSTS = 0x1
ERRINFO[0]	0x0	

TC36A - COLD POWER-ON RESET AFTER LBIST EXECUTION

SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	bit 0 shall be masked to 0 (see errata MTU TC.H015)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002

SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC36x appx	
SCU_STMEM4	see TC36x appx	
SCU_STMEM5	see TC36x appx	
SCU_STMEM6	see TC36x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby: SSH (4, 9) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH (77,78) ECCD = 0x0
FAULTSTS	0x1	after Cold PORST: SSH (40, 77, 78) FAULTSTS = 0x9 after Standby: SSH (40) FAULTSTS = 0x9 SSH (77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC36A - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	

SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC36x appx	
SCU_STMEM4	see TC36x appx	
SCU_STMEM5	see TC36x appx	
SCU_STMEM6	see TC36x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0X9 SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC36A - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124800	
SMU_AG7	0x2	
SMU_AG8	0x0	

SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC36x appx	
SCU_STMEM4	see TC36x appx	
SCU_STMEM5	see TC36x appx	
SCU_STMEM6	see TC36x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x80010000	

For all RAM:

	System Reset Value	Note
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH (40) FAULTSTS = 0x9 SSH (77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC36A- APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x400
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC36x appx
SCU_STMEM4	see TC36x appx
SCU_STMEM5	see TC36x appx
SCU_STMEM6	see TC36x appx

For all RAM:

	Application Reset Value	Note
ECCD	0x0	SSH(0, 34, 35) ECCD = 0x5
FAULTSTS	0x0	SSH (0, 34, 35) FAULTSTS = 0x1
ERRINFO[0]	0x0	

TC33xEXT - COLD POWER-ON RESET AFTER LBIST EXECUTION

SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x120000	
SMU_AG7	0x92	bit 0 shall be masked to 0 (see errata MTU TC.H015) bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC33xEXT appx	
SCU_STMEM4	see TC33xEXT appx	
SCU_STMEM5	see TC33xEXT appx	
SCU_STMEM6	see TC33xEXT appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x0	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby: SSH (4, 9) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH (77,78) ECCD = 0x0
FAULTSTS	0x1	after Cold PORST: SSH (40, 77, 78) FAULTSTS = 0x9 SSH (44) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) after Standby: SSH (40) FAULTSTS = 0x9 SSH (77,78) FAULTSTS = 0x0 SSH (44) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B)
ERRINFO[0]	0x0	

TC33xEXT - WARM POWER-ON RESET

	Warm Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x120000	
SMU_AG7	0x92	bit 14 shall be masked to 0 (see errata BROM TC.H011)
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC33xEXT appx	
SCU_STMEM4	see TC33xEXT appx	
SCU_STMEM5	see TC33xEXT appx	
SCU_STMEM6	see TC33xEXT appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x0	

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0x9 SSH (44) FAULTSTS = 0x601 : when global Flash read protection is disabled (HF_PROCONPF.RPRO=0_B and HF_PROCONDF.RPRO=0_B) SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC33xEXT - SYSTEM RESET

	System Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x2490	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x120000	
SMU_AG7	0x92	
SMU_AG8	0x0	
SMU_AG9	0x0	
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC33xEXT appx	
SCU_STMEM4	see TC33xEXT appx	
SCU_STMEM5	see TC33xEXT appx	
SCU_STMEM6	see TC33xEXT appx	
SCU_LCLCON0	0x80010000	

SCU_LCLCON1	0x0	

For all RAM:

	System Reset Value	Note
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH (40) FAULTSTS = 0x9 SSH (77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC33xEXT - APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x0
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC33xEXT appx
SCU_STMEM4	see TC33xEXT appx
SCU_STMEM5	see TC33xEXT appx
SCU_STMEM6	see TC33xEXT appx

For all RAM:

	Applicatoin Reset Value	Note
ECCD	0x0	SSH(0, 34) ECCD = 0x5
FAULTSTS	0x0	SSH (0, 34) FAULTSTS = 0x1
ERRINFO[0]	0x0	

TC33/TC32A - COLD POWER-ON RESET AFTER LBIST EXECUTION

	Cold Power-on Reset Value	Notes
SMU_AG0	0x2490	
SMU_AG1	0x0	
SMU_AG2	0x0	
SMU_AG3	0x0	
SMU_AG4	0x0	
SMU_AG5	0x0	
SMU_AG6	0x124000	
SMU_AG7	0x2	bit 0 shall be masked to 0 (see errata MTU TC.H015)
SMU_AG8	0x20	
SMU_AG9	0x0	see errata DTS TC.H002
SMU_AG10	0x0	
SMU_AG11	0x0	
SCU_STMEM3	see TC33/TC32x appx	
SCU_STMEM4	see TC33/TC32x appx	
SCU_STMEM5	see TC33/TC32x appx	
SCU_STMEM6	see TC33/TC32x appx	
SCU_LCLCON0	0x80010000	
SCU_LCLCON1	0x0	

For all initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	
FAULTSTS	0x9	after Standby:
		SSH(4) FAULTSTS = 0x1
ERRINFO[0]	0x0	

For all non-initialized RAM:

	Cold Power-on Reset Value	Notes
ECCD	0x5	after Standby: SSH(77,78) ECCD = 0x0
FAULTSTS	0x1	after Cold PORST: SSH(40, 77, 78) FAULTSTS = 0x9

		after Standby: SSH(40) FAULTSTS = 0x9 SSH(77,78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC33/TC32A - WARM POWER-ON RESET

	Warm Power-on Reset Value
SMU_AG0	0x2490
SMU_AG1	0x0
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x124000
SMU_AG7	0x2
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC33/TC32x appx
SCU_STMEM4	see TC33/TC32x appx
SCU_STMEM5	see TC33/TC32x appx
SCU_STMEM6	see TC33/TC32x appx
SCU_LCLCON0	0x80010000
SCU_LCLCON1	0x0

For all initialized RAM:

	Warm Power-on Reset Value
ECCD	0x5
FAULTSTS	0x9
ERRINFO[0]	0x0

For all non-initialized RAM:

	Warm Power-on Reset Value	Notes
ECCD	0x5	SSH(77,78) ECCD= 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0x9 SSH(77,78) FAULTSTS = 0x0

ERRINFO[0]	0x0	

TC33/TC32A - SYSTEM RESET

	System Reset Value
SMU_AG0	0x2490
SMU_AG1	0x0
SMU_AG2	0x0
SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x124000
SMU_AG7	0x2
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC33/TC32x appx
SCU_STMEM4	see TC33/TC32x appx
SCU_STMEM5	see TC33/TC32x appx
SCU_STMEM6	see TC33/TC32x appx
SCU_LCLCON0	0x80010000
SCU_LCLCON1	0x0

For all RAM:

	System Reset Value	Note
ECCD	0x5	SSH(77, 78) ECCD = 0x0
FAULTSTS	0x1	SSH(40) FAULTSTS = 0x9 SSH(77, 78) FAULTSTS = 0x0
ERRINFO[0]	0x0	

TC33/TC32A - APPLICATION RESET

	Application Reset Value
SMU_AG0	0x400
SMU_AG1	0x0
SMU_AG2	0x0

SMU_AG3	0x0
SMU_AG4	0x0
SMU_AG5	0x0
SMU_AG6	0x0
SMU_AG7	0x0
SMU_AG8	0x0
SMU_AG9	0x0
SMU_AG10	0x0
SMU_AG11	0x0
SCU_STMEM3	see TC33/TC32x appx
SCU_STMEM4	see TC33/TC32x appx
SCU_STMEM5	see TC33/TC32x appx
SCU_STMEM6	see TC33/TC32x appx

For all RAM:

	Application Reset Value	Note
ECCD	0x0	SSH(0, 34) ECCD = 0x5
FAULTSTS	0x0	SSH(0, 34) FAULTSTS = 0x1
ERRINFO[0]	0x0	

8 Appendix B

8.1 Appendix B: Pseudo Code and Implementation Example for ESM[SW]:CPU:INTERNAL_BUS_MONITOR

PSEUDO CODE

The following pseudo-code illustrates the principle of the ESM[SW]:CPU:INTERNAL_BUS_MONITOR. This ESM should be implemented as a sub-routine and should make use of the CPU's upper context registers The pseudo-code is written using a mixture of macros and real Tricore CPU instructions. There are two aspects to this ESM: 1) Checking the internal internal fifo and bus interface (FIFO TEST). 2) Checking the interface to the DLMU. The suggested pseudo-code is not a real code example. An actual implementation example (in C) of the internal FIFO test is given in the following section. To fully exercise the FIFO, it is mandatory that the context store immediately follows a misaligned double store to the same area. This is done by the CONTEXT_WRITE macro.

```
MACROs:

LDA .MACRO ax,value

movh.a ax,#@his(value) ;high part with correction for signed addition

lea ax,[ax]@los(value)

.ENDM LDD .MACRO dx,value

mov.u dx,#@lo(value)

addih dx,dx,#@hi(value)

.ENDM CONTEXT_WRITE .MACRO

LDA a15, __SBST_CPU_PSPR_BASE_ADDR

st.d [a15]+4, e8;

stucx [a15]

.ENDM CONTEXT_LOAD .MACRO

LDA a15, __SBST_CPU_PSPR_BASE_ADDR

dsync

lducx [a15]

.ENDM;; All other macros are self-explanatory.


;______________________

; Pseudo-code for FIFO test:

DISABLE_INTERRUPTS_AND_SAVE_IE;

LOAD_D8_TO_D15  0xAAAAAAAA

LOAD_A12_TO_A14    0xAAAAAAAA

SAVE_PCX_VALUE_TO_TEMP_PCX;

SAVE_PSW_VALUE_TO_TEMP_PSW;

CONTEXT_WRITE

LOAD_D8_TO_D15  0x00000000
```

```
LOAD_A12_TO_A14    0x00000000

CONTEXT_LOAD

CHECK_D8_TO_D15_AND_A12_TO_A14_MATCH_PATTERN  0xAAAAAAAA

CHECK_PCX_VALUE_MATCHES_TEMP_PCX;

CHECK_PSW_VALUE_MATCHES_TEMP_PSW;

RESTORE_INTERRUPTS

; Repeat with pattern 0x55555555


;______________________

; Pseudo-code for DLMU test:

dsync

; Store pattern to DLMU location

LDA a15, __DLMU_ESM_BASE

LDD d14, 0xAAAAAAAA

LDD d15, 0xAAAAAAAA

st.d [a15], e14

; Read back and check

dsync

ld.d e12, [a15]

CHECK_D12_and_D13_MATCH_PATTERN 0xAAAAAAAA

; Repeat with pattern 0x55555555
```

IMPLEMENTATION EXAMPLE The following piece of code is a real software example for the FIFO TEST.
The assembler (GCC syntax) for the test program is embedded in a C function and follows the same principles as the previous pseudocode. Note that this does not cover the DLMU part of the ESM.

```
/***************************************************************************

*

* Copyright (C) 2018-19 Infineon Technologies AG. All rights reserved.

*

* Infineon Technologies AG (INFINEON) is supplying this file for use

* exclusively with Infineon's microcontroller products. This file can be freely

* distributed within development tools and software supporting such
microcontroller

* products.

*

* THIS SOFTWARE IS PROVIDED "AS IS".  NO WARRANTIES, WHETHER EXPRESS, IMPLIED

* OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF

* MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE.

* INFINEON SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR DIRECT, INDIRECT,
INCIDENTAL,

* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.

*

***************************************************************************/

boolean nonLS_CPU_internal_bus_test (void* csa_area, uint32 pattern)
{
boolean result; // fail = 0, success = 1
uint32 ie_prev; // previous ICR.IE state
uint64 tmp64;
uint32 tmp_pcxi;
uint32 tmp_psw;
void * tmp_a11;
__asm__ ( "disable %0" : "=d"(ie_prev) ); /* Disable Interrupts; ie_prev
contains previous interrupt enable state */
__asm__ __volatile__ ("dsync");      /* Synchronize Data - memory coherency =>
forces all Cached Data to be written to memory */
/**
Global List
------------
0 -> result
```

```
1 -> tmp64
2 -> tmp_pcxi
3 -> tmp_psw
4 -> tmp_a11
5 -> pattern
6 -> csa_area
**/
 __asm__ __volatile__ (
"mov   %L1,   %5  \n\t"  /* move "pattern" to lower 32 bits of "tmp64"*/

"mov.aa %4, %%a11 \n\t"  /* move A11 to "tmp_a11" -> copy for future
comparison*/

"mov   %H1,   %5  \n\t"  /* move "pattern" to higher 32 bits of "tmp64"*/

"mfcr %2, $pcxi   \n\t"  /* move PCXI to "tmp_pcxi" -> copy for future
comparison*/

"mov   %%d8,  %5  \n\t"  /* move "pattern" to D8*/

"mfcr %3, $psw    \n\t"  /* move PSW to "tmp_psw" -> copy for future
comparison*/

"mov   %%d9,  %5  \n\t"  /* move "pattern" to D9*/

"mov.a %%a10, %5  \n\t"  /* move "pattern" to A10*/

"mov   %%d10, %5  \n\t"  /* move "pattern" to D10*/

"mov   %%d11, %5  \n\t"  /* move "pattern" to D11*/

"mov.a %%a12, %5  \n\t"  /* move "pattern" to A12*/

"mov   %%d12, %5  \n\t"  /* move "pattern" to D12*/

"mov.a %%a13, %5  \n\t"  /* move "pattern" to A13*/

"mov   %%d13, %5  \n\t"  /* move "pattern" to D13*/

"mov.a %%a14, %5  \n\t"  /* move "pattern" to A14*/

"mov   %%d14, %5  \n\t"  /* move "pattern" to D14*/

"mov.a %%a15, %5  \n\t"  /* move "pattern" to A15*/

"mov   %%d15, %5  \n\t"  /* move "pattern" to D15*/

"st.d  [%6]0x44, %A1 \n\t" /* store "tmp64" to "csa_area"+0x44 => offset 68*/

"stucx [%6]0x0       \n\t" /* store Upper Context (A10-A15, D8-D15, PSW, PCXI)
to  "csa_area"+0x00 => 64 bytes*/

"mov   %L1,   0 \n\t" /* zero lower 32 bits of "tmp64"*/
```

```
"mov   %H1,   0 \n\t" /* zero higher 32 bits of "tmp64"*/

"mov   %%d8,  0 \n\t" /* zero contents of D8"*/

"mov   %%d9,  0 \n\t" /* zero contents of D9*/

"mov.a %%a10, 0 \n\t" /* zero contents of A10*/

"mov   %%d10, 0 \n\t" /* zero contents of D10*/

"mov   %%d11, 0 \n\t" /* zero contents of D11*/

"mov.a %%a12, 0 \n\t" /* zero contents of A12*/

"mov   %%d12, 0 \n\t" /* zero contents of D12*/

"mov.a %%a13, 0 \n\t" /* zero contents of A13*/

"mov   %%d13, 0 \n\t" /* zero contents of D13*/

"mov.a %%a14, 0 \n\t" /* zero contents of A14*/

"mov   %%d14, 0 \n\t" /* zero contents of D14*/

"mov.a %%a15, 0 \n\t" /* zero contents of A15*/

"mov   %%d15, 0 \n\t" /* zero contents of D15*/

"dsync \n\t"  /* Synchronize Data - memory coherency => forces all Cached Data
to be written to memory */

"ld.d %A1, [%6]0x44 \n\t" /* load DOUBLE-WORD "tmp64" from "csa_area"+0x44 =>
offset 68*/

"ldducx     [%6]0x0  \n\t" /* load Upper Context (A10-A15, D8-D15, PSW, PCXI)
from  "csa_area"+0x00*/

"eq.a   %0, %4, %%a11 \n\t" /* "result" = (A11 == "tmp_a11") */

"and.eq %0, %5, %L1   \n\t" /* "result" &= (lower 32 bits of "tmp64" ==
"pattern") */

"and.eq %0, %5, %H1   \n\t" /* "result" &= (higher 32 bits of "tmp64" ==
"pattern") */

"and.eq %0, %5, %%d8  \n\t" /* "result" &= (D8 == "pattern") */

"and.eq %0, %5, %%d9  \n\t" /* "result" &= (D9 == "pattern") */

"and.eq %0, %5, %%d10 \n\t" /* "result" &= (D10 == "pattern") */

"mov.d  %%d10, %%a10 \n\t" /* move A10 to D10 for comparison" */
```

```
"and.eq %0, %5, %%d11 \n\t" /* "result" &= (D11 == "pattern" */

"and.eq %0, %5, %%d12 \n\t" /* "result" &= (D12 == "pattern") */

"mov.d  %%d12, %%a12 \n\t" /* move A12 to D12 for comparison" */

"and.eq %0, %5, %%d13 \n\t" /* "result" &= (D13 == "pattern") */

"mov.d  %%d13, %%a13 \n\t" /* move A13 to D13 for comparison" */

"and.eq %0, %5, %%d14 \n\t" /* "result" &= (D14 == "pattern") */

"mov.d  %%d14, %%a14 \n\t" /* move A14 to D14 for comparison" */

"and.eq %0, %5, %%d15 \n\t" /* "result" &= (D15 == "pattern") */

"mov.d  %%d15, %%a15 \n\t" /* move A15 to D15 for comparison" */

"and.eq %0, %5, %%d10 \n\t" /* "result" &= (D10(actually A10) == "pattern") */

"and.eq %0, %5, %%d12 \n\t" /* "result" &= (D12(actually A12) == "pattern") */

"and.eq %0, %5, %%d13 \n\t" /* "result" &= (D13(actually A13) == "pattern") */

"and.eq %0, %5, %%d14 \n\t" /* "result" &= (D14(actually A14) == "pattern") */

"and.eq %0, %5, %%d15 \n\t" /* "result" &= (D15(actually A15) == "pattern") */

"ld.d %A1, [%6]0x0 \n\t" /* load DOUBLE-WORD "tmp64" from "csa_area"+0x0 =>
offset 0 (PCXI, PSW)*/

"and.eq %0, %L1, %2 \n\t" /* "result" &= (lower 32 bits of "tmp64" ==
"tmp_pcxi") */

"and.eq %0, %H1, %3 \n\t" /* "result" &= (higher 32 bits of "tmp64" ==
"tmp_psw") */

/* Global List */

: "=&d"(result), "=&d"(tmp64), "=&d"(tmp_pcxi), "=&d"(tmp_psw), "=&a"(tmp_a11)

: "d"(pattern), "a"(csa_area)

: "d8", "d9", "d10", "d11", "d12", "d13", "d14", "d15", "a12", "a13", "a14",
"a15"

);

__asm__ ( "restore %0" : : "d"(ie_prev) ); /* Restore interrupt state */

return result;
```

```
    }
```

9 Revision History

9.1 v1.03 Revision History

Chapter	Changes related to previous version	Comments
5	ESM[SW]:CPU:SOFTERR_MONITOR changed description.	
6	SM[HW]:DTS:TEMPERATURE_MONITOR, alarms ALM8[30] and ALM8[31] reassigned.	typo correction

9.2 v1.04 Revision History

Chapter	Changes related to previous version	Comments
3	"Boot and startup Procedure" chapter: execution sequence changed.	• Added ESM[SW]:SMU:REG_MONITOR_TEST. • Moved "enable SMU Alarms" at the end of the procedure.
3	"error monitor" chapter: added note to Errata.	
4	"NVM monitoring concept" chapter, added note for NVM error handling.	
4	"SPU monitoring concept" chapter, changed description.	
5	ESM[SW]:SYS:MCU_STARTUP, removed from registers´ list: • EDSADC_CHx_IRMS (x=0..13) • EVADC_VDDKC G(x) (x=EVADC groups) • EVADC_DVDDK G(x) (x=EVADC groups)	The registers contain calibration values which are specific for each device and unknown by System Integrator during SW development.
5	ESM[SW]:CLOCK:PLAUSIBILITY, changed description.	
5	ESM[SW]:PMS:VEXT_VEVRSB_OVERVOLTAGE, changed description.	
5	ESM[SW]:SPU:SBST, changed "recommendations" section.	
5	ESM[SW]:CPU:INTERNAL_BUS_MONITOR, changed description.	
5	ESM[SW]:CPU:BUS_MPU_INITCHECK, changed description.	

5	ESM[SW]:SMU:REG_MONITOR_TEST, changed "notes" section.	
5	SMC[SW]:CLOCK:OSC_MONITOR added.	
6	SM[HW]:IR:CFG_MONITOR, ALM8[22] replaced with ALM10[22].	typo correction
6	SM[HW]:CLOCK:CFG_MONITOR, changed ALM10[20] description.	typo correction
6	SM[HW]:SCU:REG_MONITOR changed ALM10[20] description.	typo correction
6	SM[HW]:VMT:REG_MONITOR changed ALM10[20] description.	typo correction
6	SM[HW]:GTM:TOM_GPT12_MONITORING_WITH_IOM removed.	GPT12 connection to IOM not existing.
7	Appendix A: TC35 tables added.	
7	Appendix A: TC38, TC39 added Errata.	
7	Appendix B added.	

9.3 v1.05 Revision History

Chapter	Changes related to previous version	Comments
Preface	Scope extended to TC37xEXT	
5	ESM[SW]:CPU:AP_CHECK - Typo correction in register name in "Note" section of safety mechanism specification	
5	Update of information regarding test execution time for ESM[SW]:*FB.RAM*:REG_MONITOR_TEST	Here *FB* refers to the functional block name and *RAM* to the SRAM instance name. For example ESM[SW]:CPU.DSPR:REG_MONITOR_TEST
5	Added ESM[SW]:CPU:BUS_MPU_INITCHECK relevant registers located in EMEM, DAM and LMU	
5	Added ATOM as a possible alternative to TOM usage in GTM TIM-TOM monitoring scenario	
5	Update of information regarding test execution time for ESM[SW]:SMU:REG_MONITOR_TEST	
5	Removed device specific information in ESM[SW]:SYS:MCU_FW_CHECK to cover all available devices	
5	Added SMC[SW]:GTM:GTM_CONFIG_FOR_ATOM to provide information on how to configure ATOM channel	

6	Updated SM[HW]:*FB.RAM*:REG_MONITOR_TEST functional description to be consistent with functional description of related ESM[SW]:*FB.RAM*:REG_MONITOR_TEST	
6	SM[HW]:CLOCK:CFG_MONITOR - Typo correction in "Fault indication interface" section of safety mechanism specification	
6	SM[HW]:CLOCK:OSC_MONITOR - Typo correction in "Description" section of safety mechanism specification and missing parenthesis added in frequency calculation formula	
6	SM[HW]:CPU:STI - Added missing Fault Indication Interface information	
6	Added SMC[SW]:GTM:GTM_CONFIG_FOR_ATOM as "Init Condition" for SM[HW]:GTM:TOM_CCU6_MONITORING_WITH_IOM and SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM	
7	Update of expected values for SCU_LCLCON0 and SCU_LCLCON1 registers after cold PORST, warm PORST and system reset	
7	Update of expected value for SMU_AG7 register after system reset	
7	Appendix A: TC37xEXT tables added.	

9.4 v1.06 Revision History

Chapter	Changes related to previous version	Comments
Preface	Added new document version	
4.2.12.1.2	Corrected typo in ESM name. ESM[SW]:SMU:FW_ALARM_CHECK changed into ESM[SW]:SYS:MCU_FW_CHECK	
5.5 & 5.7	Corrected inconsistency in register bitfield. EVRSTAT.UVC and EVRSTAT.OVC corrected into EVRSTAT.UVDDM and EVRSTAT.OVDDM	
6.31, 6.192, 6.220, 6.238, 6.252, 6.300, 6.362, 6.397, 6.417, 6.490	Added "N.A." in SM Flag column of Fault Indication Interfaces table	

6.283	Corrected faulty SM Flag information in Fault Indication interfaces table	

9.5 v1.07 Revision History

Chapter	Changes related to previous version	Comments
Preface	Added new document version	
Intended Audience	Added reference to Safety Case Report Open Problem List	
7	Appendix A: TC37A tables added	
Revision History	Removed Revision History of v1.04	
Revision History	Added a note to clarify an entry in Revision History of v1.05	

9.6 v1.08 Revision History

Chapter	Changes related to previous version	Comments
Preface	Added new document version	
4.2.10.3.2	Added information related to monitoring concept of GPT12	Extension of Application Assumption related to usage of CCU6 as mission logic
4.3.2.6	Added ASIL B Top Level Safety Requirement related Digital Acquisition function and included GPT12 as involved functional block. New Functional Use Cases added involving GPT12 and CCU6	Extension of Application Assumption related to usage of CCU6 as mission logic
4.3.2.7	New Functional Use Cases for Digital Actuation ASIL B added involving GPT12 and CCU6	Extension of Application Assumption related to usage of CCU6 as mission logic
5.11 & 5.12	Added ESM[SW]:CCU6:CCU6_CAPTURE_MON_BY_GPT12 and ESM[SW]:CCU6:CCU6_GPT12_MONITORING safety mechanisms	Extension of Application Assumption related to usage of CCU6 as mission logic

9.7 v1.09 Revision History

Chapter	Changes related to previous version	Comments
Preface	Added new document version. Scope extended to TC36x	
7	Appendix A: TC36A tables added	
7	Appendix A: Typo correction in TC37A tables	The tables were referencing the TC37xEXT appx instead of TC37x appx

9.8 v1.10 Revision History

Chapter	Changes related to previous version	Comments
Preface	Added new document version. Scope extended to TC33xEXT	
5.95	Added reference to STSTAT register	Register was listed in ESM description but reference was missing
7	Appendix A: TC33xEXT tables added	
Revision History	Removed Revision History of v1.06 and of v1.07	

9.9 v1.11 Revision History

Chapter	Changes related to previous version	Comments
Preface	Added new document version. Scope extended to TC33x/TC32xA	
3.2.1	Added list of "Development-Only” features	
4.3.3.3	SM[HW]:*:VM_AP replaced with SM[HW]:*:VM_AS_AP	typo correction
5.1	ESM[HW]:MCU:LBIST_MONITOR modified to refer to the correct register	STMEM0 substituted with HF_PROCONDF
5.1	Modified notes section of ESM[HW]:MCU:LBIST_MONITOR: FW execution influences only ESR0 pin	
5.3	Modified description section of ESM[HW]:PMS:VEXT_VEVRSB_OVERVOLTAGE	
5.42	Modified notes section ESM[SW]:EDSADC:VAREF_PLAUSIBILITY	
5.51	Modified notes section ESM[SW]:EVADC:VAREF_PLAUSIBILITY	
5.95	Modified notes section of ESM[SW]:SYS:MCU_STARTUP	PMSWSTAT.HWCFGEVR must be considered instead of STSTAT.HWCFG for devices in package variants QFP80 and QFP100
5.116	Modified description and references sections of SMC[SW]:SPU:BYPASS_CRC	RIFCRC substituted with BPCRC
6.245	Modified description of SM[HW]:GTM:TOM_TOM_MONITORING_WITH_IOM	
7	Appendix A: removed reserved SSHs (72, 73,74,75 and 76)	
7	Appendix A: SSH (44, 47) FAULTSTS = 0x1 when global Flash read protection is set.	added note
7	Appendix A: TC33xEXT SSH (44) FAULTSTS = 0x1 after System Reset	
7	Appendix A: TC33xEXT Cold PORST table modified: SMU_AG7 bit 14 shall be masked to 0	added note
7	Appendix A: header of TC33xEXT, TC36A and TC37A Application reset tables modified	typo correction

7	Appendix A: TC33x/TC32xA tables added	
9	Added missing Revision History	

Glossary

ADC
ADC
Analog-to-Digital Converter

ADC
ADC
Analog-to-Digital Converter

ADC
ADC
Analog-to-Digital Converter

ADC
ADC
Analog-to-Digital Converter

AGBT
AGBT
Aurora GigaBit Trace module on EEC

AGBT
AGBT
Aurora GigaBit Trace module on EEC

AGBT
AGBT
Aurora GigaBit Trace module on EEC

AGBT
AGBT
Aurora GigaBit Trace module on EEC

AMU.LMU_DAM
AMU.LMU_DAM
Default Application Memory

AMU.LMU_DAM
AMU.LMU_DAM
Default Application Memory

Glossary

AMU.LMU_DAM
AMU.LMU_DAM
Default Application Memory

AMU.LMU_DAM
AMU.LMU_DAM
Default Application Memory

Application SW
Application SW
Any software unit not provided by Infineon.

Application SW
Application SW
Any software unit not provided by Infineon.

Application SW
Application SW
Any software unit not provided by Infineon.

Application SW
Application SW
Any software unit not provided by Infineon.

ASCLIN
ASCLIN
Asynchronous/Synchronous Serial Controller with LIN

ASCLIN
ASCLIN
Asynchronous/Synchronous Serial Controller with LIN

ASCLIN
ASCLIN
Asynchronous/Synchronous Serial Controller with LIN

ASCLIN
ASCLIN
Asynchronous/Synchronous Serial Controller with LIN

BBB
BBB
Back Bone Bus

BBB
BBB
Back Bone Bus

BBB
BBB
Back Bone Bus

BBB
BBB
Back Bone Bus

BIST
BIST
Built-in Self Test

BIST
BIST
Built-in Self Test

BIST
BIST
Built-in Self Test

BIST
BIST
Built-in Self Test

BMHD
BMHD
Boot Mode Header

BMHD
BMHD
Boot Mode Header

BMHD
BMHD
Boot Mode Header

BMHD
BMHD
Boot Mode Header

CCF
CCF
Common Cause Failure

CCF
CCF
Common Cause Failure

CCF

CCF

Common Cause Failure

CCF

CCF

Common Cause Failure

CCU6

CCU6

Capture Compare Unit 6

CCU6

CCU6

Capture Compare Unit 6

CCU6

CCU6

Capture Compare Unit 6

CCU6

CCU6

Capture Compare Unit 6

CCU

CCU

Clock Control Unit

CCU

CCU

Clock Control Unit

CCU

CCU

Clock Control Unit

CCU

CCU

Clock Control Unit

CHSW

CHSW

Checker Software. It is part of [FW] and it is intended to evaluate whether the device configuration and preparation for user code execution are correct after completion of the Startup Software.

CHSW

CHSW

Checker Software. It is part of [FW] and it is intended to evaluate whether the device configuration and preparation for user code execution are correct after completion of the Startup Software.

CHSW

CHSW

Checker Software. It is part of [FW] and it is intended to evaluate whether the device configuration and preparation for user code execution are correct after completion of the Startup Software.

CHSW

CHSW

Checker Software. It is part of [FW] and it is intended to evaluate whether the device configuration and preparation for user code execution are correct after completion of the Startup Software.

CIF

CIF

Camera Interface

CIF

CIF

Camera Interface

CIF

CIF

Camera Interface

CIF

CIF

Camera Interface

CONVCTRL

CONVCTRL

Converter Control

CONVCTRL

CONVCTRL

Converter Control

CONVCTRL

CONVCTRL

Converter Control

CONVCTRL

CONVCTRL

Converter Control

CPU

CPU

Central Processing Unit

CPU

CPU

Central Processing Unit

CPU
CPU
Central Processing Unit

CPU
CPU
Central Processing Unit

CRC
CRC
Cyclic Redundancy Code

CRC
CRC
Cyclic Redundancy Code

CRC
CRC
Cyclic Redundancy Code

CRC
CRC
Cyclic Redundancy Code

DAP
DAP
Device Access Port

DAP
DAP
Device Access Port

DAP
DAP
Device Access Port

DAP
DAP
Device Access Port

DBAB
DBAB
Double-Bit Address Buffer

DBAB
DBAB
Double-Bit Address Buffer

DBAB
DBAB
Double-Bit Address Buffer

DBAB
DBAB
Double-Bit Address Buffer

DBE
DBE
Double-Bit Error

DBE
DBE
Double-Bit Error

DBE
DBE
Double-Bit Error

DBE
DBE
Double-Bit Error

DEBUG
DEBUG
Debug JTAG/JTAP Interface

DEBUG
DEBUG
Debug JTAG/JTAP Interface

DEBUG
DEBUG
Debug JTAG/JTAP Interface

DEBUG
DEBUG
Debug JTAG/JTAP Interface

DEC
DEC
Double Error Correction

DEC
DEC
Double Error Correction

DEC
DEC
Double Error Correction

DEC
DEC
Double Error Correction

DED
DED
Double Error Detection

DED
DED
Double Error Detection

DED
DED
Double Error Detection

DED
DED
Double Error Detection

DFX
DFX
Design For Test & Reliability

DFX
DFX
Design For Test & Reliability

DFX
DFX
Design For Test & Reliability

DFX
DFX
Design For Test & Reliability

DLMU
DLMU
Direct-connected Local Memory Unit

DLMU
DLMU
Direct-connected Local Memory Unit

Glossary

DLMU

DLMU

Direct-connected Local Memory Unit

DLMU

DLMU

Direct-connected Local Memory Unit

DMA

DMA

Direct Memory Access

DMA

DMA

Direct Memory Access

DMA

DMA

Direct Memory Access

DMA

DMA

Direct Memory Access

DMI

DMI

Data Memory Interface

DMI

DMI

Data Memory Interface

DMI

DMI

Data Memory Interface

DMI

DMI

Data Memory Interface

DSADC

DSADC

Delta-Sigma Analog to Digital Converter

DSADC

DSADC

Delta-Sigma Analog to Digital Converter

DSADC
DSADC
Delta-Sigma Analog to Digital Converter

DSADC
DSADC
Delta-Sigma Analog to Digital Converter

DSPR
DSPR
Data Scratch Pad RAM

DSPR
DSPR
Data Scratch Pad RAM

DSPR
DSPR
Data Scratch Pad RAM

DSPR
DSPR
Data Scratch Pad RAM

DTI
DTI
Diagnostic Time Interval

DTI
DTI
Diagnostic Time Interval

DTI
DTI
Diagnostic Time Interval

DTI
DTI
Diagnostic Time Interval

DTSC
DTSC
Die Temperature Monitor Core

DTSC
DTSC
Die Temperature Monitor Core

DTSC

DTSC

Die Temperature Monitor Core

DTSC

DTSC

Die Temperature Monitor Core

DTS

DTS

Die Temperature Monitor

DTS

DTS

Die Temperature Monitor

DTS

DTS

Die Temperature Monitor

DTS

DTS

Die Temperature Monitor

EBCU

EBCU

External Bus Control Unit.

EBCU

EBCU

External Bus Control Unit.

EBCU

EBCU

External Bus Control Unit.

EBCU

EBCU

External Bus Control Unit.

EBU

EBU

External Bus Unit

EBU

EBU

External Bus Unit

EBU
EBU
External Bus Unit

EBU
EBU
External Bus Unit

ECC
ECC
Error Correction Code

ECC
ECC
Error Correction Code

ECC
ECC
Error Correction Code

ECC
ECC
Error Correction Code

EDC
EDC
Error Detection Code

EDC
EDC
Error Detection Code

EDC
EDC
Error Detection Code

EDC
EDC
Error Detection Code

EDSADC
EDSADC
Enhanced Delta-Sigma Analog-to-Digital Converter

EDSADC
EDSADC
Enhanced Delta-Sigma Analog-to-Digital Converter

Glossary

EDSADC
EDSADC
Enhanced Delta-Sigma Analog-to-Digital Converter

EDSADC
EDSADC
Enhanced Delta-Sigma Analog-to-Digital Converter

EMEM
EMEM
Extension Memory

EMEM
EMEM
Extension Memory

EMEM
EMEM
Extension Memory

EMEM
EMEM
Extension Memory

ENDINIT
ENDINIT
End of Initialization

ENDINIT
ENDINIT
End of Initialization

ENDINIT
ENDINIT
End of Initialization

ENDINIT
ENDINIT
End of Initialization

ERAY
ERAY
Flexray Controller

ERAY
ERAY
Flexray Controller

ERAY
ERAY
Flexray Controller

ERAY
ERAY
Flexray Controller

ERU
ERU
External Request Unit

ERU
ERU
External Request Unit

ERU
ERU
External Request Unit

ERU
ERU
External Request Unit

ES
ES
Emergency Stop

ES
ES
Emergency Stop

ES
ES
Emergency Stop

ES
ES
Emergency Stop

ESM
ESM
External Safety Mechanism

ESM
ESM
External Safety Mechanism

ESM

ESM

External Safety Mechanism

ESM

ESM

External Safety Mechanism

EVADC

EVADC

Enhanced Versatile Analog-to-Digital Converter

EVADC

EVADC

Enhanced Versatile Analog-to-Digital Converter

EVADC

EVADC

Enhanced Versatile Analog-to-Digital Converter

EVADC

EVADC

Enhanced Versatile Analog-to-Digital Converter

EVR

EVR

Embedded Voltage Regulator

EVR

EVR

Embedded Voltage Regulator

EVR

EVR

Embedded Voltage Regulator

EVR

EVR

Embedded Voltage Regulator

FB

FB

Functional Block

FB

FB

Functional Block

FB
FB
Functional Block

FB
FB
Functional Block

FCE
FCE
Flexible CRC Engine

FCE
FCE
Flexible CRC Engine

FCE
FCE
Flexible CRC Engine

FCE
FCE
Flexible CRC Engine

FFI
FFI
Freedom From Interference

FFI
FFI
Freedom From Interference

FFI
FFI
Freedom From Interference

FFI
FFI
Freedom From Interference

FPI
FPI
Flexible Peripheral Interconnect (Bus protocol)

FPI
FPI
Flexible Peripheral Interconnect (Bus protocol)

FPI

FPI

Flexible Peripheral Interconnect (Bus protocol)

FPI

FPI

Flexible Peripheral Interconnect (Bus protocol)

FRT

FRT

Fault Reaction Time

FRT

FRT

Fault Reaction Time

FRT

FRT

Fault Reaction Time

FRT

FRT

Fault Reaction Time

FSP

FSP

Fault Signal Protocol

FSP

FSP

Fault Signaling Protocol

FSP

FSP

Fault Signal Protocol

FSP

FSP

Fault Signaling Protocol

FSP

FSP

Fault Signal Protocol

FSP

FSP

Fault Signaling Protocol

FSP
FSP
Fault Signal Protocol

FSP
FSP
Fault Signaling Protocol

FTTI
FTTI
Fault Tolerant Time Interval

FTTI
FTTI
Fault Tolerant Time Interval

FTTI
FTTI
Fault Tolerant Time Interval

FTTI
FTTI
Fault Tolerant Time Interval

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FUC
FUC
Functional Use Case

FW
FW
System Firmware

FW
FW
Firmware

FW
FW
System Firmware

FW
FW
Firmware

FW
FW
System Firmware

FW
FW
Firmware

FW
FW
System Firmware

FW
FW
Firmware

GETH
GETH
Giga Ethernet Controller

GETH
GETH
Giga Ethernet Controller

Glossary

GETH
GETH
Giga Ethernet Controller

GETH
GETH
Giga Ethernet Controller

GPI
GPI
General Purpose Input

GPI
GPI
General Purpose Input

GPI
GPI
General Purpose Input

GPI
GPI
General Purpose Input

GPIO
GPIO
General Purpose Input/Output

GPIO
GPIO
General Purpose Input/Output

GPIO
GPIO
General Purpose Input/Output

GPIO
GPIO
General Purpose Input/Output

GPO
GPO
General Purpose Output

GPO
GPO
General Purpose Output

GPO

GPO

General Purpose Output

GPO

GPO

General Purpose Output

GPT12

GPT12

General Purpose Timer 12

GPT12

GPT12

General Purpose Timer 12

GPT12

GPT12

General Purpose Timer 12

GPT12

GPT12

General Purpose Timer 12

GTM

GTM

Generic Timer Module

GTM

GTM

Generic Timer Module

GTM

GTM

Generic Timer Module

GTM

GTM

Generic Timer Module

HSM

HSM

Hardware Security Module

HSM

HSM

Hardware Security Module

HSM
HSM
Hardware Security Module

HSM
HSM
Hardware Security Module

HSSL
HSSL
High Speed Serial Link

HSSL
HSSL
High Speed Serial Link

HSSL
HSSL
High Speed Serial Link

HSSL
HSSL
High Speed Serial Link

I2C
I2C
Inter-Integrated Circuit Controller

I2C
I2C
Inter-Integrated Circuit Controller

I2C
I2C
Inter-Integrated Circuit Controller

I2C
I2C
Inter-Integrated Circuit Controller

ICU
ICU
Interrupt Control Unit

ICU
ICU
Interrupt Control Unit

ICU
ICU
Interrupt Control Unit

ICU
ICU
Interrupt Control Unit

IOM
IOM
I/O Monitor Unit

IOM
IOM
I/O Monitor Unit

IOM
IOM
I/O Monitor Unit

IOM
IOM
I/O Monitor Unit

IR
IR
Interrupt Router

IR
IR
Interrupt Router

IR
IR
Interrupt Router

IR
IR
Interrupt Router

ISP
ISP
Interrupt Service Provider

ISP
ISP
Interrupt Service Provider

ISP

ISP

Interrupt Service Provider

ISP

ISP

Interrupt Service Provider

ISR

ISR

Interrupt Service Routine

ISR

ISR

Interrupt Service Routine

ISR

ISR

Interrupt Service Routine

ISR

ISR

Interrupt Service Routine

JTAG

JTAG

Joint Test Action Group = IEEE1149.1

JTAG

JTAG

Joint Test Action Group = IEEE1149.1

JTAG

JTAG

Joint Test Action Group = IEEE1149.1

JTAG

JTAG

Joint Test Action Group = IEEE1149.1

LBIST

LBIST

Logic Built-in Self Test

LBIST

LBIST

Logic Built In Self Test

LBIST
LBIST
Logic Built-in Self Test

LBIST
LBIST
Logic Built In Self Test

LBIST
LBIST
Logic Built-in Self Test

LBIST
LBIST
Logic Built In Self Test

LBIST
LBIST
Logic Built-in Self Test

LBIST
LBIST
Logic Built In Self Test

LF
LF
Latent Fault (see ISO 26262-1.71)

LF
LF
Latent Fault (see ISO 26262-1.71)

LF
LF
Latent Fault (see ISO 26262-1.71)

LF
LF
Latent Fault (see ISO 26262-1.71)

LMU
LMU
Local Bus Memory Unit

LMU
LMU
Local Bus Memory Unit

LMU
LMU
Local Bus Memory Unit

LMU
LMU
Local Bus Memory Unit

LMU_DAM
LMU_DAM
Default Application Memory

LMU_DAM
LMU_DAM
Default Application Memory

LMU_DAM
LMU_DAM
Default Application Memory

LMU_DAM
LMU_DAM
Default Application Memory

MBIST
MBIST
Memory Built-In Self Test

MBIST
MBIST
Memory Built-In Self Test

MBIST
MBIST
Memory Built-In Self Test

MBIST
MBIST
Memory Built-In Self Test

MCDS
MCDS
Multi Core Debug Solution

MCDS
MCDS
Multi Core Debug Solution

MCDS
MCDS
Multi Core Debug Solution

MCDS
MCDS
Multi Core Debug Solution

MCMCAN
MCMCAN
CAN controller

MCMCAN
MCMCAN
CAN controller

MCMCAN
MCMCAN
CAN controller

MCMCAN
MCMCAN
CAN controller

MCU
MCU
Micro Controller Unit

MCU
MCU
Micro Controller Unit

MCU
MCU
Micro Controller Unit

MCU
MCU
Micro Controller Unit

MMIC
MMIC
Monolithic Microwave Integrated Circuit

MMIC
MMIC
Monolithic Microwave Integrated Circuit

MMIC
MMIC
Monolithic Microwave Integrated Circuit

MMIC
MMIC
Monolithic Microwave Integrated Circuit

MONBIST
MONBIST
Monitor Built In Self Test

MONBIST
MONBIST
Monitor Built In Self Test

MONBIST
MONBIST
Monitor Built In Self Test

MONBIST
MONBIST
Monitor Built In Self Test

MPF
MPF
Multiple-point fault (see ISO 26262-1.77)

MPF
MPF
Multiple-point fault (see ISO 26262-1.77)

MPF
MPF
Multiple-point fault (see ISO 26262-1.77)

MPF
MPF
Multiple-point fault (see ISO 26262-1.77)

MPU
MPU
Memory Protection Unit

MPU
MPU
Memory Protection Unit

MPU
MPU
Memory Protection Unit

MPU
MPU
Memory Protection Unit

MSC
MSC
Micro Second Channel

MSC
MSC
Micro Second Channel

MSC
MSC
Micro Second Channel

MSC
MSC
Micro Second Channel

MTU
MTU
Memory Test Unit

MTU
MTU
Memory Test Unit

MTU
MTU
Memory Test Unit

MTU
MTU
Memory Test Unit

NMI
NMI
Non-Maskable Interrupt

NMI
NMI
Non-Maskable Interrupt

NMI

NMI

Non-Maskable Interrupt

NMI

NMI

Non-Maskable Interrupt

NVM

NVM

Non Volatile Memory

NVM

NVM

Non Volatile Memory

NVM

NVM

Non Volatile Memory

NVM

NVM

Non Volatile Memory

OCDS

OCDS

On-Chip Debug Support

OCDS

OCDS

On-Chip Debug Support

OCDS

OCDS

On-Chip Debug Support

OCDS

OCDS

On-Chip Debug Support

OCDS

OCDS

On-Chip Debug Support

OCDS

OCDS

On-Chip Debug Support

OCDS
OCDS
On-Chip Debug Support

OCDS
OCDS
On-Chip Debug Support

PBIST
PBIST
Power Built-in Self Test

PBIST
PBIST
Power Built-in Self Test

PBIST
PBIST
Power Built-in Self Test

PBIST
PBIST
Power Built-in Self Test

PES
PES
Port Emergency Stop

PES
PES
Port Emergency Stop

PES
PES
Port Emergency Stop

PES
PES
Port Emergency Stop

PFI
PFI
Program Flash Interface

PFI
PFI
Program Flash Interface

PFI
PFI
Program Flash Interface

PFI
PFI
Program Flash Interface

PFLASH
PFLASH
Program FLASH

PFLASH
PFLASH
Program FLASH

PFLASH
PFLASH
Program FLASH

PFLASH
PFLASH
Program FLASH

PLL
PLL
Phase-Locked Loop.

PLL
PLL
Phase-Locked Loop.

PLL
PLL
Phase-Locked Loop.

PLL
PLL
Phase-Locked Loop.

PMI
PMI
Program Memory Interface

PMI
PMI
Program Memory Interface

PMI
PMI
Program Memory Interface

PMI
PMI
Program Memory Interface

PMS
PMS
Power Management System

PMS
PMS
Power Management System

PMS
PMS
Power Management System

PMS
PMS
Power Management System

PORST
PORST
Power-On Reset

PORST
PORST
Power-On Reset

PORST
PORST
Power-On Reset

PORST
PORST
Power-On Reset

PORT
PORT
General Purpose I/O ports and Peripheral I/O Lines

PORT
PORT
General Purpose I/O ports and Peripheral I/O Lines

PORT
PORT
General Purpose I/O ports and Peripheral I/O Lines

PORT
PORT
General Purpose I/O ports and Peripheral I/O Lines

PSI5
PSI5
Peripheral Sensor Interface

PSI5
PSI5
Peripheral Sensor Interface

PSI5
PSI5
Peripheral Sensor Interface

PSI5
PSI5
Peripheral Sensor Interface

PSPR
PSPR
Program Scratch Pad RAM

PSPR
PSPR
Program Scratch Pad RAM

PSPR
PSPR
Program Scratch Pad RAM

PSPR
PSPR
Program Scratch Pad RAM

QSPI
QSPI
Queued SPI Controller

QSPI
QSPI
Queued SPI Controller

QSPI
QSPI
Queued SPI Controller

QSPI
QSPI
Queued SPI Controller

RESET
RESET
Reset control unit

RESET
RESET
Reset control unit

RESET
RESET
Reset control unit

RESET
RESET
Reset control unit

RHF
RHF
Random Hardware Failure

RHF
RHF
Random Hardware Failure

RHF
RHF
Random Hardware Failure

RHF
RHF
Random Hardware Failure

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

RIF
RIF
Radar Interface

SBAB
SBAB
Single-Bit Address Buffer

SBAB
SBAB
Single-Bit Address Buffer

SBAB
SBAB
Single-Bit Address Buffer

SBAB
SBAB
Single-Bit Address Buffer

SBCU
SBCU
System Peripheral Bus Control Unit

SBCU
SBCU
System Peripheral Bus Control Unit

SBCU

SBCU

System Peripheral Bus Control Unit

SBCU

SBCU

System Peripheral Bus Control Unit

SBE

SBE

Single-Bit Error

SBE

SBE

Single-Bit Error

SBE

SBE

Single-Bit Error

SBE

SBE

Single-Bit Error

SBST

SBST

Software-based Self Test

SBST

SBST

Software-based Self Test

SBST

SBST

Software-based Self Test

SBST

SBST

Software-based Self Test

SCR

SCR

Standby Controller

SCR

SCR

Standby Controller

SCR
SCR
Standby Controller

SCR
SCR
Standby Controller

SCU
SCU
System Control Unit

SCU
SCU
System Control Unit

SCU
SCU
System Control Unit

SCU
SCU
System Control Unit

SDMMC
SDMMC
SD- and eMMC Interface

SDMMC
SDMMC
SD- and eMMC Interface

SDMMC
SDMMC
SD- and eMMC Interface

SDMMC
SDMMC
SD- and eMMC Interface

SEC
SEC
Single Error Correction

SEC
SEC
Single Error Correction

SEC

SEC

Single Error Correction

SEC

SEC

Single Error Correction

SENT

SENT

Single Edge Nibble Transmission

SENT

SENT

Single Edge Nibble Transmission

SENT

SENT

Single Edge Nibble Transmission

SENT

SENT

Single Edge Nibble Transmission

SEooC

SEooC

Safety Element Out Of Context.

SEooC

SEooC

Safety Element Out Of Context.

SEooC

SEooC

Safety Element Out Of Context.

SEooC

SEooC

Safety Element Out Of Context.

SFR

SFR

Special Function Register

SFR

SFR

Special Function Register

SFR
SFR
Special Function Register

SFR
SFR
Special Function Register

SMC
SMC
Safety Mechanism Configuration

SMC
SMC
Safety Mechanism Configuration

SMC
SMC
Safety Mechanism Configuration

SMC
SMC
Safety Mechanism Configuration

SM
SM
Safety Mechanism

SM
SM
Safety Mechanism

SM
SM
Safety Mechanism

SM
SM
Safety Mechanism

SMU
SMU
Safety Management Unit

SMU
SMU
Safety Management Unit

SMU

SMU

Safety Management Unit

SMU

SMU

Safety Management Unit

SPB

SPB

System Peripheral Bus (based on FPI protocol)

SPB

SPB

System Peripheral Bus (based on FPI protocol)

SPB

SPB

System Peripheral Bus (based on FPI protocol)

SPB

SPB

System Peripheral Bus (based on FPI protocol)

SPF

SPF

Single-point fault (see ISO 26262-1.22)

SPF

SPF

Single-point fault (see ISO 26262-1.22)

SPF

SPF

Single-point fault (see ISO 26262-1.22)

SPF

SPF

Single-point fault (see ISO 26262-1.22)

SPI

SPI

Serial Peripheral Interface

SPI

SPI

Serial Peripheral Interface

SPI
SPI
Serial Peripheral Interface

SPI
SPI
Serial Peripheral Interface

SPU
SPU
Signal Processing Unit

SPU
SPU
Signal Processing Unit

SPU
SPU
Signal Processing Unit

SPU
SPU
Signal Processing Unit

SRI
SRI
Shared Resource Interconnect

SRI
SRI
Shared Resource Interconnect

SRI
SRI
Shared Resource Interconnect

SRI
SRI
Shared Resource Interconnect

SRN
SRN
Service Request Node

SRN
SRN
Service Request Node

SRN

SRN

Service Request Node

SRN

SRN

Service Request Node

SRPN

SRPN

Service Request Priority Number

SRPN

SRPN

Service Request Priority Number

SRPN

SRPN

Service Request Priority Number

SRPN

SRPN

Service Request Priority Number

SSH

SSH

Sram Support Hardware

SSH

SSH

Sram Support Hardware

SSH

SSH

Sram Support Hardware

SSH

SSH

Sram Support Hardware

SSW

SSW

Startup Software. It is part of the [FW] and it is automatically executed after a reset. It contains various procedures for initializing the device.

SSW

SSW

Startup Software. It is part of the [FW] and it is automatically executed after a reset. It contains various procedures for initializing the device.

SSW

SSW

Startup Software. It is part of the [FW] and it is automatically executed after a reset. It contains various procedures for initializing the device.

SSW

SSW

Startup Software. It is part of the [FW] and it is automatically executed after a reset. It contains various procedures for initializing the device.

STM

STM

System Timer

STM

STM

System Timer

STM

STM

System Timer

STM

STM

System Timer

System level

System level

Refers to any measure to be implemented by the developer in one of the elements that compose a system. According to ISO 26262-1, a system relates at least a sensor, a controller and an actuator.

System level

System level

Refers to any measure to be implemented by the developer in one of the elements that compose a system. According to ISO 26262-1, a system relates at least a sensor, a controller and an actuator.

System level

System level

Refers to any measure to be implemented by the developer in one of the elements that compose a system. According to ISO 26262-1, a system relates at least a sensor, a controller and an actuator.

System level

System level

Refers to any measure to be implemented by the developer in one of the elements that compose a system. According to ISO 26262-1, a system relates at least a sensor, a controller and an actuator.

TBE

TBE

Triple Bit Error

TBE
TBE
Triple Bit Error

TBE
TBE
Triple Bit Error

TBE
TBE
Triple Bit Error

TCS
TCS
Transaction Control Set

TCS
TCS
Transaction Control Set

TCS
TCS
Transaction Control Set

TCS
TCS
Transaction Control Set

TCU
TCU
Test Control Unit

TCU
TCU
Test Control Unit

TCU
TCU
Test Control Unit

TCU
TCU
Test Control Unit

TED
TED
Triple Error Detection

TED
TED
Triple Error Detection

TED
TED
Triple Error Detection

TED
TED
Triple Error Detection

TIM
TIM
Timer Input Module

TIM
TIM
Timer Input Module

TIM
TIM
Timer Input Module

TIM
TIM
Timer Input Module

TOM
TOM
Timer Output Module

TOM
TOM
Timer Output Module

TOM
TOM
Timer Output Module

TOM
TOM
Timer Output Module

TRACE
TRACE
Signal Trace functionalities offered by MCDS, mini MCDS and TRAM.

TRACE
TRACE
Signal Trace functionalities offered by MCDS, mini MCDS and TRAM.

TRACE
TRACE
Signal Trace functionalities offered by MCDS, mini MCDS and TRAM.

TRACE
TRACE
Signal Trace functionalities offered by MCDS, mini MCDS and TRAM.

VADC
VADC
Versatile Analog-to-Digital Converter

VADC
VADC
Versatile Analog-to-Digital Converter

VADC
VADC
Versatile Analog-to-Digital Converter

VADC
VADC
Versatile Analog-to-Digital Converter

VMT
VMT
Volatile Memory Test

VMT
VMT
Volatile Memory Test

VMT
VMT
Volatile Memory Test

VMT
VMT
Volatile Memory Test

WDT
WDT
Watchdog Timer

Glossary

WDT

WDT

Watchdog Timer

WDT

WDT

Watchdog Timer

WDT

WDT

Watchdog Timer

Edition 2020-10-29

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about any aspect of this document?

Email: erratum@infineon.com

Document reference
IFX-jobid__20201029151444243