TriCore™ AURIX™ Family

32-bit

AURIX Safety Manual

AP32224

Application Note

V1.5 2017-05

Z8F52027859

Microcontrollers

Edition 2017-05

Published by
Infineon Technologies AG
81726 Munich, Germany

Document Change History

Date	Version	Change Description
2017-03	1.5	[0000047041-376] 4.16 Removed FSI0 from table 5. FSI0 RAM is neither accessible nor testable. Typo corrected: GTM MSC0/1 replaced by GTM MCS0/1 [0000047041-377] 5.3.10 SM_AURIX_DMA_3:1 completely reworked the description of SM1[AoU].DMA:Monitor with a better identification of the failure modes to be covered by software and additional implementation hints [0000047041-379] 4.15 Added TC23x ADAS in the list of devices implementing the LMU SRAM [0000047041-381, 0000047041-413] 4.25.1 Updated figure 9 and 10 and added table 7 and 9 to support BGA416 package [0000047041-383, 0000047041-384] 1.1, 1.6.1 Added notes to introduce requirement tags and differentiate them from safety mechanism tags [0000047041-388] 5.1.31 Added limitation to usage of DMA Address CRC safety mechanism [0000047041-389] 5.1.27 Modified limitation on timestamp usage with circular buffer with regards to erratum DMA_TC34 [0000047041-390] 5.3.13 Added SENT, ASCLIN and PSI5 SafeProtocol safety mechanisms [0000047041-392] 5.2.3 Removed reference to SM2[HW].BANDGAP:HWBIST. This safety mechanism is obsolete [0000047041-395] 5.2.31, 5.2.32, 5.2.33 Extended description of CPU MPU, BUS MPU and LMU MPU safety mechanisms [0000047041-398]

Document Change History

Date	Version	Change Description
		5.2.34 Added WDTxCON0.PW, WDTxCON1.PAR and WDTxCON1.TCR to list of register to check [0000047041-399] 1.4 Replaced the term “single channel software applications” by “non-Lockstep CPU” [0000047041-400] 3.1 Added SPB to SRI Bridge and IR to list of critical parts [0000047041-401] 4.7.2 Added reference to AP32329 AURIX™ Secondary Voltage Monitors - Setting threshold levels [0000047041-402] 4.15.2, 5.3.15 New AoU SM_AURIX_LMU_SRAM_2 [0000047041-404] 5.2.26 Typo in SMU Alarm configuration register names [0000047041-414] 4.5.1 Updated 1st and 2nd attention section. Scope of SBST on CPU core refined according to SBST Safety Analysis [0000047041-416] 3.1 Added ERU to list of non-safety related parts [0000047041-417] 7.3 Added attention section to mention missing dependent failure initiator in VADC-VADC Application Dependant Scenario [0000047041-419] 3.8.2 Changed cross-reference from 4.2.5 to 4.3 [0000047041-421] 4.25.1 Added note for usage of TC23x ADAS ADCs in redundant VADC-VADC scenario [0000047041-422] 5.3.10 Added cross-references
2015-11	1.4	[0000047041-363] 4.28.1 SM_AURIX_ADAS_2 New AoU specific to ADAS products

Document Change History

Date	Version	Change Description
		[0000047041-365] 4.26.4, 4.26.5 Fixed figures 14 and 15 as they were not representing the correct scenario (swapped the two figures) [0000047041-368] 5.1.40 extended SM1[HW].LMU:AP safety mechanism description [0000047041-354, 0000047041-358] 5.1.50 fixed description of SM1[HW].Register:SFF, and added reference to SM1[HW].Register:FTSFF [0000047041-360] 5.2.27 changed hint that SMU need to be in START state for this test, actually it must be in RUN state [0000047041-348] 8 SM_AURIX_31 New AoU specific to bare die products

Trademarks

Table of Contents

References[1]

[1] ISO 26262:2011, Road vehicles — Functional safety
ISO/FDIS 26262-10:2012, Road vehicles — Functional safety, Part 10: Guideline on ISO 26262

[2] TriCore V1.6P & TC1.6E Core Architecture, User's Manual (Volume 1, v1.0D10 2012-02 and Volume 2, v1.0D15 2013-07), Infineon Technologies AG Munich

[3] TC29x B-Step User's Manual, v1.3, 2014-12, Infineon Technologies AG Munich
TC27x B-Step User's Manual, v1.4.1, 2014-02, Infineon Technologies AG Munich
TC27x C-Step User's Manual, v2.2, 2014-12, Infineon Technologies AG Munich
TC27x D-Step User's Manual, v2.2, 2014-12, Infineon Technologies AG Munich
TC26x B-Step User's Manual, v1.3, 2014-12, Infineon Technologies AG Munich
TC21x/TC22x/TC23x User's Manual, v1.1 2014-12, Infineon Technologies AG Munich

[4] TC290/TC297/TC298/TC299 BB-Step Data Sheet, v1.1, 2015-05
Infineon Technologies AG Munich
TC275/TC277 Data Sheet, BC-Step, v1.0, 2014-07 Infineon Technologies AG Munich
TC275/TC277 Data Sheet, CA-Step, v1.0, 2014-11 Infineon Technologies AG Munich
TC270/TC275/TC277 Data Sheet, DB-Step, v1.0, 2015-07 Infineon Technologies AG Munich
TC260/TC264/TC265/TC267 BB-Step Data Sheet, v1.1, 2015-07,
Infineon Technologies AG Munich
TC233/TC234/TC237 Data Sheet A-Step, v1.1, 2015-06, Infineon Technologies AG Munich
TC212/ TC213/ TC214/ TC222/ TC223/ TC224 Data Sheet v1.0, 2015-06,
Infineon Technologies AG Munich

[5] Latest available Errata Sheet for the considered product, Infineon Technologies AG Munich

[6] AURIX SafeTlib User Manual, v8.11, 2015-09, Infineon Technologies AG Munich

[7] AURIX Family FMEDA v3.7, 2017-05, Infineon Technologies AG Munich

[8] TriCore AURIX SBST User Manual v1.0, 2016-10, Infineon Technologies AG Munich

[1] For references [2] to [8], Newer versions replace older versions, unless specifically noted otherwise

1 About this document

1.1 Cautions

Please read this manual carefully and be sure you understand the information provided before integrating the Infineon product in a safety-related system [1].

This document is the safety manual for the Infineon 32-bit TriCore^{TM} Microcontrollers of the AURIX family. Although this is a family document for the complete AURIX family, updates may target only a subset of the devices of the AURIX family. The Table 1 lists each of the supported devices together with an instance number and the latest safety manual version with changes relevant for this device. The instance number is incremented with every new release of this document in which changes relevant for the considered devices were made.

Table 1 Device Relevant Changes

Device	Instance	Latest relevant changes
TC290/TC297/TC298/TC299	3	V1.5
TC270/TC275/TC277 B-Step	3	V1.5
TC270/TC275/TC277 C-Step	3	V1.5
TC260/TC264/TC265	3	V1.5
TC233/TC234/TC237	3	V1.5
TC212/ TC213/ TC214/ TC222/ TC223/ TC224	3	V1.5

It is the responsibility of the system integrator to ensure that the AURIX 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 in safety-related systems:

- [SM_AURIX_25]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.[/req]
- 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 fulfill a recommendation.

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

Note: Assumptions of use are marked by beginning and ending tags which could be used to import the assumptions of use into a requirements management tool

1.2 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 microcontroller hardware and SafeTlib software together as a Safety Element out of Context (SEooC) into their system.

1.3 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.

The purpose of this manual is to support the validation of the product integration regarding the usage of safety features and mechanism. 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.

1.4 ISO 26262

The AURIX 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 **[1]**. It was developed as a Safety Element out of Context (SEooC) in accordance with the ISO 26262-10 **[1]** with ASIL D capability.

The scope of the SEooC comprises:

- The AURIX microcontroller hardware component
- Assumptions of use related to the software elements that
  - support the integration to the AURIX microcontroller hardware components in a safety-related application
  - support the single point fault metric up to ASIL B for a non-Lockstep CPU
  - Note: The SafeTlib software product offered by Infineon supports the implementation of some of these assumptions [6]
- 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 26262-10:2011 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 (AoU) of the microcontroller with respect to its intended use
- the description of the microcontroller configuration and related HW and/or SW 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 microcontroller in safety-related applications.

This document is targeting high functional safety integrity levels. For functional safety goals which do not require high functional safety integrity levels, system integrators will need to tailor the requirements for their specific application.

1.5 Required additional reading

This manual assumes that readers are familiar with

- the TriCore architecture manual **[2]**
- the User's Manual **[3]**

- the data sheet [4]

[SM_AURIX_25]Before using the AURIX microcontroller and during the system product lifetime the information provided in the latest errata sheet [5] and SafeTlib software release notes addendum shall be evaluated for their relevance to the safety-related system performance.[/req]

1.6 Definitions of terms

1.6.1 Safety Mechanisms Identifiers

The safety mechanisms identifiers are the link with the AURIX FMEDA [7] which provides the information about the coverage of specific failure modes by specific safety mechanisms described in the safety manual.

The safety measures described in this manual are classified as:

- hardware safety mechanism
- assumptions of use

They are also classified as

- mechanisms to mitigate single points and residual faults; that is supporting the single-point fault metric
- or mechanisms to avoid dual-point faults from being latent; that is supporting the latent fault metric

To identify these properties, following conventions are followed when providing the identifier:

SM1[HW].<Part Name>:<Safety Mechanism>

SM1[HW] refers to a safety mechanism supporting the single-point fault metric and implemented in hardware, monitoring the failure modes of the part <Part Name>. <Safety Mechanism> provides a way to identify the essential characteristics of the safety mechanism: for example SM1[HW].CPU:LOCKSTEP for the lockstep architecture detecting errors in the CPU.

SM1[AoU].<Part Name>:<Safety Mechanism>

Same as SM1[HW], but it is assumed the application will provide this safety mechanism: for example SM1[AoU].CAN:SafeProtocol requires the application to implement an End-to-End safe communication protocol, if the MultiCAN+ module is used for communicating safety related data.

Assumption of Use (AoU) shall be implemented by the system integrator as safety features of the system or application safety software.

SM2[HW/AoU].<Part Name>:<Safety Mechanism>

SM2 refers to a safety mechanism supporting the latent fault metric: for example SM2[AoU].CLK:CLKMON for the SafeTlib test of the clock frequency monitors.

1.6.2 Application Software

Application Software is all software that is integrated by the **System Integrator**, including software products produced by Infineon such as PRO-SIL^{TM} SafeTlib and the MC-ISAR AUTOSAR MCAL, by opposition to the software embedded by Infineon in the AURIX during production such as the microcontroller Startup Software (SSW).

1.6.3 System Level

System Level measures are measures that have to be taken by the System Integrator when integrating the AURIX in a system.

1.7 Abbreviations & Acronyms

This manual uses abbreviations defined in [1],[2],[3],[5]. It does not define any new abbreviation or acronyms.

2 Product Overview

The AURIX 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 analyze 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 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 microcontroller is a high-performance microcontroller with up to three TriCore CPUs, program and data memories, buses, bus arbitration, interrupts system, DMA controller and a powerful set of on-chip peripherals. The AURIX 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 microcontroller offers several versatile on-chip peripheral units such as serial controllers, timer units, and analog-to-digital converters. Within the AURIX microcontroller, all these peripheral units are connected to the TriCore CPUs and system via the System Peripheral Bus (SPB) and the Shared Resource Interconnect (SRI). A number of I/O lines on the AURIX microcontroller ports are reserved for these peripheral units to communicate with the external world.

Figure 1 shows an example block diagram of an AURIX microcontroller TC27x derivate.

Please note that not all features that are shown in the block diagram are available in all TC27x package variants. For other derivatives, please consult the User's Manual **[3]**.

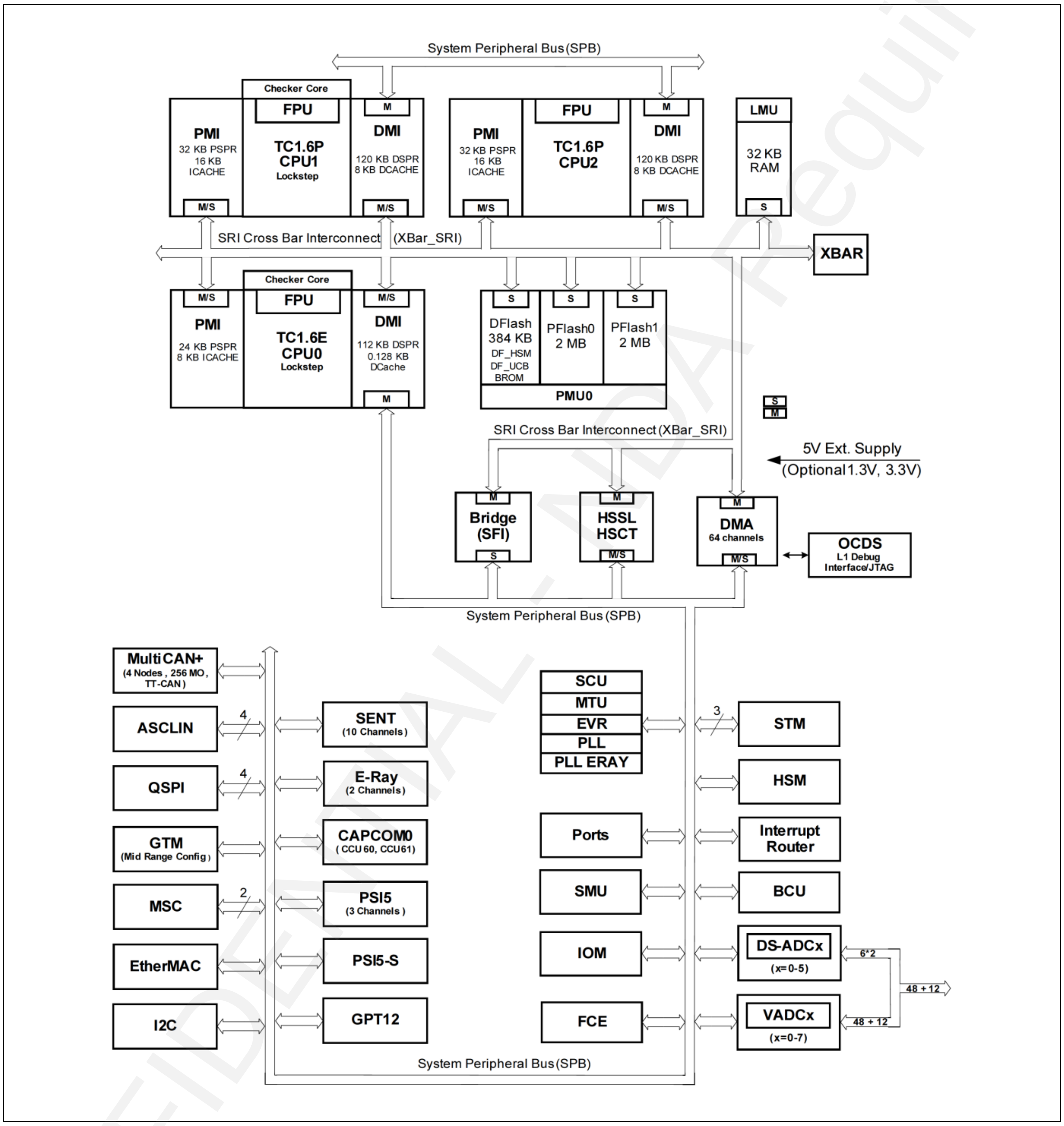

Figure 1 TC27x Block Diagram

3 Assumptions on System Level

The AURIX microcontroller is developed as a safety element out of context (SEooC) following the guidelines for a SEooC hardware component development according to ISO 26262-10 9.2.3. Accordingly, assumptions on system-level technical safety requirements have been specified. These assumptions define the baseline for the specification of the AURIX microcontroller safety architecture.

3.1 Purpose of the SEooC

The AURIX microcontroller is intended to be used as the central processing element of a controller subsystem according to ISO 26262-10, clause 4.2 definitions, supporting one or several safety functions.

Figure 2 shows the assumed purpose of the AURIX microcontroller in the context of an Electronic Control Unit (ECU).

- Input data is provided by one or multiple sensors to the ECU.
- The signal is processed by HW components on the ECU and forwarded to the digital, analog or communication input acquisition channels of the microcontroller.
- The AURIX microcontroller processes the signal data and provides outputs to drive one or multiple actuators, or transmits the output via a communication network to another ECU.

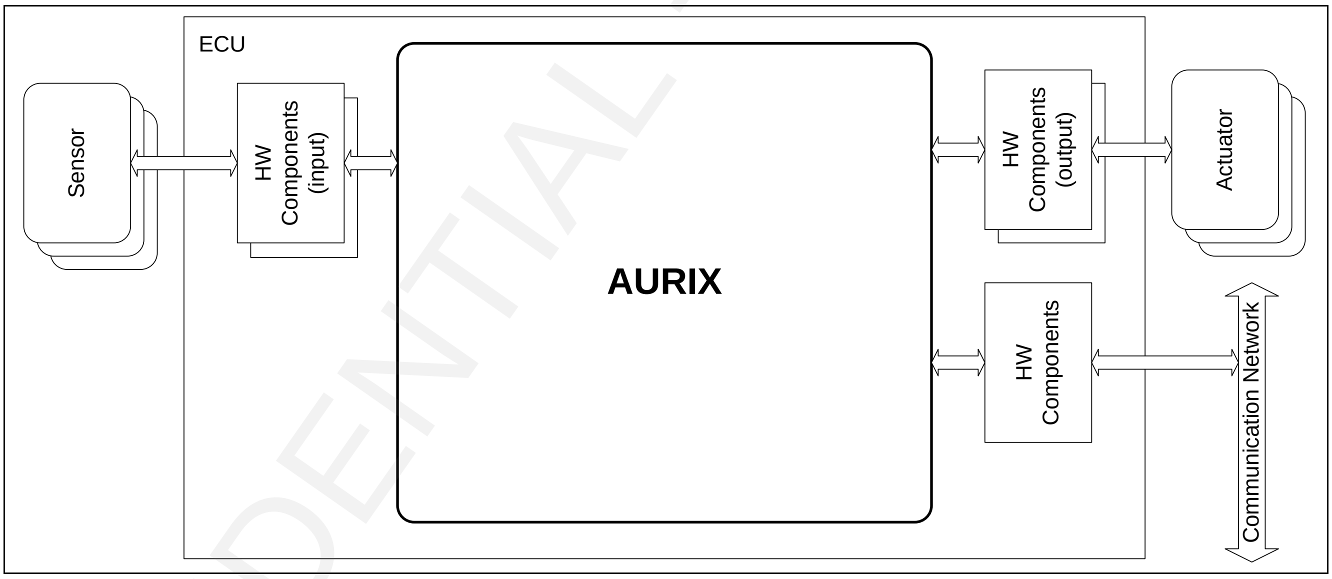

Figure 2 AURIX microcontroller in context of an Electronic Control Unit (ECU)

The AURIX microcontroller

- may be used for applications in safety related systems including systems that are classified as ASIL A, ASIL B, ASIL C or ASIL D.
- provides hardware and software functions that can be used to realize a safety function in conjunction with other hardware and software components.
- provides safety mechanisms, implemented by hardware parts and/or software components that are intended to monitor the failure modes of its functional parts.

From ISO 26262-10 classification point of view the microcontroller device is considered as a hardware component made of parts and sub-parts. A sub-part can itself be a part according to the design hierarchies. The parts of the AURIX microcontroller are classified in five categories. Some parts may belong to more than one category.

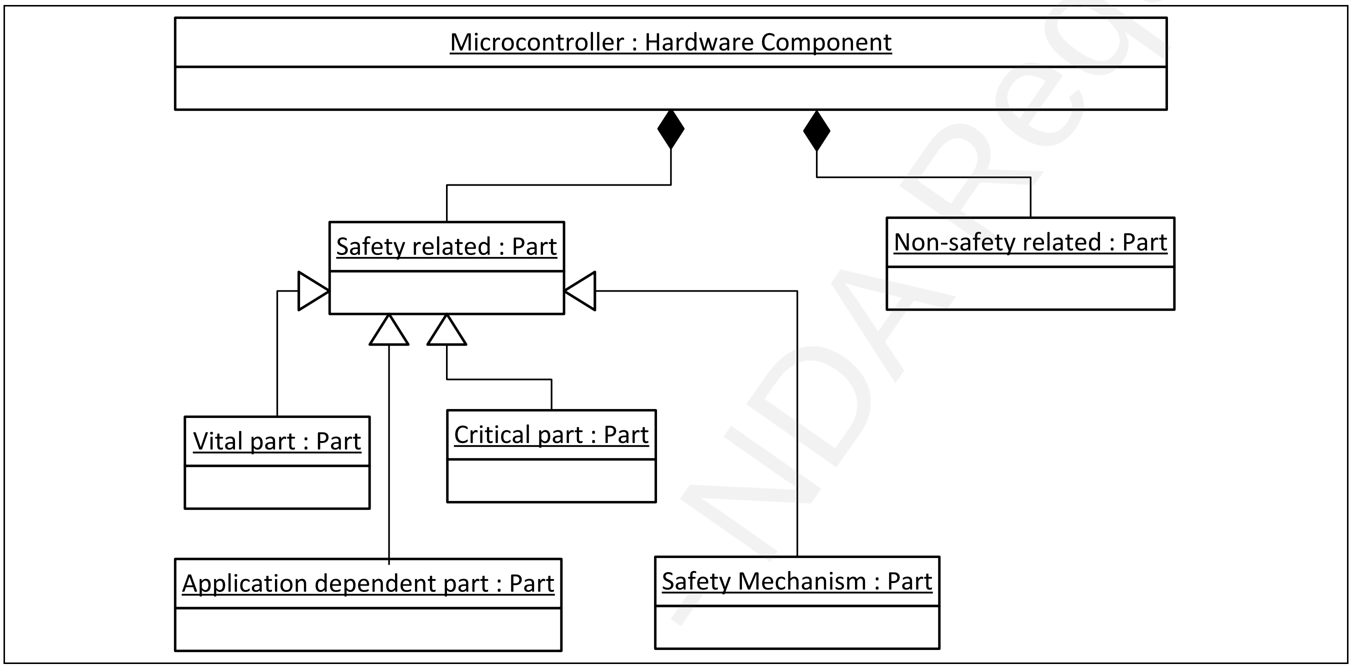
Microcontroller : Hardware Component
Safety related : Part
Non-safety related : Part
Vital part : Part
Critical part : Part
Application dependent part : Part
Safety Mechanism : Part

Figure 3 Microcontroller Parts

Table 2 AURIX Part Description

Category	**Description**
Vital Parts	These are the parts of the microcontroller for which application independent safety mechanisms are provided. The safety mechanisms control with high coverage the failure modes of the vital parts.
Application Dependent Parts	Parts of the microcontroller for which the fulfillment of the safety requirements requires a combination of application-level safety mechanisms and safety mechanisms provided by the microcontroller. Typical application dependent parts are peripheral modules participating in the data-acquisition, actuation control and system-level communication
Critical Parts	Parts of the microcontroller which have the potential to silently affect resources located in the vital parts used by the safety application software, or to simultaneously affect a part and its safety mechanism. A critical part is a potential dependent failure initiator. One typical example is a DMA engine that overwrites safety-related variables stored in a SRAM as a consequence of a malfunction in the address generation. The critical parts are considered in the probabilistic analysis and in the dependent failure analysis.
Safety Mechanisms	Parts that implement the safety mechanisms[1]
Non-Safety related Parts	Parts (and/or sub-parts) of the microcontroller that are not intended to be used in safety applications. They do not contribute to the ISO

Category	**Description**
	26262 metrics unless the safety analysis demonstrates that the expected independence of the safety related parts cannot be guaranteed.
Comment	[1] The "Safety Mechanisms" category classifies dedicated parts that implement one or more safety mechanisms (as defined by ISO 26262) but no intended functionality

Table 3 provides the list of vital parts, application dependent parts, critical parts, safety mechanisms, and non-safety related parts in the devices of the AURIX microcontroller family. All the parts and sub-parts are not necessarily present in all devices. Please refer to the appropriate device's user's manual [3].

Table 3 AURIX Parts Classification

Classification	**Part**
Vital parts	SRI, CPU core (with lockstep), PMI, DMI, SCU, PMU SRI, PMU FSI, PMU PFLASH, LMU SRAM
Application dependent parts	SPB, SPB to SRI Bridge, PMU DFLASH, PORTS, DMA[2b], IR, ASCLIN, QSPI, MSC, MultiCAN+, SENT, E-Ray, GTM, CCU6, GPT12, VADC, DS-ADC, PSI5, I2C, ETH[2b], HSSL[2b], LMU FFT, STM, Extended ADAS Memory (EMEM), CPU core (without lockstep)
Critical parts	Non-Lockstep CPU core (without lockstep), HSM[1a], OCDS, DMA[2a], ETH[2a], HSSL[2a], IR[2a], SPB to SRI Bridge[2a], CIF, FSI
Safety mechanisms	SMU, FCE, IOM
Non-safety related parts	HSM[1b], ERU[3]
Comments	[1] The HSM is classified as Critical Part [1a] as well as Non-Safety related Part [1b]. [1a] The classification as a Critical Part owes to the fact that the HSM has the potential to silently affect resources located in the Vital Parts used by the safety application software, or to simultaneously affect a Vital Part and its Safety Mechanism. [1b] The classification as a Non-Safety related Part owes to the fact that the HSM itself is a self-contained programmable CPU subsystem with separate intrinsic parts, for which no self-contained claim is made that random HW faults are covered by dedicated HW measures with a specific ASIL. [2] Specific parts (DMA, ETH, HSSL, IR and SPB to SRI Bridge) are classified as Critical Parts [2a] as well as Application Dependent Parts [2b]. [2a] The classification as a Critical Part owes to the fact that these parts have the potential to silently affect resources located in the Vital Parts used by the safety application software, or to simultaneously affect a Vital Part and its Safety Mechanism. [2b] The classification as Application Dependent Part owes to the fact that these parts also require a combination of application-level Safety Mechanisms and Safety Mechanisms provided by the

Classification	Part
	microcontroller to fulfill the safety requirements. [3] The peripheral triggering feature of the ERU is considered as non-safety related

3.2 Assumed Top-Level Safety Requirements

Top Level Safety Requirement 1: Avoid undetected error of a microcontroller SEooC vital parts for longer than 500 (five hundred) micro-seconds. An error is detected when the SMU reacts to an error event (alarm) provided by a hardware safety mechanism of the microcontroller SEooC.

Top Level Safety Requirement 2: Provide a Software-Based Self-Test detecting permanent random hardware faults in the fetch unit and pipelines of the TriCore CPU with a diagnostic coverage of 90%. The goal is to supports reaching the SPFM and LFM targets for ASIL B safety goals with regard to permanent failures.

Top Level Safety Requirement 3: Provide information related to the avoidance of dependent failures to enable the monitoring of application dependent parts failure modes caused by random hardware faults by means of internal functional redundancies. At least the following redundancy use cases shall be considered:

- GTM TIM /CCU6 (or GPT12)
- GTM TIM/TIM,
- GTM TOM/TIM
- GTM TOM/CCU6/IOM
- GTM TOM/TOM/IOM
- GPI/GPI
- GPO/GPO
- QSPI/QSPI
- VADC/VADC
- DSADC/VADC

Top Level Safety Requirement 4: Avoid undetected unavailability of a safety mechanism implemented by the microcontroller SEooC for longer than the multiple point fault detection interval in order to reach the target value for LFM

3.3 Assumed Conditions of Operation

[SM_AURIX_17]It is assumed that the operating conditions given in the device data sheet [4] are not exceeded.[/req]

3.4 Operating Modes

For the AURIX microcontroller, the following operating modes have been considered:

- **Completely un-powered**: No failure of the microcontroller has the potential to violate the safety goal.
- **Power-On Reset**: as long as the $\overline{\text{PORST}}$ pin is asserted, the I/O pins of the microcontroller are in their reset state as indicated in the device data sheet [4] (tri-state or internal pull-up or pull-down).
- **Reset and Startup Software** (SSW) execution
- **Pre-run safety integrity checks**: during this phase, the safety mechanisms that are required are initialized and tested
- **Normal run-time mode**: execution of the safety and non-safety related applications
- **Standby mode**: the microcontroller is not powered with the exception of the standby domain that enables to wake-up the microcontroller
- **CPU Idle mode**: The CPU code execution is halted and CPU clock is disabled. The peripherals continue to remain active. The CPU RAM memories (PSPR / DSPR) are accessible to other bus masters such as other CPUs or DMA.
- **Post-run**: Termination of all the services provided by the microcontroller. During this phase safety integrity checks may take place controlled by the application. They are call post-run safety integrity checks.
- **Explicitly indicating an internal error**: Because of an internal failure the microcontroller outputs are potentially dangerous.

SMOnly the normal run-time mode is assumed in the safety concept. Any other operating mode entered in an intended or un-intended manner shall be controlled by the system integrator and dedicated measures taken to ensure that, in this specific operating mode, the AURIX microcontroller does not have the potential to lead to a violation of a safety goal.

3.5 Fault Tolerant Time Interval

The fault tolerant time interval is defined in ISO 26262-1:2011. As this time is application dependent, the AURIX microcontroller safety concept does not assume any fault tolerant time interval.

3.6 Multiple-Point Fault Detection Interval

To avoid multiple-point faults from being latent, ISO 26262-5 [1] requires that every safety mechanism used by the application is tested at least once per multiple-point faults detection interval.

[SM_AURIX_19]It is assumed the multiple-point fault detection interval is one driving cycle. As a consequence, every hardware safety mechanism of the AURIX microcontroller shall be tested once, either at power-up, runtime, or power-down. [/req]

Assumptions about latent fault avoidance are given for each hardware safety mechanisms in section 5.1, as a list of necessary tests.

3.7 Startup

This section lists the safety assumptions on startup.

The AURIX microcontroller startup sequence can be divided into 3 main phases shown in Figure 4. Each of these phases has its own particular constraints and context related to the safety aspects. Furthermore, each phase has its own safety target(s) and fault models which need to be separately analyzed and evaluated.

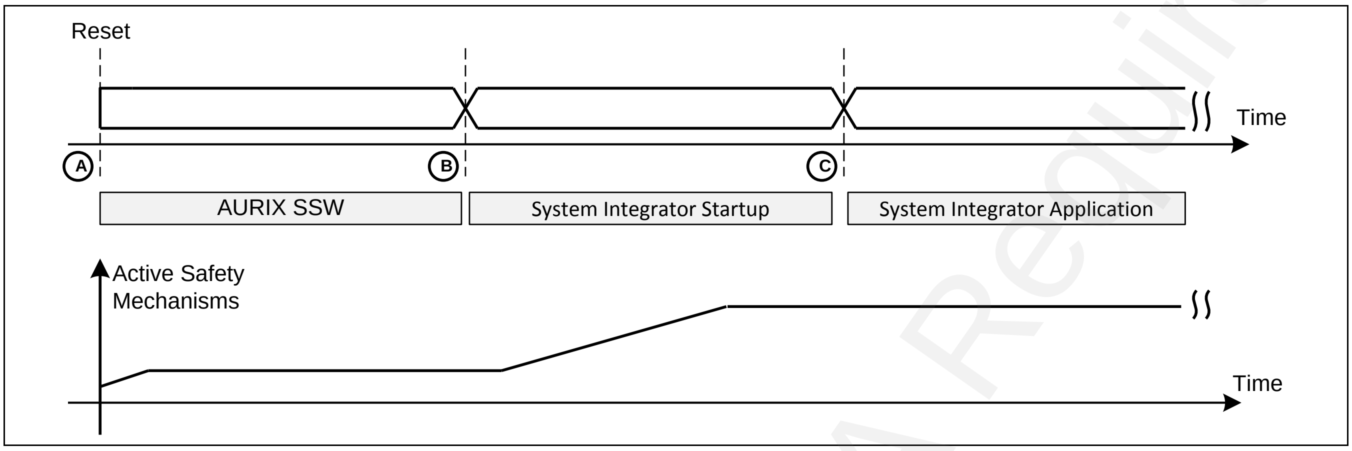

Figure 4 Microcontroller Startup Phases Overview

1. Execution of AURIX Startup Software (SSW) (A - B)
2. Execution of system Integrator specific initialization and startup (B - C)
3. Execution of system Integrator specific application (after C)

The AURIX microcontroller startup phase starts with the reset release moment (A) and continues until the first instruction of the system integrator software is executed (B). Depending on the type of reset (e.g. cold reset, warm reset, etc.); the Startup Software (SSW) will perform its intended execution with the target to bring the controller into a well-defined state at time point (B). Infineon SSW, related assumptions of use, and recommendations are described in 3.9.1.

3.7.1 Multi-Core aspects

[SM_AURIX_20]In safety related applications requiring a lockstep CPU, it is assumed that the whole startup sequence, including evaluation of the reset cause, initialization of the hardware and startup safety tests is executed on a lockstep CPU. This applies to the phase 2 (B-C) in Figure 4.

Rationale: using a lockstep CPU avoid having to consider the CPU and its side memories for the latent fault metric according to ISO26262-10 8.1.8.[/req]

Note: After a reset, multi-core devices always boot from CPU0. The other CPUs, when available, are halted. In some devices of the AURIX microcontroller family, CPU0 is not a lockstep CPU; therefore CPU0 should activate CPU1 and go to idle. The startup sequence is executed on CPU1.

3.7.2 Initialization of safety mechanisms

[SM_AURIX_22]After any power-on reset, system reset or application reset, the hardware safety mechanisms shall be initialized. The initialization includes:

- Initialization of the safety mechanism as documented for each safety mechanism in section 5.1
- Initialization of the SMU module and individual SMU alarms. By default the SMU alarms are dis-abled. [/req]

[SM_AURIX_15]It is assumed that the application initializes the relevant safety mechanisms and SMU alarms before starting the execution of the safety related software. This is before time point (C) in **Figure 4**. [/req]

[SM_AURIX_30]It is assumed that the application verifies the correct configuration of safety mechanisms and SMU alarms, after all the safety mechanisms and alarms have been configured and activated, and before starting the execution of the safety related software. This is just before time point (C) in Figure 4. [/req]

Note: By default, after a power-on reset, the error correction in FLASH and SRAMs are activated but no error is signaled to the application by the SMU as long as the corresponding SMU alarms are not configured.

3.8 Assumed Hardware Environment

This section lists the safety measures that are assumed or recommended to be implemented on separate hardware devices at the system level.

The Figure 5 shows the safety related functions and signals in the case the AURIX microcontroller is used together with an external component providing the external supply voltage and the external safety mechanisms required by the AURIX:

- External Voltage Monitor
- Error Monitor
- Watchdog
- Safe State Control

This external component could be an ASIC or a safety power supply such as the Infineon's TLF35584.

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

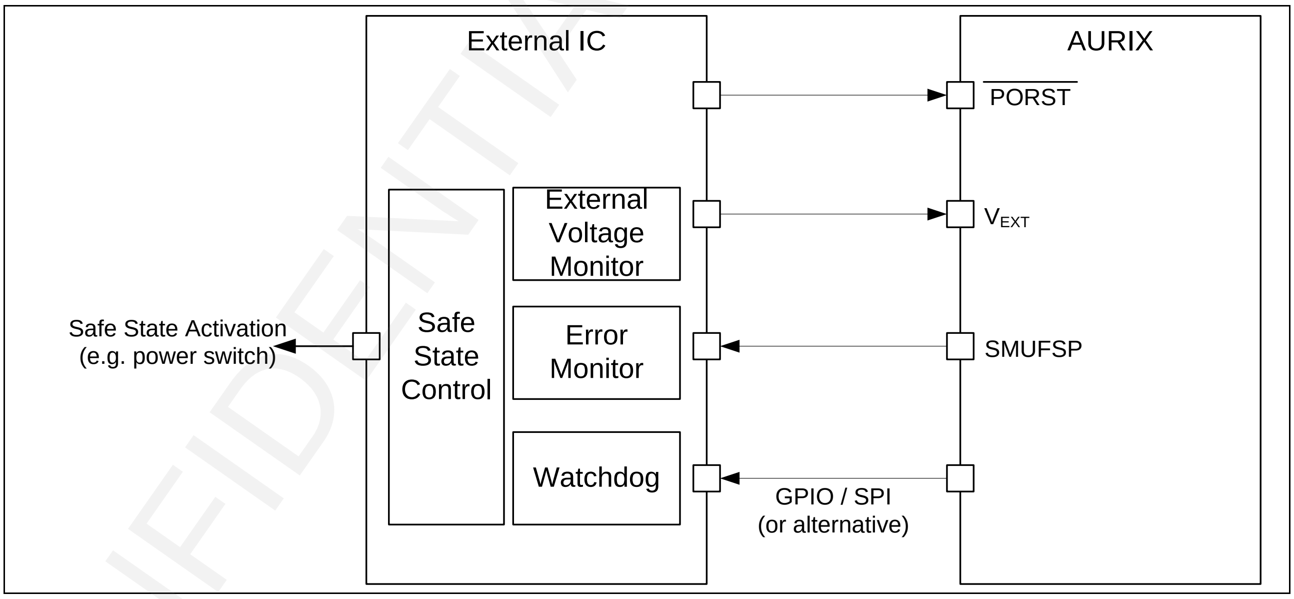

Figure 5 External Safety Mechanisms Overview

3.8.1 Power Supply and External Voltage Monitor

[SM_AURIX_01]It is assumed that the system supplies the AURIX microcontroller (VEXT) within the operating conditions given in the respective device data sheet [4].

Rationale: the reliability of the AURIX microcontroller, including its capability to detect random hardware faults, cannot be guaranteed if the voltage applied to VEXT violates the operating conditions.
[/req]

[SM_AURIX_02]It is assumed that the system can transition to and maintain its safe state in case of over-voltage, under-voltage, drift, oscillation, or power spikes above the maximum operating conditions, fault conditions on the external supply voltage, as reliable operation of the AURIX microcontroller cannot be guaranteed under these conditions. **[/req]**

[SM_AURIX_07]In case of over-voltage it is assumed that external circuitry disables the external supply to AURIX.

Rationale: in case of transient overvoltage, the system may be able to recover, but the reliability of the AURIX for further operation could be impacted if the absolute maximum ratings are exceeded. **[/req]**

[SM_AURIX_03]It is assumed that VEXT will not stay between the maximum operating conditions and the absolute maximum ratings for longer than the time given in the respective device data sheet.

Rationale: exceeding the operating conditions for longer than the specified time may lead to an increase of the microcontroller failure rate.
[/req]

Note: The AURIX microcontroller detects under-voltage on V_{EXT} and automatically enters the Cold Power-On Reset operating mode. Therefore under-voltage monitoring by an external component is not strictly necessary, although this is good practice and allows the system to trigger a reaction if needed.

Note: For TC23x, TC22x and TC21x devices of the AURIX family, the main external supply voltage is called V_{DDP3} instead of V_{EXT} in the device User's Manual [3].

3.8.2 Error Monitor

The AURIX microcontroller can signal an internal failure through the error pin (SMUFSP) as described in section 4.3.

[SM_AURIX_04]It is assumed that the SMUFSP pin is monitored by a component independent from the AURIX microcontroller.

Rationale: the SMUFSP pin provides a fast and software independent information about the presence of an internal failure of the microcontroller.

Note: When a failure condition is indicated, the system cannot rely on the microcontroller's output and should transition to its safe state. Depending on the application functionalities, the system might reset the AURIX microcontroller and try to restart the application, or completely shut it off and maintain the system safe state.

[/req]

[SM_AURIX_05]It is assumed that an external pull-down device is connected to the SMUFSP pin.

Rationale: when PORST is applied and until the FSP functionality of the SMU is initialized, SMUFSP is in high-impedance mode.
[/req]

3.8.3 Microcontroller Supervision with External Watchdog

[SM_AURIX_06]It is assumed that an external device with the functionality of a time-window watchdog supervises the AURIX microcontroller. This external device must initiate the transition to the system safe state if a fault is detected.

Rationale: This is necessary to control some failure modes of the microcontroller. Moreover, common cause failures initiated by external stress conditions or internal failures of the microcontroller, for example of the Power Management Controller controlling the power saving modes control, may lead to a state where the microcontroller cannot signal an internal failure.
[/req]

This safety mechanism is identified as SM1[AoU].ExtWDT in the rest of this document.

Dependent failure failures between the external watchdog function and the AURIX microcontroller should be avoided.

Note: 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.

3.9 Assumed Software Environment

This section describes the assumptions of use related to the software environment.

3.9.1 Startup Software

The AURIX microcontroller startup software (SSW) is located in the BootROM. This application independent software element is provided by Infineon on every device and is the very first software executed after a reset. It is in charge of the basic configuration of the microcontroller as described in the device user's manual [3] in section BootROM Content > startup software. This includes the configuration and/or activation of hardware functions as well as hardware safety mechanisms such as the lockstep CPU. No safety requirements are allocated to the SSW.

[SM_AURIX_9]It is assumed that the basic configuration performed by the SSW, including the safety mechanisms configuration, is checked by independent software during the startup sequence.
Rationale: No safety requirements are allocated to the SSW [/req]

Please refer to the following application note that lists the critical configuration steps of the SSW which may need to be checked: “AP32320 Startup Software Safety Considerations”.

3.9.2 SafeTlib

The Infineon's SafeTlib software provides:

- a software driver to configure and control the SMU
- test routines to test the error reaction capability of specific hardware safety mechanisms

- Software-Based Self-Tests for TC1.6E and TC1.6P CPU[/req]Details as well as assumptions on the integration and usage of SafeTlib are contained in the SafeTlib User's Manual [6].

All the specific test routines and Software-Based Self-Tests are referred to as Assumption of Use: Software-Based Tests, listed in section 5.2, or Assumption of Use: Software-Based Runtime Tests, listed in section 5.2.45. The mapping to the available SafeTlib functions is contained in SafeTlib release documentation.

3.9.3 Application Software

Application software is composed of safety related software elements with the same or different safety integrity levels as well as non-safety related software.

[SM_AURIX_14]It is assumed that the requirements for coexistence of elements from ISO 26262-9:2011, Clause 6, are implemented by the system integrator with support of the AURIX microcontroller hardware features supporting the freedom from interference listed in section 4.

Note: Commercial operating systems developed for safety applications usually support mechanisms such as task-level memory protection and temporal protection.[/req].

[SM_AURIX_16]It is assumed that data cache is not used for safety related variables shared across CPUs. Rational: use of data cache for shared variable can cause data coherency problems. [/req]

Note: It is recommended to use data cache for constants located in FLASH.

3.10 Debug Environment

[SM_AURIX_DEBUG_1]Debugging is supported by the On-Chip Debug Support (OCDS) module. Debug is not a normal mode of operation and it is assumed the OCDS is not enabled while safety related software is executing.

Rationale: it could interfere with the correct software execution. [/req]

Note: By default, after PORST, OCDS is disabled. It requires a sequence of four write accesses to activate it.

Note: Additionally, many modules connected to the SPB can be suspended for debug purposes. After PORST, this suspend feature is disabled. It is recommended not to change this setting (OCS.SUS).

[SM_AURIX_DEBUG_2]It is assumed no tool is connected via DAP or JTAG interface in the normal mode of operation. Rationale: any tool connected to these interfaces can strongly interfere with the normal application software execution.[/req]

4 Architecture for Management of Faults

4.1 Overview

For a safety critical development it is necessary to manage both systematic and random faults. The AURIX microcontroller product architecture includes many safety mechanisms to control random hardware faults. The AURIX safety concept is divided into three main architectural safety elements:

- Safe computation
- Safe actuation and acquisition
- Safe communication

The AURIX microcontroller also provides hardware measures to avoid or detect interferences caused by software faults or hardware faults.

Error management is centralized in the Safety Management Unit (SMU).

4.1.1 Safe Computation

Safe computation refers to the execution of safety-related software in a safe computation element. The following safety integrity measures are used in the safe computation element:

- **Hardware redundancy**: permanent and transient hardware faults in lockstep CPU are detected by real time comparison of all the functional outputs of redundant instances of the CPU. One instance is called the master CPU, and the other is called the checker CPU.
- **Software-Based Self-Test**: permanent hardware faults in non-lockstep CPU are detected by a periodic execution of software-based self-test with medium coverage.
- **Error correction and detection**: Error Correcting Codes (ECC) detects data corruption due to permanent or transient faults in Flash and SRAM memory arrays and in the path between the CPU and the memory. Error Detection Codes (EDC) also protect the communication over the SRI bus and the interrupt vector sent from the IR to the interrupt service provider (CPU or DMA).
- **Software-based tests of safety mechanisms** are generally provided to avoid dual-point faults from being latent.
- **Dependent failures** are mitigated with a mix of design measures such as diversity and physical separation and monitoring of operating conditions such as internal voltages, clocks, and temperature. Additional measures are implemented for the lockstep CPU, including delayed execution of the checker CPU by two clock cycles and other diversity measures.

4.1.2 Safe Acquisition & Actuation

The interactions with the environment are assumed to be covered by system-level safety mechanisms such as a plausibility check of sensor data against a software model or redundant information, injection of test patterns, output monitoring, and other techniques.

To support multiple channels acquisition and actuation, information related to the avoidance of dependent failures is provided in sections 4.23, 4.25, 4.26, and 7

Additionally, injection of test patterns in the VADC is supported by built-in hardware.

4.1.3 Safe Communication

[SM_AURIX_08]Safe Communication refers to the transfer of safety related data over a communication network. The integrity of the safety related data shall be ensured by safety measures at system level, such as end-to-end safe communication protocol implemented by application software. This assumption applies to all safety related communication peripherals if available in the considered device:

- E-Ray

- MultiCAN+
- QSPI
- PSI5
- SENT
- ETH
- ASCLIN
- MSC
- I2C
- HSSL[/req]

Note: An end-to-end safe communication protocol is a communication protocol that implements integrity functions to detect communication failures such as data corruption, incorrect sequence and loss of information. Standards such as EN 50129 and IEC 62280 should be considered.

4.2 Freedom from Interference

To meet the criteria for coexistence of elements specified in ISO 26262-9:2011, clause 6, as well as to prove the independence between a function and its safety mechanisms, interference between hardware/software elements must be avoided or detected. This includes interference between:

- Software components
- Software components and hardware parts
- Hardware parts

The AURIX microcontroller family supports freedom from interference in data, time and resource domain:

- Data domain: a hardware/software element corrupts information belonging to another hardware/software element.
- Time domain: a hardware/software element consumes too much CPU performance, or uses a shared resource (internal bus, peripheral, communication network) for a too long time, preventing other elements to reach their execution latency requirements.
- Resource domain: a hardware/software element uses, or modify the configuration, of a resource (internal bus, peripheral, communication network) belonging to another application.

4.2.1 Data domain

To implement freedom from interference between software components in the data domain, memory partitioning is generally used. In the AURIX microcontroller, this is supported by the Memory Protection Unit (MPU) of the TriCore CPU also identified in this document as **SM1[HW].CPU.DATA:MPU** and **SM1[HW].CPU.CODE:MPU**.

Moreover, CPU side memories are also accessible to all bus masters such as other CPUs and DMA. To avoid interference from other bus masters, each shared memory implements a region access protection mechanism. This mechanism is identified in this document as **SM1[HW].CPU.BUS:MPU**. This mechanism should be used together with **SM1[HW].CPU:STI**.

4.2.2 Time domain

The AURIX microcontroller provides a set of hardware functions that enable the application to verify that the static timing properties of the safety related tasks are met during run-time.

To support this monitoring, AURIX provides internal watchdogs. There is one internal watchdog per CPU, plus one Safety watchdog. This mechanisms are identified in this document as **SM1[HW].WDT** and **SM1[HW].SWDT**.

The TriCore CPU also implements a temporal protection system to guard against task runtime overrun. This mechanism is identified in this document as **SM1[HW].CPU:TPS**.

4.2.3 Resource Domain

To control the access to their resources, and provide freedom from interference in the resource domain, all modules implement a register access protection, identified in this document as **SM1[HW].<module>:AP**. This mechanism should be used together with **SM1[HW].CPU:STI**. Additionally, software access to peripherals can also be controlled by the **SM1[HW].CPU.DATA:MPU**.
The TriCore privilege levels control the access to peripheral devices and SFRs. Each task is allocated a privilege level supervisor, or User-1, or User-0. This mechanism is identified in this document as **SM1[HW].CPU:SV**, **SM1[HW].CPU:USER1** and **SM1[HW].CPU:USER0**, and used in combination with **SM1[HW].Register:SV**.

[SM_AURIX_ISR_1]Special care shall be taken by the application to isolate hardware interrupt service routines (ISR) if they are not developed with the highest of the ASIL levels coexisting in the AURIX microcontroller

Rationale: a hardware interrupt service routine (ISR) when started has the supervisor privilege per default; this is controlled by the CPU hardware. A task/ISR in supervisor mode has no restrictions with respect to the use and control of all vital hardware resources and execution of critical, supervisor only, instructions.[/req]

The Memory Protection Unit (MPU) can be used to control accesses to resources and the execution the critical instruction. Per default, the ISR is mapped to the protection register set 0. The MPU is referenced as **SM1[HW].CPU.DATA:MPU** and **SM1[HW].CPU.CODE:MPU**.

The CPU End-of-Initialization signal, ENDINIT, is a global hardware signal used by some parts to provide a protection of the write accesses to its register. It is usually used to protect access to configuration register. A write access to the ENDINIT protected registers is only possible if the signal is active. The ENDINIT signal is activated by writing a special sequence to one of the CPU watchdogs by software. This is referenced as **SM1[HW].WDT:CE**.

In a similar way, a safety End-of-Initialization signal, called Safety ENDINIT, is activated by writing a special sequence to the safety watchdog by software. This signal is used to protect write accesses to some safety mechanisms configuration registers. This is referenced as **SM1[HW].SWDT:SE**.

4.2.4 Critical Parts

[SM_AURIX_FFI_3]To avoid interferences from the critical parts, it is assumed the register access protection and bus memory protection mechanisms are configured to avoid any unwanted access to memory regions or registers by critical parts. It involves correctly configuring following safety mechanisms:

- **SM1[HW].CPU.BUS:MPU**
- **SM1[HW].LMU.BUS:MPU**
- **SM1[HW].CPU:AP**
- **SM1[HW].LMU:AP**
- **SM1[HW].<module>:AP**
- **SM1[HW].SRI:AP**

Note: Critical parts are listed in Table 3

Note: The OCDS, for example, is supposed to be used only for debugging and it should be taken care of by the system integrator that it is not allowed write access to any resource during normal operation. [/req]

4.2.5 Data Access Overlay

The data overlay provides the capability to redirect selected data accesses to the Overlay memory. Data accesses made by the TriCore to Program Flash, Online Data Acquisition space, or External Bus Unit space can be redirected. This is described in more details in the section Data Access Overlay (OVC) of the User's Manual [3].

[SM_AURIX_OVC_1]The overlay feature should only be used for calibration applications where neither the original data nor the overlay data in the defined region would present a potential to violate any safety goal. If this cannot be assured then the overlay feature must not be configured for use, or shall be used in such a way that the overlay data is also safe so an unexpected overlay switch cannot cause harm. Rationale: the global overlay control registers in the SCU are not monitored by hardware; therefore a silent corruption is possible that would activate/deactivate the overlay. If the overlay is in its default configuration after reset, a single fault cannot cause its activation. [/req]

4.3 Safety Management Unit (SMU)

4.3.1 Error Management Concept

In most of the cases a fault affecting the microcontroller can be diagnosed and different actions can be performed. For that purpose a dedicated part is available: the Safety Management Unit (SMU) that enables the system integrator to implement multiple error handling strategies. **Figure 6** shows a function block of the SMU and its interfaces to react to an error and control the transition to the safe state.

When a built-in hardware safety mechanism detects a fault it issues an event called alarm. Additionally, safety mechanisms implemented in software can also trigger an alarm using the dedicated interface of the SMU. The SMU processes hardware and software alarms in the same way. For each individual alarm the SMU can trigger an internal and/or external action as configured by the system integrator.

The internal action is one of the following options:

- .SMU:Reset]Request an application or system reset (microcontroller internal reset)
- .SMU:Idle]Force a configurable set of CPUs into idle mode (CPU isolation)[1]
- .SMU:NMI]Issue a NMI to all CPUs
- .SMU:IRQ]Issue an interrupt to selectable CPUs

In case of an NMI or interrupt, it is recommended to use one of the SMU recovery watchdog timers to monitor the duration of the alarm handling by the software. The SMU can be configured to start automatically a recovery watchdog timer for a given alarm.

[1] Standalone CPU reset is not supported.

The external action is the activation of the error pin running one of the following Fault Signaling Protocol (FSP):

- Bi-stable protocol (default)
- Time-switching protocol[/req]

[SM_AURIX_SMU_4]If the bi-stable protocol is used, the PORT Input/Output Control register corresponding to the SMUFSP pin shall be monitored cyclically. This is not necessary if the time-Switching protocol is used, because of its dynamic properties.
Rationale: the PORT registers of the error pin have no built-in hardware monitoring against soft errors.[/req]

Additionally, The FSP signal is dependent on the SPB clock f_{SPB}. Because of this dependency, a failure of f_{SPB} may also prevent the FSP signal from changing state. Therefore it is recommended to use the time-switching protocol.

[SM_AURIX_SMU_5]The Error Pin (SMUFSP) shall be configured to output with push-pull characteristics.
Rational: if the output multiplexer has a fault and accidentally switches from SMU mode to GPIO mode, the GPIO low state forces the system into a safe state.[/req]

Additional information for the integration and usage of the error pin are given in the section SMU Integration Guidelines in the AURIX user's manual [3].

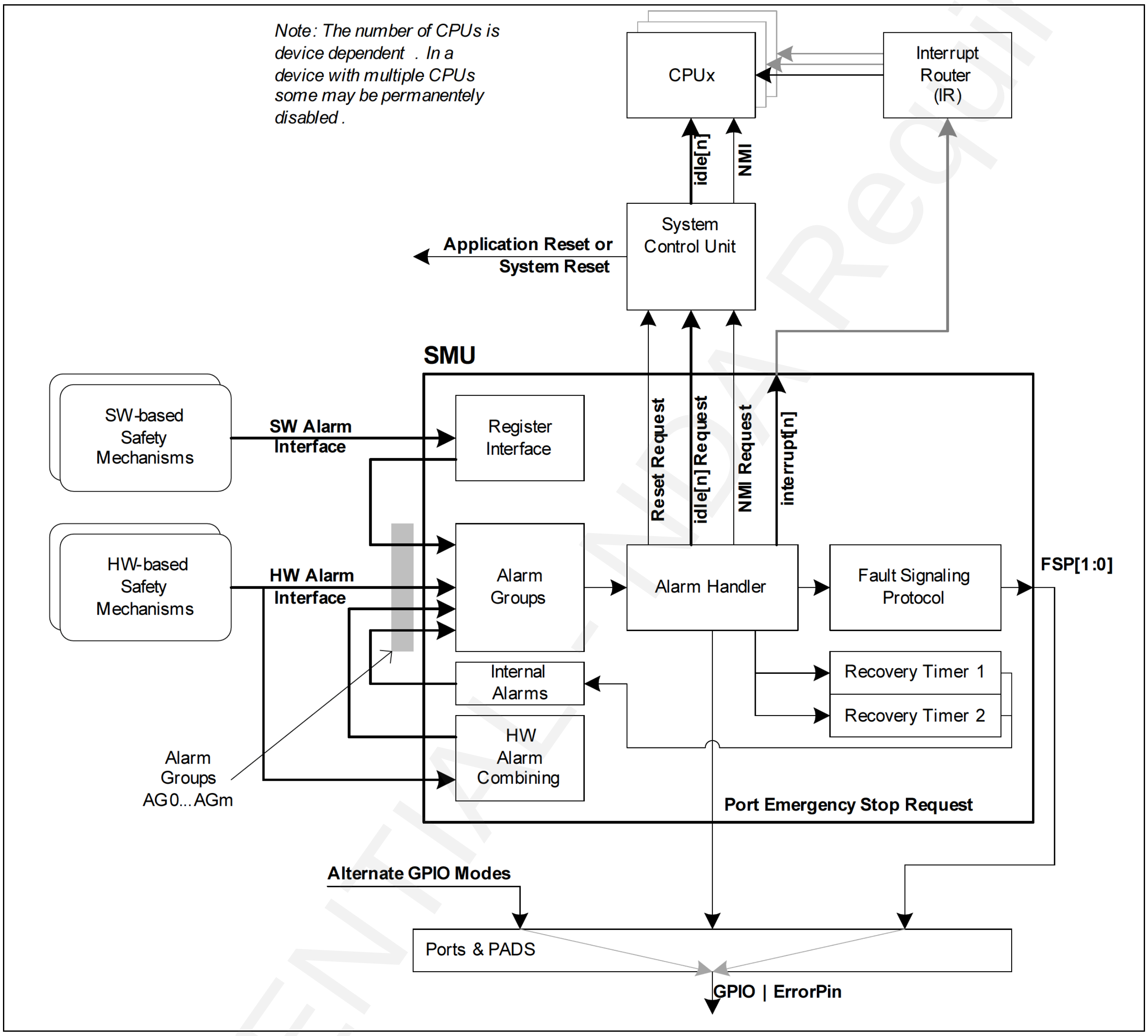

Figure 6 SMU and Safe State Control

In addition, the SMU can be configured to request a port emergency stop when the Fault Signaling Protocol is activated because of an alarm. The port emergency stop feature of AURIX is described in the user's manual [3] in the section Emergency Stop Output Control. This enables for instance to stop communication protocols like FlexRay^{TM} to avoid communication failures or to isolate dedicated outputs. A port emergency stop can also be triggered by an external event on the emergency stop input pin (see [4] Emergency Stop Function). When an external emergency stop event is detected, the SMU alarm ALM3[29] is also generated.

Reset, NMI, emergency stop and idle requests issued by the SMU are handled by the System Control Unit (SCU)

4.3.2 SMU access protection

[SM_AURIX_SMU_3]To avoid the corruption of the SMU configuration, the access to the SMU configuration registers is protected. The following safety mechanisms shall be used to ensure freedom from interference:

- **SM1[HW].SMU:LOCK**
- **SM1[HW].SMU:AP**[/req]

4.3.3 Testing of the SMU

[SM_AURIX_SMU_2]The following tests shall be implemented by application software and executed once at startup:

- **SM2[AoU].SMU:InitCheck**
- **SM2[AoU].SMU.LOCK:TEST**
- **SM2[AoU].SMU.FSP:TEST**
- **SM2[AoU].SMU.ALARM:TEST**
- **SM2[AoU].SMU:TEST**
- **SM2[AoU].SMU.RT:TEST**
- **SM2[AoU].SMU:AP**[/req]

4.4 Lockstep CPU

Permanent random hardware faults such as register stuck-at and transient random hardware faults such as a soft error caused by a neutron or alpha particle may cause an incorrect output of the CPU. The safety mechanisms listed in this section mitigate these errors.

4.4.1 Monitoring Concept

The 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 is illustrated in Figure 7.

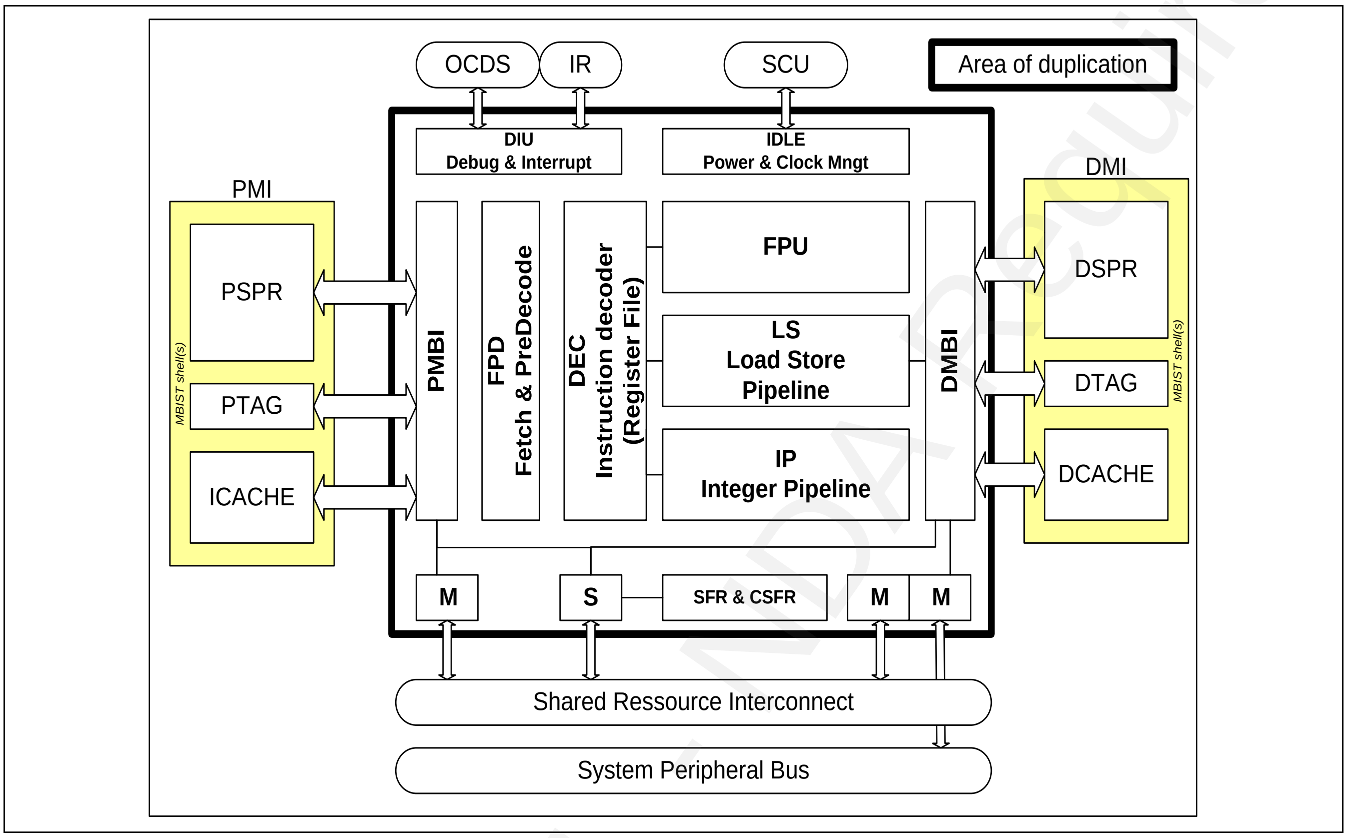

Figure 7 CPU area of duplication

The following sub-parts are in the area of duplication of a lockstep CPU:

- The TriCore TC16E or TC16P core
- The CPU SFR and CSFR
- The master and slave interfaces to SRI
- The master interface to SPB
- The interface to the interrupt router
- The interface to the SCU
- The interface to the program memories (PMI)
- The interfaces to the data memories (DMI)

4.4.2 Hardware Safety Mechanisms

[SM_AURIX_CPU_1]It is assumed that the following safety mechanisms, implemented in hardware, are used:

- **SM1[HW].CPU:LOCKSTEP**
- **SM2[HW].CPU:LOCKSTEP.SCC**
- **SM1[HW].CPU.TRAP:IOPC**

[/req]

[SM_AURIX_CPU_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].CPU:SV**

- **SM1[HW].CPU:USER1**
- **SM1[HW].CPU:USER0**
- **SM1[HW].CPU:AP**
- **SM1[HW].CPU.BUS:MPU**
- **SM1[HW].CPU.DATA:MPU**
- **SM1[HW].CPU.CODE:MPU**[/req]

4.5 Non-Lockstep CPU

Permanent random hardware faults such as register stuck-at and transient random hardware faults such as soft error caused by a neutron or alpha particle may cause an incorrect output of the CPU. The safety mechanisms listed in this section mitigate these errors.

4.5.1 Monitoring Concept

The permanent random hardware faults are monitored by a software-based self-test (SBST). Transient random hardware faults in a non-lockstep CPU can be detected by software-level safety mechanisms such as redundant software execution.

Attention: *[SM_AURIX_NLSCPU_4]* ***The Software-Based Self-Test provided by Infineon for the non-lockstep CPU reaches 90% diagnostic coverage of permanent faults on the Fetch and Pipeline Unit of the CPU core. For higher diagnostic coverage, a lockstep CPU must be used. For more details about the scope of the Software-Based Self-Test provided by Infineon please refer to the FMEDA [7] and to the SBST User Manual [8].*** *[/req]*

Attention: *[SM_AURIX_NLSCPU_5]* ***The hardware safety mechanisms together with the software-based Self-Test,*** *SM1[AoU].CPU:SBST* ***and the assumptions of use listed in the SBST User Manual [8] enable reaching the SPFM and LFM targets for ASIL B with regard to permanent failures. It is the responsibility of the system integrator to analyze the influence of transient failures and provide the necessary safety mechanisms.*** *[/req]*

4.5.2 Hardware Safety Mechanisms

[SM_AURIX_NLCPU_3]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].CPU:SV**
- **SM1[HW].CPU:USER1**
- **SM1[HW].CPU:USER0**
- **SM1[HW].CPU:AP**
- **SM1[HW].CPU.BUS:MPU**
- **SM1[HW].CPU.DATA:MPU**
- **SM1[HW].CPU.CODE:MPU**[/req]

4.5.3 Runtime Tests

This section lists the necessary runtime tests:

- **SM1[AoU].CPU:SBST**

[SM_AURIX_NLCPU_2]If it is necessary for a considered application to control transient failures in a non-lockstep CPU, the following safety mechanism shall be implemented by application software:

- **SM1[AoU].CPU:SoftErrorMon**[/req]

4.6 Power Management Controller (PMC)

Permanent and transient hardware faults in the Power Management Control may cause the unintended selection of CPU idle mode, or of the microcontroller sleep or standby mode. The safety mechanisms listed in this section mitigate these errors.

4.6.1 Monitoring Concept

The PMC monitoring is based on temporal monitoring by an independent watchdog.

4.6.2 System-Level Safety Mechanisms

This section lists the necessary safety mechanisms at system level:

- **SM1[AoU].WDT:TIMEOUT**
- **SM1[AoU].ExtWDT**

4.7 Embedded Voltage Regulators (EVR)

Permanent faults and transient faults affecting the intended EVR operation may cause an incorrect voltage level of the internal core voltages. The safety mechanisms listed in this section mitigate these errors. Support to detect over/under-voltage of the external power supply is also provided.

4.7.1 Monitoring Concept

The monitoring concept is based on embedded voltage monitors independent from the voltage generators. The voltage monitors measure continuously the voltage level produced by the voltage regulator.

4.7.2 Hardware Safety Mechanisms

[SM_AURIX_EVR_1]It is assumed that the following safety mechanisms, implemented in hardware, are used:

- **SM1[HW].EVR13:MON**
- **SM1[HW].EVR33:MON**
- **SM1[HW].BANDGAP:MON** [/req]

Please refer to the following application note that explains how to set the threshold levels of the secondary voltage monitors: "AP32329 AURIX™ Secondary Voltage Monitors - Setting threshold levels".

The following hardware safety mechanisms may be used to support system-level safety mechanisms:

- **SM1[HW].ExtVREG:MON**

[SM_AURIX_EVR_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].SCU:AP**[/req]

4.7.3 Runtime Tests

This section lists the necessary runtime tests:

- SM1[AoU].EVR:CFGMON

4.8 Reset Control Unit (RCU)

Permanent and transient hardware faults in the Reset Control Unit may cause, for example, unintended reset, or no reset.

4.8.1 Monitoring Concept

The RCU monitoring is based on hardware redundancy and system-level safety mechanisms.

4.8.2 Hardware Safety Mechanisms

[SM_AURIX_RCU_1]It is assumed that the following safety mechanisms, implemented in hardware are used:

- **SM1[HW].Register:SFF**[/req]

[SM_AURIX_RCU_3]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].SCU:AP**[/req]

4.8.3 System-Level Safety Mechanisms

This section lists the necessary safety mechanisms at system level:

- **SM2[AoU].RESET:SSCheck**

4.9 Clocking System

Permanent and transient hardware faults of the clocking system may cause a wrong frequency of the clocks signals generated by the clock generation unit (CGU) or of the PLL signals. The safety mechanisms listed in this section mitigate these errors.

4.9.1 Monitoring Concept

The monitoring concept is based on frequency monitors and the alive supervision of the AURIX by an external watchdog.

4.9.2 Hardware Safety Mechanisms

[SM_AURIX_CCU_1]It is assumed that the following safety mechanisms, implemented in hardware are used:

- **SM1[HW].SystemPLL:CLKMON**
- **SM1[HW].SystemPLL:LOCKMON**
- **SM1[HW].Register:SFF**[/req]

Note: AURIX implements frequency monitors for all the clocks derived from the system PLL. These safety mechanisms are listed in the corresponding module sections.

[SM_AURIX_CCU_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].SCU:AP** [/req]

4.10 Watchdog Timers

Some errors caused by permanent or transient faults in the microcontroller, environmental conditions such as electromagnetic interferences (EMI) or high temperature, as well as interferences between hardware or software elements, are likely to have an impact on timing and execution of a program sequence.

4.10.1 Monitoring Concept

Such errors can be detected by a combination of alive supervision with an external watchdog, and time and sequence based monitoring of the safety related software execution.

4.10.2 System-Level Safety Mechanisms

[SM_AURIX_WDT_1]The following safety mechanisms shall be implemented at system level with support of external hardware components or AURIX internal watchdog timers:

- **SM1[AoU].ExtWDT**
- **SM1[AoU].WDT:TIMEOUT**
- **SM1[AoU].WDT:PFM**[/req]

4.11 Shared Resource Interconnect (SRI)

The SRI Crossbar is a communication channel: permanent and transient hardware faults may corrupt data and address signals or the control logic, including the master and slave interfaces.

4.11.1 Monitoring Concept

The monitoring of the SRI Crossbar is based on a combination of information redundancy, frame counter and timeout monitoring.

4.11.2 Hardware Safety Mechanisms

[SM_AURIX_SRI_1]It is assumed that the following safety mechanisms, implemented in hardware, are used:

- **SM1[HW].SRI.ADDR:EDC**
- **SM1[HW].SRI.DATA:EDC**
- **SM1[HW].SRI:EH**
- **SM1[HW].SRI:CLKMON**[/req]

[SM_AURIX_SRI_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].SRI:AP** [/req]

4.12 System Peripheral Bus (SPB)

The SPB is a communication channel: permanent and transient hardware faults may corrupt data and address signals or the control logic, including the master and slave interfaces.

4.12.1 Monitoring Concept

The monitoring of the SPB is based on a combination of system level safety mechanisms and timeout monitoring.

4.12.2 Hardware Safety Mechanisms

[SM_AURIX_SPB_1]It is assumed that the following safety mechanisms, implemented in hardware, are used:

- **SM1[HW].SPB:CLKMON**
- **SM1[HW].SPB:EH**
- **SM1[HW].SPB:Timeout**[/req]

[SM_AURIX_SPB_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].SBCU:AP**[/req]

4.12.3 Runtime Tests

This section lists the necessary runtime tests:

- **SM1[AoU].SPB:TEST**

4.13 Lockstep CPU SRAM

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. The Safety Mechanisms described in this section mitigate these errors.

The SRAM instances considered in this section are identified in Table 4. The interfaces, including ECC encoding and decoding, of the lockstep CPU SRAM instances are part of the area of duplication.

Table 4 Lockstep CPU SRAM instances

SRAM	TC29x	TC27x	TC26x	TC23x/TC22x
CPU2.DSPR	-	-	N/A	N/A
CPU2.DCACHE	-	-	N/A	N/A
CPU2.DTAG	-	-	N/A	N/A
CPU2.PSPR	-	-	N/A	N/A
CPU2.PTAG	-	-	N/A	N/A
CPU2.PCACHE	-	-	N/A	N/A
CPU1.DSPR	☐	☐	☐	N/A
CPU1.DCACHE	☐	☐	☐	N/A
CPU1.DTAG	☐	☐	☐	N/A

SRAM	TC29x	TC27x	TC26x	TC23x/TC22x
CPU1.PSPR	☐	☐	☐	N/A
CPU1.PCACHE	☐	☐	☐	N/A
CPU1.PTAG	☐	☐	☐	N/A
CPU0.DSPR	-	☐	-	☐
CPU0.DCACHE	-	N/A	N/A	N/A
CPU0.DTAG	-	N/A	N/A	N/A
CPU0.PSPR	-	☐	-	☐
CPU0.PCACHE	-	☐	-	☐
CPU0.PTAG	-	☐	-	☐

☐	Lockstep CPU SRAM
-	Non Lockstep CPU SRAM
N/A	SRAM not implemented in these devices

4.13.1 Monitoring concept

The SRAM monitoring is based on error detection and correction codes.

4.13.2 Hardware Safety Mechanisms

[SM_AURIX_CPU_3]It is assumed that the following safety mechanisms, implemented in hardware are enabled for all safety related SRAM modules:

- **SM1[HW].SRAM:ECC**
- **SM1[HW].SRAM:EDC**
- **SM1[HW].SRAM:ADDRMON**
- **SM1[HW].SRAM.ECC:LOCKSTEP** [/req]

[SM_AURIX_CPU_4]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].CPU.BUS:MPU**
- **SM1[HW].CPU.DATA:MPU**
- **SM1[HW].CPU.CODE:MPU**
- **SM1[HW].MTU:AP**[/req]

4.14 Non-Lockstep CPU SRAM

Data in SRAM may be corrupted by permanent hardware faults such as bit stuck-at and by 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. The Safety Mechanisms described in this section mitigate these errors.

4.14.1 Monitoring concept

The SRAM monitoring is based on error detection and correction codes.

[SM_AURIX_SHARED_SRAMS]If the side memories (DSPR, PSPR) of a non-lockstep CPU are used as shared memory to store safety related data used by application software running on a lockstep CPU, the user shall consider the uncovered failure modes, according to the FMEDA results [7], and accordingly implement system level safety mechanisms.

Rationale: the main difference between the Non-Lockstep CPU SRAM and the Lockstep CPU SRAM is that the interfaces, including ECC encoder/decoder, of Non-Lockstep CPU SRAM instances are not monitored by hardware as they are not part of the area of duplication of a lockstep CPU[/req]. The Non-Lockstep CPU SRAM instances are identified in Table 4.

4.14.2 Hardware Safety Mechanisms

[SM_AURIX_NLCPU_4]It is assumed that the following safety mechanisms, implemented in hardware are enabled for all safety related SRAM modules:

- **SM1[HW].SRAM:ECC**
- **SM1[HW].SRAM:EDC**
- **SM1[HW].SRAM:ADDRMON**[/req]

[SM_AURIX_NLCPU_5]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].CPU.BUS:MPU**
- **SM1[HW].CPU.DATA:MPU**
- **SM1[HW].CPU.CODE:MPU**
- **SM1[HW].MTU:AP** [/req]

4.15 Local Memory Unit (LMU) SRAM

The LMU is a module connected to the SRI bus, providing access to shared volatile memory (SRAM).

Note: The LMU shared SRAM, identified in [3] as LMURAM, is only available in TC27x, TC29x and TC23x ADAS series.

4.15.1 Monitoring Concept

The SRAM monitoring is based on error detection and correction codes covering the complete path between the SRI bus interface and the SRAM.

4.15.2 Hardware Safety Mechanisms

[SM_AURIX_LMU_1]It is assumed that the following safety mechanisms, implemented in hardware are enabled for all safety related SRAM modules:

- **SM1[HW].SRAM:ECC**
- **SM1[HW].SRAM:ADDRMON**
- **SM1[HW].LMU.DP:EDC**
- **SM1[HW].LMU.SRAM.ECC:MONITOR**[/req]

[SM_AURIX_LMU_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].LMU.BUS:MPU**
- **SM1[HW].LMU:AP**
- **SM1[HW].MTU:AP**
- **SM1[AoU].LMU:Interference**[/req]

4.15.3 Runtime Tests

This section lists the necessary runtime tests:

- **SM1[AoU].LMU.DP:TEST**

4.16 Peripheral Modules SRAM

Data in SRAM may be corrupted by permanent hardware faults such as bit stuck-at and by 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. The Safety Mechanisms listed in this section mitigate these errors.

The SRAM instances considered in this section are listed in Table 5.

Table 5 Safety-Relevant Peripherals SRAM Instances

SRAM	TC29x	TC27x	TC26x	TC23x/TC22x
ETHERMAC	☐	☐	☐	☐
GTM FIFO0	☐	☐	☐	N/A
GTM MCS0	☐	☐	☐	N/A
GTM MCS1	☐	☐	☐	N/A
GTM DPLL RAM1A	☐	☐	☐	N/A
GTM DPLL RAM1B	☐	☐	☐	N/A
GTM DPLL RAM2	☐	☐	☐	N/A
PSI5	☐	☐	☐	N/A
CAN0	☐	☐	☐	☐
CAN1	☐	N/A	N/A	☐
ERAY0 OBF	☐	☐	☐	☐
ERAY0 IBF_TBF	☐	☐	☐	☐
ERAY0 MBF	☐	☐	☐	☐
ERAY1 OBF	☐	N/A	N/A	N/A
ERAY1 IBF_TBF	☐	N/A	N/A	N/A
ERAY1 MBF	☐	N/A	N/A	N/A
DMA	☐	☐	☐	N/A

☐	SRAM implemented in the device
N/A	SRAM not implemented in the device

4.16.1 Monitoring concept

The peripheral modules are application dependent parts, assumed to be monitored by application-level safety mechanisms, which also include the SRAM in the peripheral modules. SRAM hardware safety mechanisms such as ECC and address monitor may be used if required by the application.

4.16.2 Hardware Safety Mechanisms

The following safety mechanisms, implemented in hardware, may be used if required by the application:

- **SM1[HW].SRAM:ECC**
- **SM1[HW].SRAM:EDC**
- **SM1[HW].SRAM:ADDRMON**

4.17 Program Memory Unit (PMU) and Program Flash (PFLASH)

Permanent and transient hardware faults may cause the corruption of a PFLASH cell or the incorrect addressing to a PFLASH word line. The safety mechanisms listed in this section mitigate these errors.

[SM_AURIX_PMU_3]It is assumed that PFlash is not programmed or erased during the safety related application run-time. Rationale: doing so has side effects on the normal execution of the application as the PFlash bank is busy and unreadable, but also error messages, for example caused by reading erased PFlash regions, can propagate to the system.[/req]

[SM_AURIX_PMU_4]It is assumed that the PFLASH content is valid after programming, which means failure during programming are detected by check after programming including reading back the programmed content, and checking the Flash Status Register (FSR) for errors during programming, especially PVER, and OPER bits. CRC check at every startup is also recommended as stated in section 5.3.13. Aborted Flash processes are assumed to be detected by the programming tool.

Rationale: ECC rely on reasonable PFLASH content.[/req]*Please also consider the hints given by the user's manual [3], section 10.8.4.1*

4.17.1 Monitoring Concept

The PFLASH monitoring is based on error detection and correction codes.

4.17.2 Hardware Safety Mechanisms

[SM_AURIX_PMU_1]It is assumed that the following safety mechanisms, implemented in hardware, are used:

- **SM1[HW].PFLASH:ECC**
- **SM1[HW].PFLASH:ADDRMON**
- **SM1[HW].PFLASH.ECC:Monitor**
- **SM2[HW].PFLASH.EDC:CMP**
- **SM1[HW].CPU.TRAP:IOPC** [/req]

[SM_AURIX_PMU_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].PMU:AP**[/req]

4.18 Direct Memory Access (DMA)

Permanent and transient hardware faults in the DMA part may cause data corruption and addressing errors. The safety mechanisms listed in this section mitigate these errors.

Attention: [SM_AURIX_DMA_4]***In TC29x, TC27x, and TC26x series, the DMA channel transaction control set is stored in SRAM, therefore section 4.16 shall also be taken into account.***[/req]

Attention: [SM_AURIX_DMA_5]***In TC23x, TC22x, and TC21x series the DMA channel transaction control set is stored in registers instead of SRAM like for the other series. For these devices, the increased failure rate, related to faults in transaction control set registers, shall be considered by the application. Rationale: in a register-based DMA, the transaction control set of each channel is not monitored by ECC as it is in a SRAM-based DMA.***[/req]

4.18.1 Monitoring Concept

The DMA monitoring is based on error detection codes and other system-level safety mechanisms

4.18.2 Hardware Safety Mechanisms

[SM_AURIX_DMA_1]The following hardware safety mechanisms shall be used:

- **SM1[HW].DMA:EDC**[/req]

The following hardware safety features may be used as support for system-level safety mechanisms:
- **SM1[HW].DMA.DATA:CRC32**
- **SM1[HW].DMA.ADDR:CRC32**
- **SM1[HW].DMA:TIMESTAMP**

[SM_AURIX_DMA_2]The following hardware safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:
- **SM1[HW].DMA:AP**
- **SM1[HW].DMA:DMTI**
- **SM1[HW].DMA:MSEL**[/req]

4.18.3 System-Level Safety Mechanisms

This section lists the necessary safety mechanisms at system level:
- **SM1[AoU].DMA:Monitor**

4.19 System Timer (STM)

The microcontroller's STM is designed for global system timing applications requiring both high precision and long range. There is a dedicated system timer per CPU.

Permanent and transient hardware faults in the STM may corrupt the timer value or the interrupt generation.

4.19.1 Monitoring Concept

The STM is not monitored by hardware - it is not part of the area of duplication of a lockstep CPU. The monitoring concept is based on plausibility check using an independent hardware timer.

4.19.2 Hardware Safety Mechanisms

[SM_AURIX_STM_1]It is assumed that the following safety mechanisms, implemented in hardware, are used

- **SM1[HW].STM:CLKMON**[/req]

[SM_AURIX_STM_2]The following hardware safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].STM:AP**[/req]

4.19.3 System-Level Safety Mechanisms

This section lists the necessary safety mechanisms at system level:

- **SM1[AoU].STM:Plausibility**

4.20 Interrupt Router (IR)

Permanent and transient hardware faults in the interrupt router module may cause errors such as “No interrupt”, “unintended interrupt” or “wrong interrupt”. The safety mechanisms listed in this section mitigate these errors.

4.20.1 Monitoring Concept

The monitoring concept is based on error detection codes to detect the corruption of a service request node (SRN) and application dependent safety mechanisms to detect other errors “No interrupt” or “wrong interrupt”.

4.20.2 Hardware Safety Mechanisms

[SM_AURIX_IR_1]It is assumed that the following safety mechanisms, implemented in hardware, are used:

- **SM1[HW].IR:EDC**[/req]

[SM_AURIX_IR_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].IR:AP**[/req]

4.20.3 Runtime Tests

This section lists the runtime tests necessary to cover the failure modes of the interrupt router which cannot be controlled by **SM1[HW].IR:EDC**:

- **SM1[AoU].IR:Monitor**
- **SM1[AoU].IR:SrcCheck**

4.21 Controller Area Network (CAN)

Permanent and transient hardware faults in the CAN module may cause communication errors, including corrupted or delayed data, and other typical communication errors.

Attention: *[SM_AURIX_CAN_4]* ***SRAM is implemented in the CAN module, therefore section 4.16 shall also be taken into account.*** *[/req]*

4.21.1 Monitoring Concept

The CAN monitoring concept is based on end-to-end safe protocol.

4.21.2 Hardware Safety Mechanisms

[SM_AURIX_CAN_1]The following hardware safety mechanisms shall be used:

- **SM1[HW].SPB:CLKMON**[/req]

[[SM_AURIX_CAN_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].CAN:AP**[/req]

4.21.4 System-Level Safety Mechanisms

This section lists the necessary safety mechanisms at system level:

- **SM1[AoU].CAN:SafeProtocol**

4.22 E-Ray

Permanent and transient hardware faults in the E-Ray module may cause communication errors, including corrupted or delayed data, and other typical communication errors.

Attention: *[SM_AURIX_ERAY_4]* ***SRAM is implemented in the E-Ray module, therefore section 4.16 shall also be taken into account.*** *[/req]*

4.22.1 Monitoring Concept

The E-Ray monitoring concept is based on end-to-end safe protocol.

4.22.2 Hardware Safety Mechanisms

[SM_AURIX_ERAY_1]The following hardware safety mechanisms shall be used:

- **SM1[HW].ErayPLL:CLKMON**
- **SM1[HW].ErayPLL:LOCKMON**[/req]

[SM_AURIX_ERAY_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].ERAY:AP**[/req]

4.22.4 System-Level Safety Mechanisms

This section lists the necessary safety mechanisms at system level:

- **SM1[AoU].ERAY:SafeProtocol**

4.23 QSPI

Permanent and transient hardware faults in the QSPI module may cause communication errors, including corrupted or delayed data, and other typical communication errors.

[SM_AURIX_QSPI_4]This section focuses on the internal failures of the AURIX QSPI modules; faults in external components shall be considered by the system integrator.[/req]

4.23.1 Monitoring Concept

As the QSPI modules are application dependent parts for which there are no application independent safety mechanisms, specific application scenarios have to be assumed to explain specific deployment needs. For the purpose of illustration and explanation this section assumes two possible application scenarios consisting of:

1. Recommended: One QSPI instance is used to communicate with external elements using an end-to-end safe protocol as described in **SM1[AoU].QSPI:SafeProtocol**.
2. .QSPI:Redundancy]Alternative: two QSPI communication channels used redundantly to read values from two sensors measuring the same or a correlated physical value. The underlying considerations for this application scenario are given in **4.23.2**. These underlying considerations are valid for other usage scenarios in an equivalent way.

4.23.2 Redundant QSPI Communication Channels

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 8 depicts the hardware of the chosen application scenario of two redundant QSPI communication channels in absence of any additional application dependent safety measures..

The redundant channels depicted in green and blue in this hardware diagram represent the mission signals "S1" and "S2". The channels are henceforth referred to as the S1 and S2 channels.The basic assumption is that any faults in the S1 or S2 channel can be detected by a combination of simple measures:

- Comparison of results (period, digital result) within the Fault Tolerant Time of the application
- System level Plausibility checks on result value bounds and conversion rates
- Two redundant sensors deliver two different results for the same measured physical value

[SM_AURIX_QSPI_DFI_1]The AURIX microcontrollers do not provide full intrinsic independence between acquisition channels. The registers and paths marked in red text in **Figure 8** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

The potential dependent failure initiators for this scenario are divided into three classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 8**, and in section **7.3** where they are described
- Dependent Failure Initiators specific to the redundant QSPI/QSPI scenario identified as QSPIxyz in **Figure 8**, and in **Table 6** where they are described

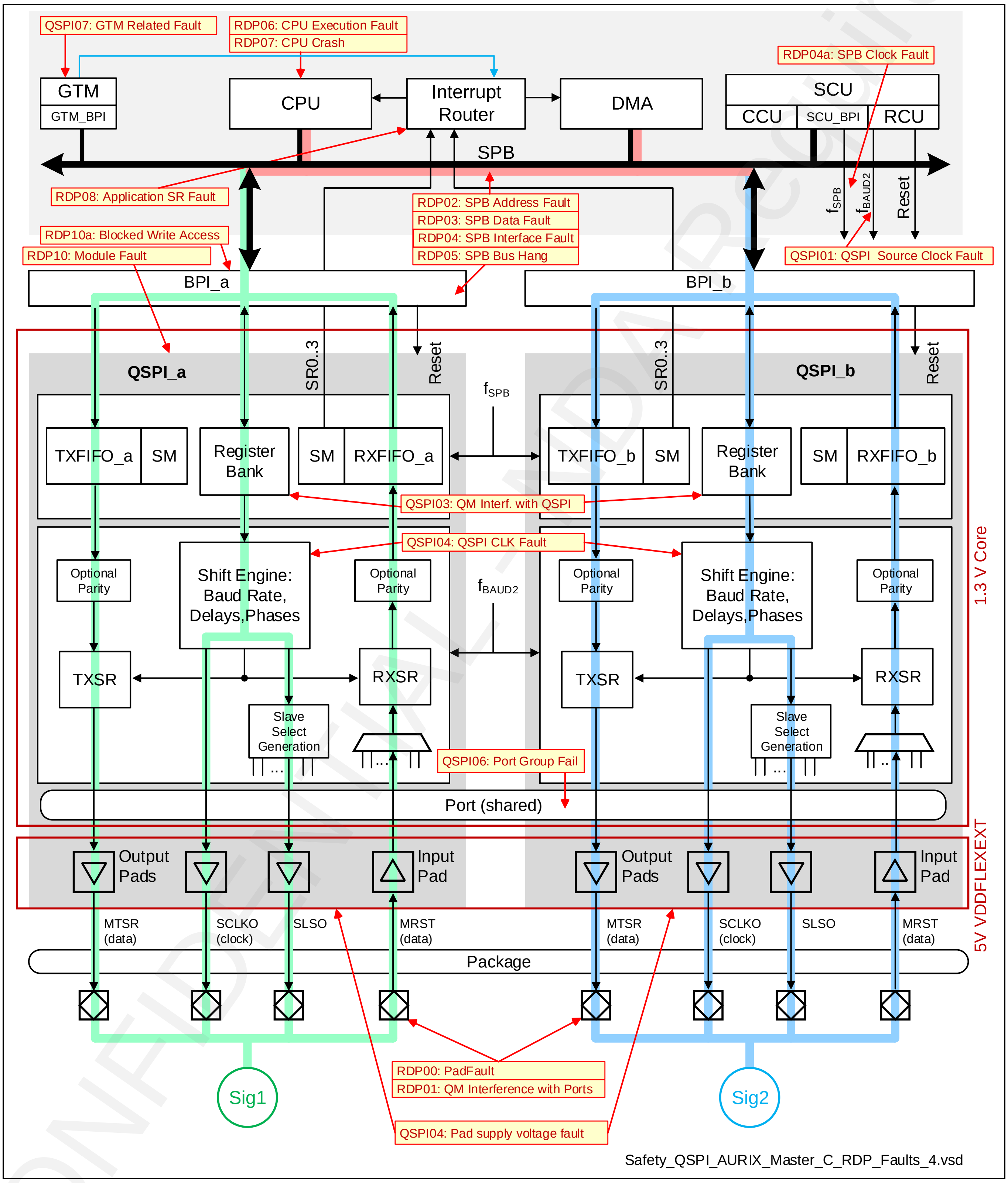

Figure 8 Basic Architecture & Potential Faults - Redundant Communication with two QSPI

Every device of the AURIX family implements several QSPI modules. QSPI0 is physically separated from the others.

4.23.2.1 Dependent Failure Initiators specific to Redundant QSPI/QSPI Scenario

There are potential dependent failure initiators which are specific to the application scenario using two QSPI modules, which may affect both S1 and S2 channels. These dependent failure initiators are listed in **Table 6**.

Table 6 Dependent Failure Initiators specific to QSPI/QSPI scenario

ID	Dependent Failure Initiator
QSPI01: QSPI source clock fault	QSPI source clock incorrect or missing for both S1 and S2 channel modules
QSPI02: QM Interference with QSPI	Interference from co-existent QM software changes QSPI configuration
QSPI03: QSPI CLK Fault	Fault in QSPI clock divider causes incorrect clock to digital blocks
QSPI04: Pad supply voltage fault	The supply voltage is at a wrong level leading to incorrect results
QSPI05: PCB connection between MRST and MTSR fails	The QSPI cannot transmit or receive
QSPI06: Port logic failure	Both slave select pins cannot be toggled by software.
QSPI07: GTM related failure	GTM does not generate the interrupts with the correct time period.

4.23.3 Hardware Safety Mechanisms

[SM_AURIX_QSPI_1]The following hardware safety mechanisms shall be used:

- **SM1[HW].SPB:CLKMON**[/req]

req featureID=SM_AURIX_QSPI_2 parentID=SM1[HW].QSPI:AP]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].QSPI:AP**[/req]

4.24 Other Serial Peripherals

Permanent and transient hardware faults in the serial peripherals, including ASCLIN, HSSL, PSI5, PSI5-S, SENT and Ethernet, may cause communication errors, including corrupted or delayed data, and other typical communication errors.

4.24.1 Monitoring Concept

There is no generic concept allocated to these peripherals. End to end safe protocol is an example of possible monitoring measure.

4.24.2 Hardware Safety Mechanisms

[SM_AURIX_SERIAL_COMM_1]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].ASCLIN:AP**
- **SM1[HW].HSSL:AP SM1[HW].PSI5:AP**
- **SM1[HW].PSI5-S:AP**

- **SM1[HW].SENT:AP**
- **SM1[HW].Ethernet:AP** [/req]

4.25 Analog Acquisition

The on-chip analog-digital converters in AURIX may be used for the acquisition of safety related analog signals. Permanent and transient hardware faults in the analog part as well as in the digital part may corrupt the signal. As the VADC and DSADC are AURIX application dependent parts, specific application scenarios to be assumed to explain specific deployment needs. For the purpose of illustration and explanation this section assumes two possible application scenarios consisting of

1. one specific single VADC analog input channel that is monitored by another specific single VADC analog input channel as described in section 4.25.1, or
2. one specific single DSADC analog input channel that is monitored by a specific single VADC analog input channel as described in section 4.25.2.

[SM_AURIX_ADC_3]This section focuses on the internal failures of the AURIX analog inputs; faults in external components shall be considered by the system integrator.[/req]

4.25.1 Redundant Acquisition Using two VADC Groups

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.
Figure 9 depicts the hardware of the chosen application scenario of a specific single VADC analog input channel that is monitored by another specific single VADC analog input channel from a different group[1] in absence of any additional application dependent safety measures..

The acquisition channel that is depicted in green in this hardware diagram processes a mission signal "S1". This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal "S2". This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

3. Comparison of S1 results against the S2 monitor results within the Fault Tolerant Time of the application
4. System level Plausibility checks on result value bounds and conversion rates

[SM_AURIX_ADC_4]The AURIX microcontrollers do not provide full intrinsic independence between acquisition channels. The registers and paths marked in red text in **Figure 9** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

[1] **[TC27x B-Step]**VADC module implementation is different in TC27x B-Step compared to the rest of the AURIX Family. In TC27x B-Step, several VADC groups are clustered around one converter, therefore it is recommended to select a channel from a different converter cluster as monitoring channel.

The potential dependent failure initiators for this scenario are divided into three classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 9**, and in section **7.3** where they are described
- Dependent Failure Initiators specific to the redundant VADC/VADC scenario identified as ADCxyz and VADCxyz in **Figure 9**, and in **Table 8** where they are described

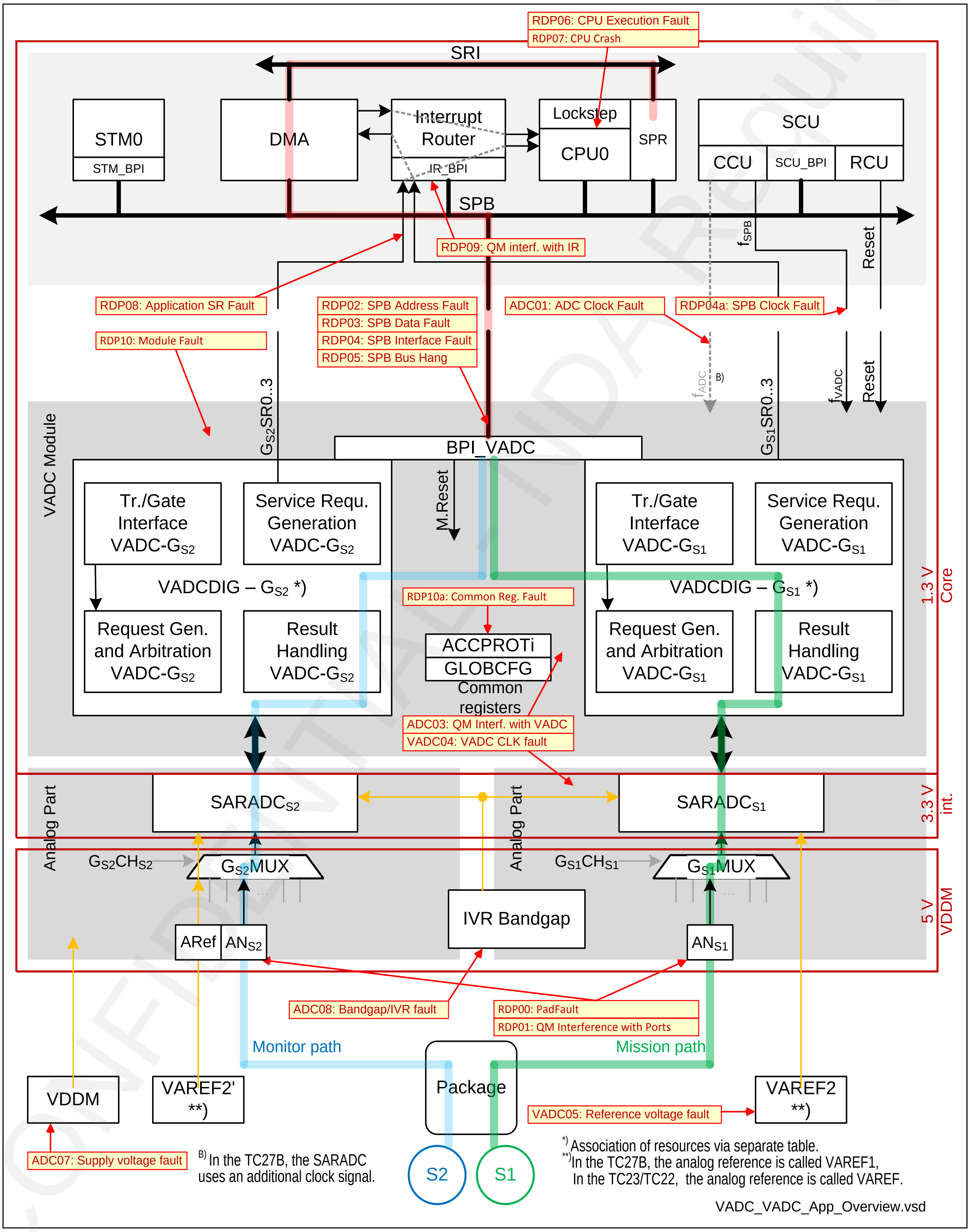
RDP06: CPU Execution Fault
RDP07: CPU Crash
SRI
STM0
STM_BPI
DMA
Interrupt Router
IR_BPI
Lockstep
CPU0
SPR
SCU
CCU
SCU_BPI
RCU
SPB
RDP09: QM interf. with IR
fSPB
Reset
RDP08: Application SR Fault
RDP10: Module Fault
RDP02: SPB Address Fault
RDP03: SPB Data Fault
RDP04: SPB Interface Fault
RDP05: SPB Bus Hang
ADC01: ADC Clock Fault
RDP04a: SPB Clock Fault
fADC B)
fVADC
Reset
GS2SR0..3
GS1SR0..3
VADC Module
BPI_VADC
M.Reset
Tr./Gate Interface VADC-GS2
Service Requ. Generation VADC-GS2
VADCDIG – GS2 *)
Request Gen. and Arbitration VADC-GS2
Result Handling VADC-GS2
Tr./Gate Interface VADC-GS1
Service Requ. Generation VADC-GS1
VADCDIG – GS1 *)
Request Gen. and Arbitration VADC-GS1
Result Handling VADC-GS1
RDP10a: Common Reg. Fault
ACCPROTi
GLOBCFG
Common registers
ADC03: QM Interf. with VADC
VADC04: VADC CLK fault
1.3 V Core
Analog Part
SARADCS2
SARADCS1
3.3 V int.
GS2CHS2
GS2MUX
GS1CHS1
GS1MUX
ARef
ANS2
IVR Bandgap
ANS1
5 V VDDM
ADC08: Bandgap/IVR fault
RDP00: PadFault
RDP01: QM Interference with Ports
Monitor path
Mission path
VDDM
VAREF2' **)
Package
VAREF2 **)
VADC05: Reference voltage fault
ADC07: Supply voltage fault
B) In the TC27B, the SARADC uses an additional clock signal.
S2
S1
*) Association of resources via separate table.
**) In the TC27B, the analog reference is called VAREF1, In the TC23/TC22, the analog reference is called VAREF.
VADC_VADC_App_Overview.vsd

Figure 9 Basic Architecture & Potential Faults - Redundant Acquisition Using Two VADC Groups

The recommended resources depend on the employed package. The table below lists the recommended resources for the various application scenarios

Table 7 Association of Resources for Application Example

Resource	BGA292/BGA516	BGA416
Mission Signal Group G_{S1}	G4 / G2[1] / G1[2]	G2
Mission Signal Channel CH_{S1}	CH1 / CH4[2]	CH4
Mission Signal Pin AN_{S1}	AN33[1] / AN16[2]	AN20
Monitor Signal Group G_{S2}	G0	G4
Monitor Signal Channel CH_{S2}	CH2	CH1
Monitor Signal Pin AN_{S2}	AN2	AN33
Alternate Reference Channel	G0CH0	G4CH0
Alternate Reference Pin ARef	AN0	AN32

1) In the TC26, the VADC groups have 16 inputs. In this case, AN33 belongs to group G2.
2) In the TC23, the corresponding ball for the mission signal is AN16 (G1CH4). When using the TC23x-ADAS device, do not use G2 and G3 together for mission signal and monitor signal.

4.25.1.1 Dependent Failure Initiators specific to Redundant VADC/VADC Scenario

There are potential dependent failure initiators which are specific to the application scenario using two VADC groups, which may affect both S1 and S2 channels. These dependent failure initiators are listed in **Table 8**.

Table 8 Dependent Failure Initiators specific to VADC/VADC scenario

ID	Dependent Failure Initiator
ADC01: ADC source clock fault	VADC source clock incorrect or missing for both S1 and S2 channel modules
ADC03: QM Interference with VADC	Interference from co-existent QM software changes VADC configuration
VADC04: VADC CLK Fault	Fault in VADC clock divider (DIVA/DIVD) causes incorrect clock to analog or digital blocks
VADC05: Reference voltage fault	The analog reference voltage is at a wrong level leading to incorrect results
ADC07: Supply voltage fault	The mixed-signal supply voltage is at a wrong level leading to incorrect results
ADC08: Bandgap/IVR fault	The bandgap that controls the IVR for the comparator generates a wrong voltage

4.25.2 Redundant Acquisition Using one VADC Channel and one DSADC Channel

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 10 depicts the hardware of the chosen application specific use case of a specific single DSADC analog input channel that is monitored by a specific single VADC analog input channel in absence of any additional application dependent safety measures.

The acquisition channel that is depicted in green in this hardware diagram processes a mission signal “S1”. This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal “S2”. This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

5. Comparison of S1 results against the S2 monitor results within the Fault Tolerant Time of the application
6. System level Plausibility checks on result value bounds and conversion rates

[SM_AURIX_ADC_5]The AURIX microcontrollers do not provide full intrinsic independence between acquisition channels. The registers and paths marked in red text in **Figure 10** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

The potential dependent failure initiators for this scenario are divided into three classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 10**, and in section **7.3** where they are described
- Dependent Failure Initiators specific to the redundant VADC/VADC scenario identified as ADCxyz **Figure 10**, and in **Table 10** where they are described

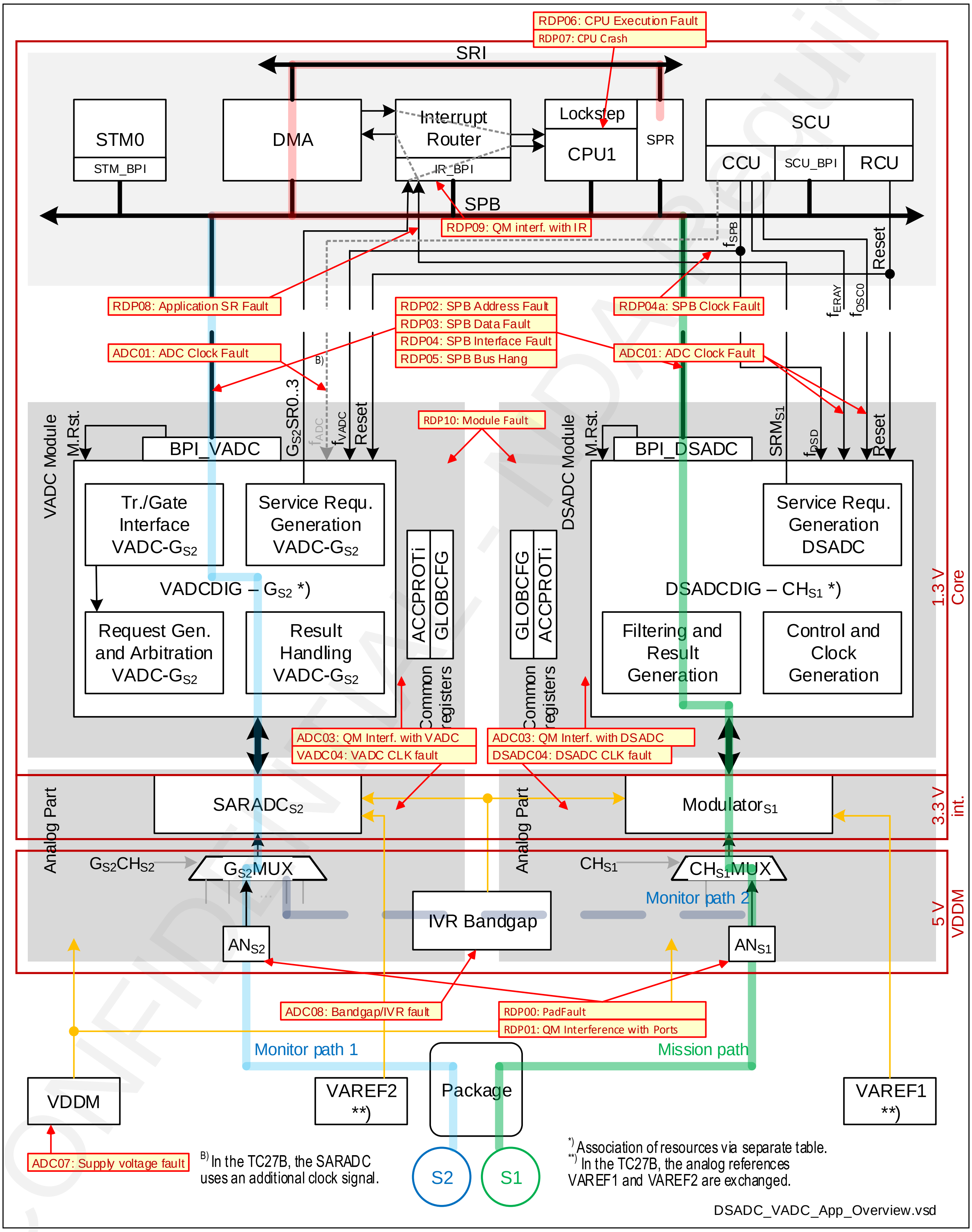

Figure 10 Basic Architecture & Potential Faults - Redundant Acquisition Using DSADC and VADC

The recommended resources depend on the employed package. The table below lists the recommended resources for the various application scenarios

Table 9 Association of Resources to Application Example

Resource	BGA292/BGA516	BGA416
Mission Signal Channel CH_{S1}	CH3C	CH2A
Mission Signal Pin AN_{S1}	AN44	AN20
Monitor Signal Group G_{S2}	G0	G4
Monitor Signal Channel CH_{S2}	CH2	CH1
Monitor Signal Pin AN_{S2}	AN2	AN33

4.25.2.1 Dependent Failure Initiators specific to Redundant VADC/VADC Scenario

There are potential dependent failure initiators which are specific to the application scenario using two VADC groups, which may affect both S1 and S2 channels. These dependent failure initiators are listed in **Table 10**.

Table 10 Dependent Failure Initiators specific to DSADC/VADC scenario

ID	Dependent Failure Initiator
ADC01: ADC source clock fault	VADC source clock incorrect or missing for both S1 and S2 channel modules
ADC03: QM Interference with VADC	Interference from co-existent QM software changes VADC configuration
ADC07: Supply voltage fault	The mixed-signal supply voltage is at a wrong level leading to incorrect results
ADC08: Bandgap/IVR fault	The bandgap that controls the IVR for the comparator generates a wrong voltage

4.25.3 Single Channel with Test Pattern

In this scenario, one safety related analog signal is converted through one VADC channel. In addition a test pattern, independent from the analog safety-related signal is available.

The test pattern technique is described in ISO26262-5 D.2.6.1 - see **[1]** - as *"a dataflow-independent cyclical test of input and output units. It uses a defined test pattern to compare observations with the corresponding expected values. Test coverage is dependent on the degree of independence between the test pattern information, the test pattern reception, and the test pattern evaluation. In a good design, the functional behavior of the system is not unacceptably influenced by the test pattern"*

Transient faults may be detected using oversampling.

4.25.4 Hardware Safety Mechanisms

[SM_AURIX_ADC_1]The following hardware safety mechanisms shall be used:

- **SM1[HW].VADC:CLKMON**[/req]

[SM_AURIX_ADC_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].VADC:AP**
- **SM1[HW].DSADC:AP**[/req]

Moreover, The VADC implements test features which may be used in a safety related application:

- The Converter Diagnostics connects an internally generated, defined, signal to the converter to test the proper operation of the converter.
- The Multiplexer Diagnostics connects a weak pull-up or pull-down device to an input channel to test the correct operation of the internal analog input multiplexer. A subsequent conversion can then confirm the expected modified signal level. Multiplexer diagnostics can be enabled for channels CH1 and CH2 of each group.
- The Broken-wire-detection preloads the converter network with a selectable level before sampling the input channel to test the proper connection of an external analog sensor to its input pin. The result will then reflect the preload value if the input signal is no longer connected. If buffer capacitors are used, a certain number of conversions may be required to reach the failure indication level. Broken wire detection can be enabled for each channel separately.

These test features are described with more details in
[3] Versatile Analog-to-Digital Converter (VADC) > Signal Path Test Modes, and
[3] Versatile Analog-to-Digital Converter (VADC) > Broken Wire Detection

4.26 Digital Inputs / Outputs

The digital inputs/outputs in AURIX include general purpose inputs/outputs (GPIO) as well as input capture and output generation by hardware timers (GTM, CCU6). GPIO, GTM, and CCU6 are application dependent parts for which there are no application independent safety mechanisms therefore specific application scenarios needs to be assumed to explain specific deployment needs. For the purpose of illustration and explanation this section assumes the following possible application scenarios consisting of

1. one specific single GTM Timer Input Module (TIM) channel that is monitored by application software, using a Capture/Compare Unit 6 (CCU6) module as redundant input module - **4.26.1**
2. one specific single GTM TIM channel that is monitored by application software, using another specific single GTM TIM channel as redundant input module - **4.26.2**
3. one specific single GTM Timer Output Module (TOM) channel that is monitored by application software based on output feedback acquired by a specific single GTM TIM channel - **4.26.3**
4. once specific single GTM TOM channel that is monitored by the AURIX Input / Output Monitor (IOM) comparing the output feedback with a reference signal generated by a CCU6 module - **4.26.4**
5. one specific single GTM TOM channel that is monitored by the AURIX Input / Output Monitor (IOM) comparing the output feedback with a reference signal generated by another specific GTM TOM channel - **4.26.5**
6. One specific single General Purpose Input (GPI) that is monitored by application software, using another specific GPI as redundant input - **4.26.6**
7. One specific single General Purpose Output that is monitored by application software based on output feedback acquired by another specific single General Purpose Input - **4.26.7**

[SM_AURIX_DIO_3]This section focuses on the internal failures of the AURIX digital input/outputs; faults in external components shall be considered by the system integrator.[/req]

Attention: *[SM_AURIX_GTM_1]* ***If the GTM SRAM is used, section 4.16 shall also be taken into account.*** *[/req]*

4.26.1 Redundant Input Capture with GTM TIM and CCU6

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 11**Figure 11** depicts the hardware of the chosen application scenario of a specific single TIM channel that is monitored by a specific CCU6 timer in absence of any additional application dependent safety measures..

The acquisition channel that is depicted in green in this hardware diagram processes a mission signal "S1". This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal "S2". This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

1. Comparison of S1 results against the S2 monitor results within the Fault Tolerant Time of the application
2. System level plausibility checks on result value bounds and rates of change of period, duty, and edge count

[SM_AURIX_GTM_DFI_1]The AURIX microcontrollers do not provide full intrinsic independence between acquisition channels. The registers and paths marked in red text in **Figure 11** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator. [/req]

The potential dependent failure initiators for this scenario are divided into two classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 11**, and in section **7.3** where they are described

Note: Due to the hardware diversity between GTM TIM and CCU6 timer, there is no dependent failure initiator specific to this scenario.

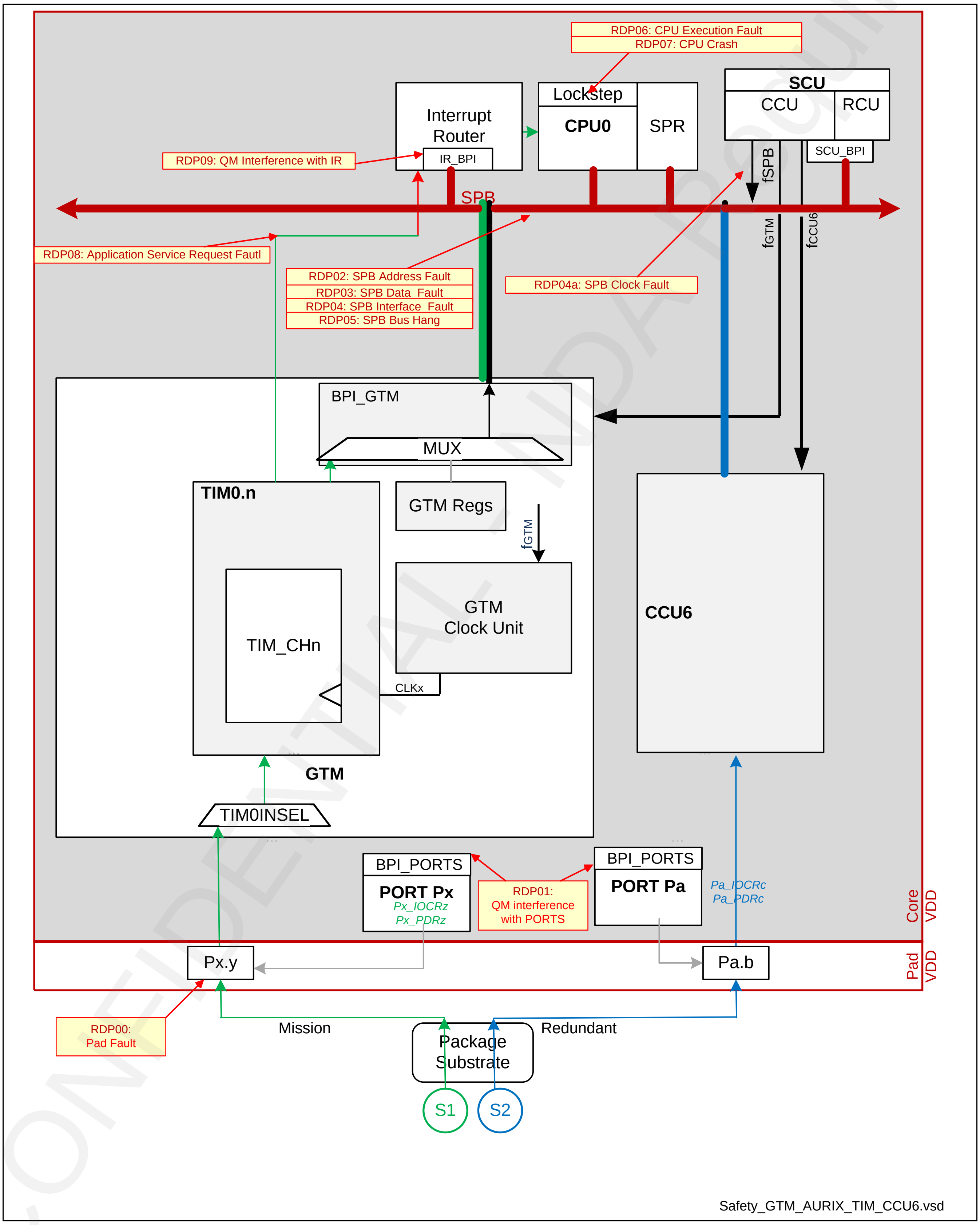

Figure 11 Basic Architecture & Potential Faults - Redundant Input Capture with GTM TIM and CCU6

4.26.2 Redundant Input Capture with two GTM TIMs

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 11Figure 12 depicts the hardware of the chosen application specific use case of a single that is monitored by another specific single TIM channel in absence of any additional application dependent safety measures..

The input channel that is depicted in green in this hardware diagram processes a mission signal "S1". This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal "S2". This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

1. Comparison of S1 results (period, duty, edge count) against the S2 monitor results within the Fault Tolerant Time of the application
2. System level plausibility checks on bounds and rates of change of period, duty and edge count

[SM_AURIX_GTM_DFI_2]The AURIX microcontrollers do not provide full intrinsic independence between these channels. The registers and paths marked in red text in Figure 12 are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

The potential dependent failure initiators for this scenario are divided into three classes:

- Dependent Failure Initiators applicable to basic device operation, described in section 7.2
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in Figure 12, and in section 7.3 where they are described with more details.
- Dependent Failure Initiators specific to the GTM redundant TIM scenario (GTMxyz) identified in Figure 12, and in Table 11 where they are described

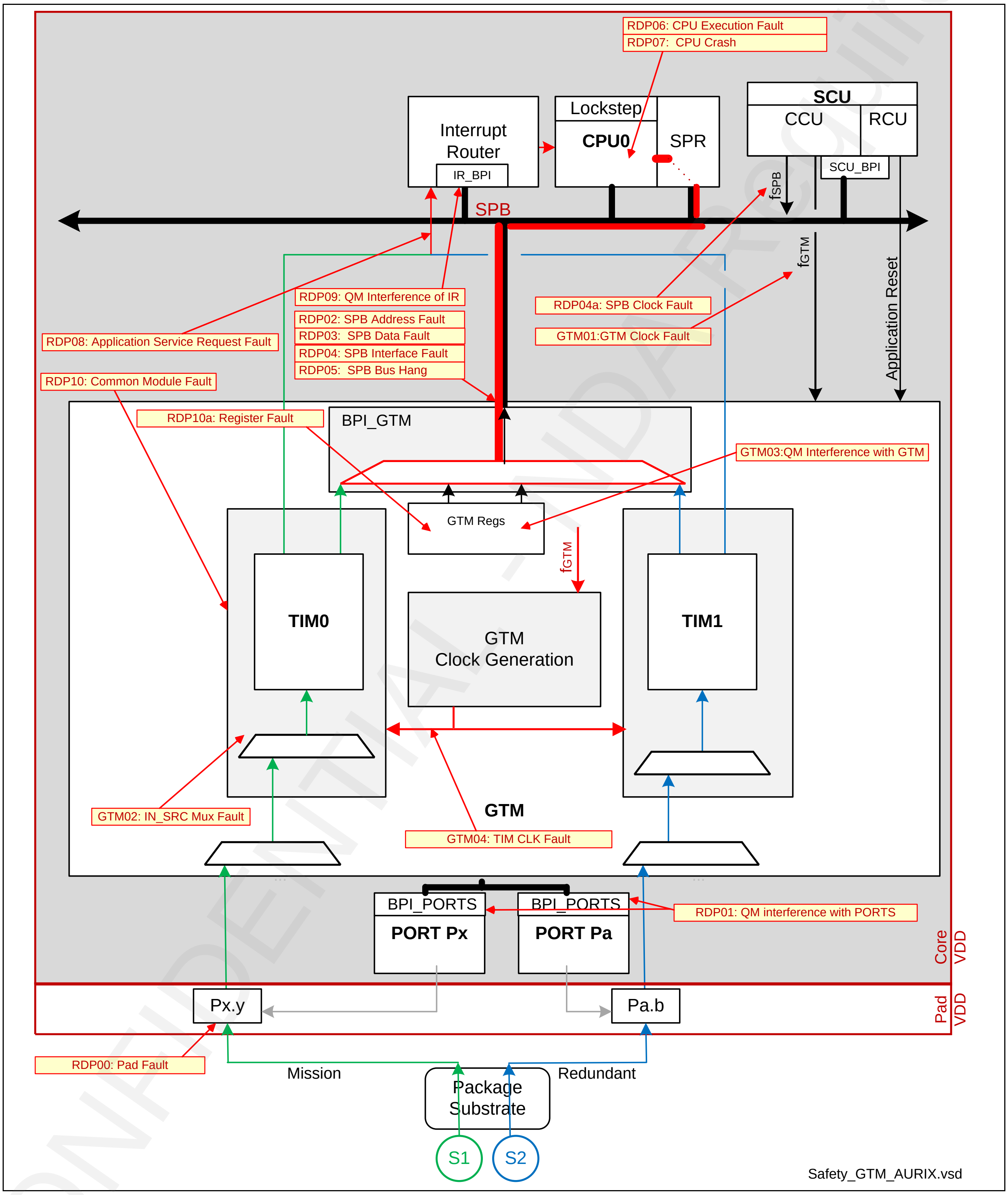

Figure 12 Basic Architecture & Potential Faults - Redundant Input Capture with two GTM TIMs

Note: RDP10 is especially significant in the GTM redundant TIM channel scenario because there is insufficient spatial separation of the TIM channels to ensure that potential faults are guaranteed to be completely independent.

4.26.2.1 Dependent Failure Initiators specific to GTM Redundant TIM Scenario

Table 11 Dependent Failure Initiators specific to GTM Redundant TIM scenario

ID	Dependent Failure Initiator
GTM01: GTM source clock fault	GTM source clock incorrect or missing for both S1 and S2 channel modules
GTM02: GTM IN_SRC mux fault	IN_SRC mux select fault selects adjacent S1 signal as input for S2 TIM
GTM03: QM Interference with GTM	Interference from co-existent QM software changes GTM TIM configuration
GTM04: TIM CLK Fault	Fault in GTM GCLK divider or CLKx divider causes incorrect clock to TIM channel S1 and S2

For a given application, additional application dependent measures would be required to address these common causes in a sufficient way to achieve the application safety goals.

4.26.3 GTM Output Monitoring by Software

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 11**Figure 13** depicts the hardware of the chosen application specific use case of a specific single GTM TOM channel that is monitored by application software over another specific single GTM TIM in absence of any additional application dependent safety measures..

The actuation channel that is depicted in green in this hardware diagram produces a mission signal “S1”. This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal “S2” feedback by an external system element E1. This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

1. Comparison of S1 results (period, duty, edge count) against the S2 monitor results within the Fault Tolerant Time Interval of the application
2. System level plausibility checks on bounds and rates of change of period, duty and edge count

[SM_AURIX_GTM_DFI_3]The AURIX microcontrollers do not provide full intrinsic independence between these channels. The registers and paths marked in red text in **Figure 13** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

The potential dependent failure initiators for this scenario are divided into three classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 13**, and in section **7.3** where they are described

- Dependent Failure Initiators specific to the GTM output monitoring by software scenario (GTMxyz) identified in **Figure 13**, and in **Table 12** where they are described

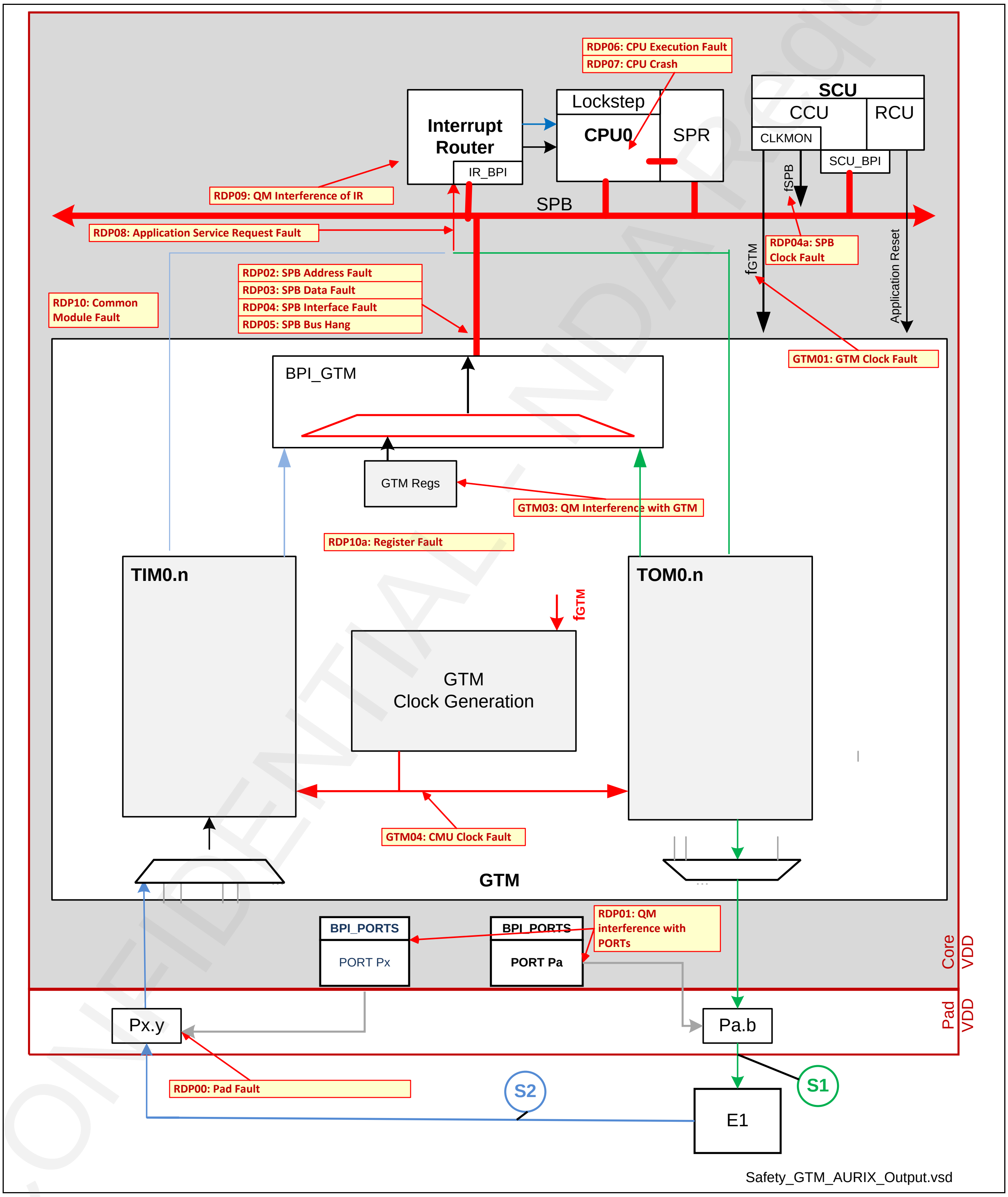

Figure 13 Basic Architecture & Potential Faults - GTM Output Monitoring by Software

4.26.3.1 Dependent Failure Initiators specific to GTM Output Monitoring by Software

Table 12 Dependent Failure Initiators specific to GTM Output Monitoring by Software

ID	Dependent Failure Initiator
GTM01: GTM source clock fault	GTM source clock incorrect or missing for both S1(TOM) and S2(TIM) channel modules
GTM03: QM Interference with GTM	Interference from co-existent QM software changes GTM TOM/TIM configuration
GTM04: TIM CLK Fault	Fault in GTM GCLK divider or CLKx divider causes incorrect clock to TOM/TIM channel S1 and S2

For a given application, additional application dependent measures would be required to address these common causes in a sufficient way to achieve the application safety goals.

4.26.4 GTM Output Monitoring with IOM and CCU6

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 11**Figure 14** depicts the hardware of the chosen application specific scenario of a specific single GTM TOM channel generating a “mission” output signal that is feedback by system element E1 to the IOM module where it is compared against a reference signal generated by the independent and diverse CCU6 module in absence of any additional application dependent safety measures..

The actuation channel that is depicted in green in this hardware diagram produces a mission signal “S1”. This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal “S2” feedback by an external system element E1. This channel is henceforth referred to as the S2 channel. The reference signal “S3” is generated by the CCU6. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by the IOM module within the Fault Tolerant Time Interval of the application.

[SM_AURIX_GTM_DFI_4]The AURIX microcontrollers do not provide full intrinsic independence between acquisition channels. The registers and paths marked in red text in **Figure 14** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator. [/req]

The potential dependent failure initiators for this scenario are divided into two classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 14**, and in section **7.3** where they are described
- Dependent Failure Initiators specific to the GTM TOM monitoring with IOM and CCU6 scenario (GTMxyz) identified in **Figure 14**, and in **Table 13** where they are described

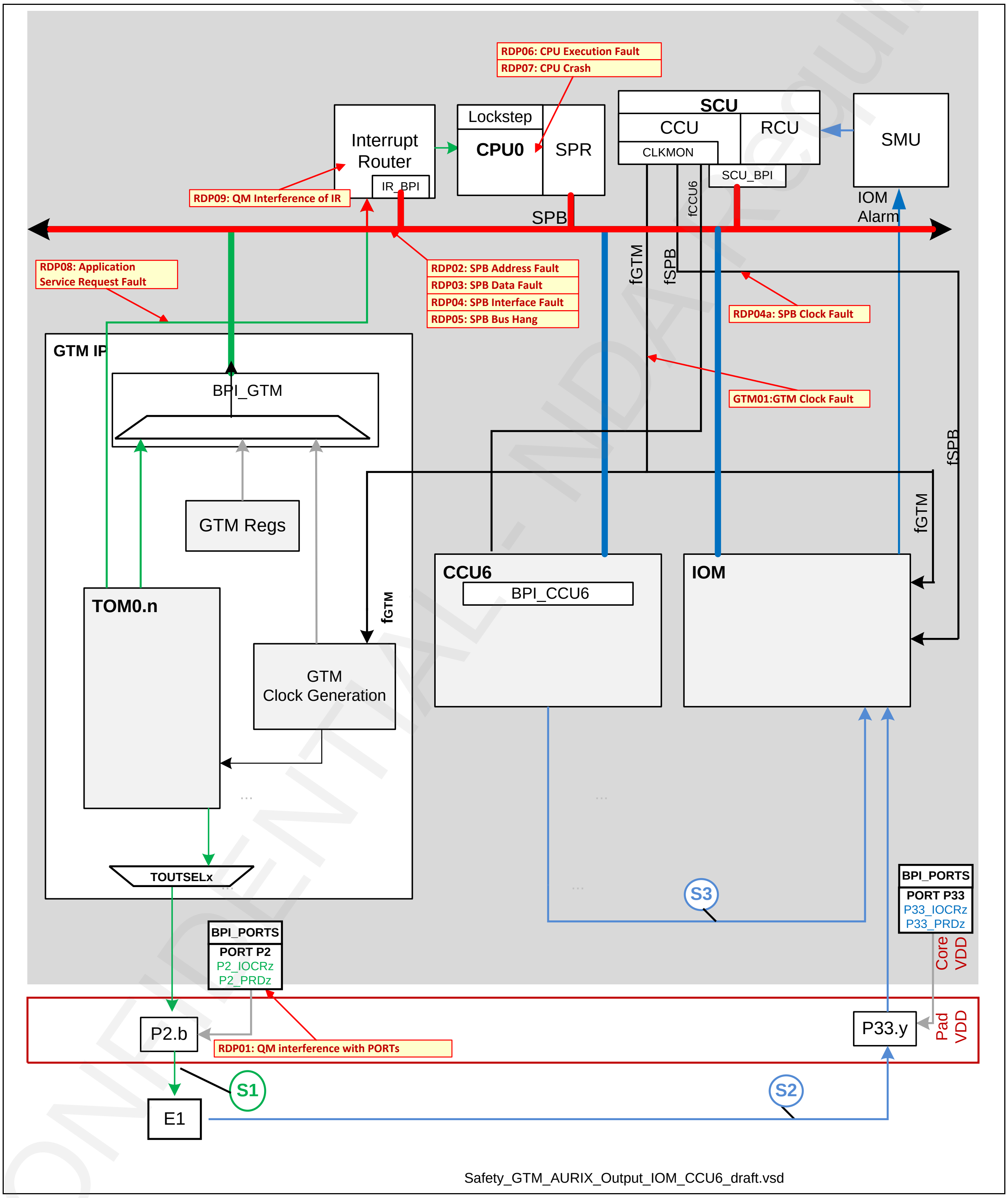

Figure 14 Basic architecture & potential faults - GTM Output Monitoring with IOM and CCU6

Note: If necessary, it is possible to synchronize the PWM generation in GTM and CCU6, using the signals listed in [3] Generic Timer Module (GTM) > GTM Implementation > CCU6x Connections.

Further information for usage of the IOM including the description of the different IOM usage modes, IOM inputs mapping to ports, and potentially latent faults considerations is available in section 4.26.9

4.26.4.1 Dependent Failure Initiators specific to GTM Output Monitoring with IOM and CCU6

Table 13 Dependent Failure Initiators specific to GTM Output Monitoring with IOM and CCU6

ID	Dependent Failure Initiator
GTM01: GTM source clock fault	GTM source clock incorrect or missing causing both S1 generation with GTM TOM and error detection by IOM to fail

For a given application, additional application dependent measures would be required to address these common causes in a sufficient way to achieve the application safety goals.

4.26.5 GTM Output Monitoring with IOM and TOM

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 11Figure 15 depicts the hardware of the chosen application specific scenario of a specific single GTM TOM channel generating a "mission" output signal that is feedback by system element E1 to the IOM module where it is compared against a reference signal generated by another specific single GTM TOM channel in absence of any additional application dependent safety measures..

The actuation channel that is depicted in green in this hardware diagram produces a mission signal "S1". This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal "S2" feedback by an external system element E1. This channel is henceforth referred to as the S2 channel. The reference signal "S3" is generated by a GTM TOM channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by the IOM module within the Fault Tolerant Time Interval of the application.

[SM_AURIX_GTM_DFI_5]The AURIX microcontrollers do not provide full intrinsic independence between acquisition channels. The registers and paths marked in red text in Figure 15 are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator. [/req]

The potential dependent failure initiators for this scenario are divided into two classes:

- Dependent Failure Initiators applicable to basic device operation, described in section 7.2
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in Figure 15, and in section 7.3 where they are described
- Dependent Failure Initiators specific to the GTM TOM monitoring with IOM and GTM TOM scenario (GTMxyz) identified in Figure 15, and in Table 14 where they are described
-

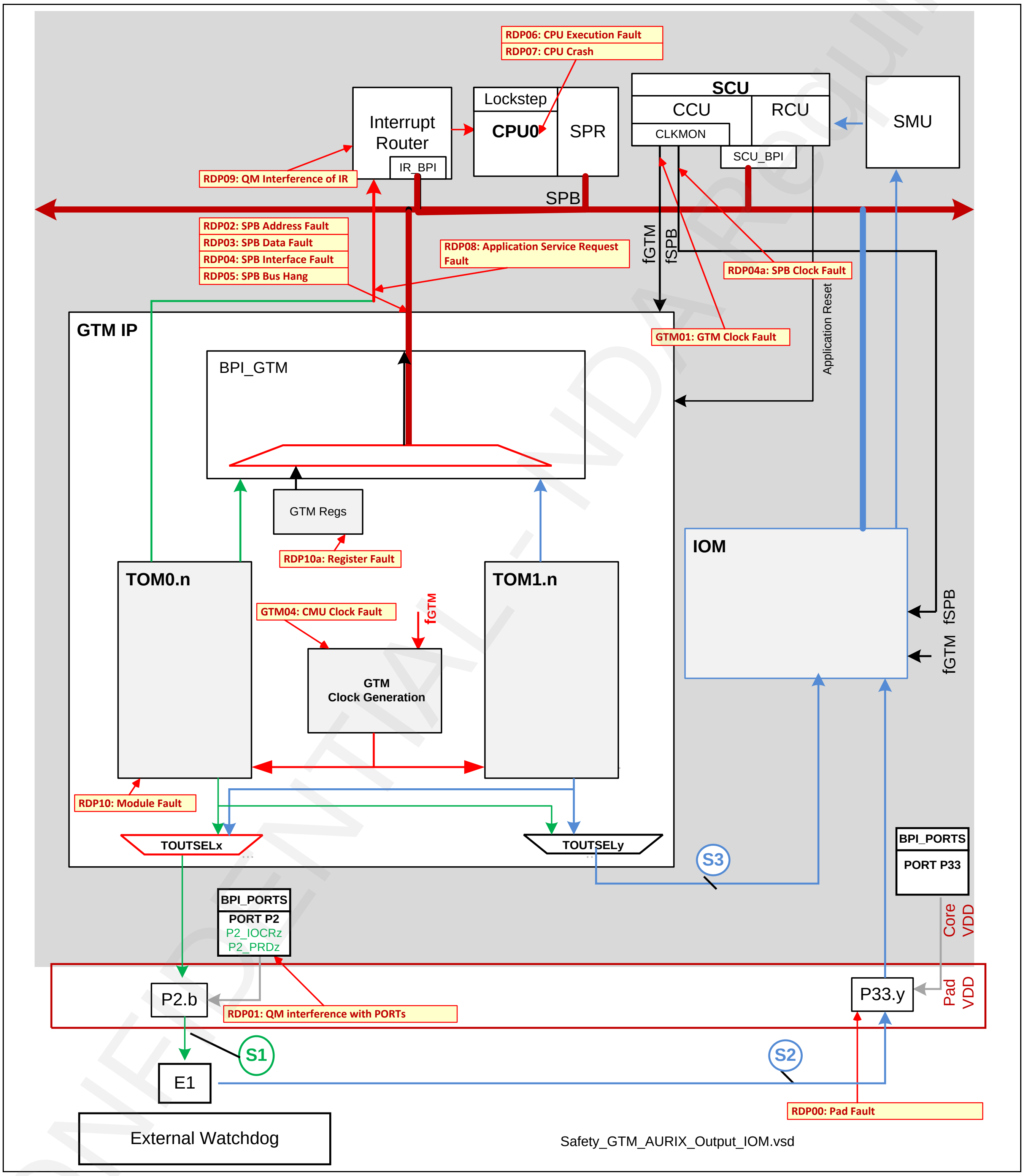

Figure 15 Basic architecture & potential faults - GTM Output Monitoring with IOM and TOM

Further information for usage of the IOM including the description of the different IOM usage modes, IOM inputs mapping to ports, and potentially latent faults considerations is available in section 4.26.9

Note: ATOM channels can be used as TOM channels, directly controlled by the CPU. Other GTM parts which can interact with ATOM channels, such as ARU and MCS are not considered in this usage scenario.

4.26.5.1 Dependent Failure Initiators specific to GTM Output Monitoring with IOM and TOM

Table 14 Dependent Failure Initiators specific to GTM Output Monitoring with IOM and TOM

ID	Dependent Failure Initiator
GTM01: GTM source clock fault	GTM source clock incorrect or missing for both all TOM channels
GTM03: QM Interference with GTM	Interference from co-existent QM software changes GTM TOM/IOM configuration
GTM04: Internal CLK Fault	Fault in GTM GCLK divider or CLKx divider causes incorrect clock to all TOM channels
GTM05: TOUTSEL Fault	Fault in GTM TOUTSEL multiplexer causes incorrect signal to be forwarded to the Port

For a given application, additional application dependent measures would be required to address these common causes in a sufficient way to achieve the application safety goals.

4.26.6 Redundant General Purpose Inputs

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

[Figure 11]**Figure 16** depicts the hardware of the chosen application specific use case of a specific single general purpose input "Px.y" that is redundantly captured by another specific single general purpose input "Pa.b" in absence of any additional application dependent safety measures..

The input channel that is depicted in green in this hardware diagram processes a mission signal "S1". This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The redundant input channel that is depicted in blue in this hardware diagram processes a related or identical signal "S2". This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

1. Comparison of S1 results against the S2 monitor results within the Fault Tolerant Time of the application
2. System level plausibility checks

[SM_AURIX_GPI_DFI_1]The AURIX microcontrollers do not provide full intrinsic independence between these channels. The registers and paths marked in red text in **Figure 16** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

The potential dependent failure initiators for this scenario are divided into two classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 16**, and in section **7.3** where they are described

Note: Because Ports are used in all application dependent parts application scenarios, there are no dependent failure initiators specific to this scenario.

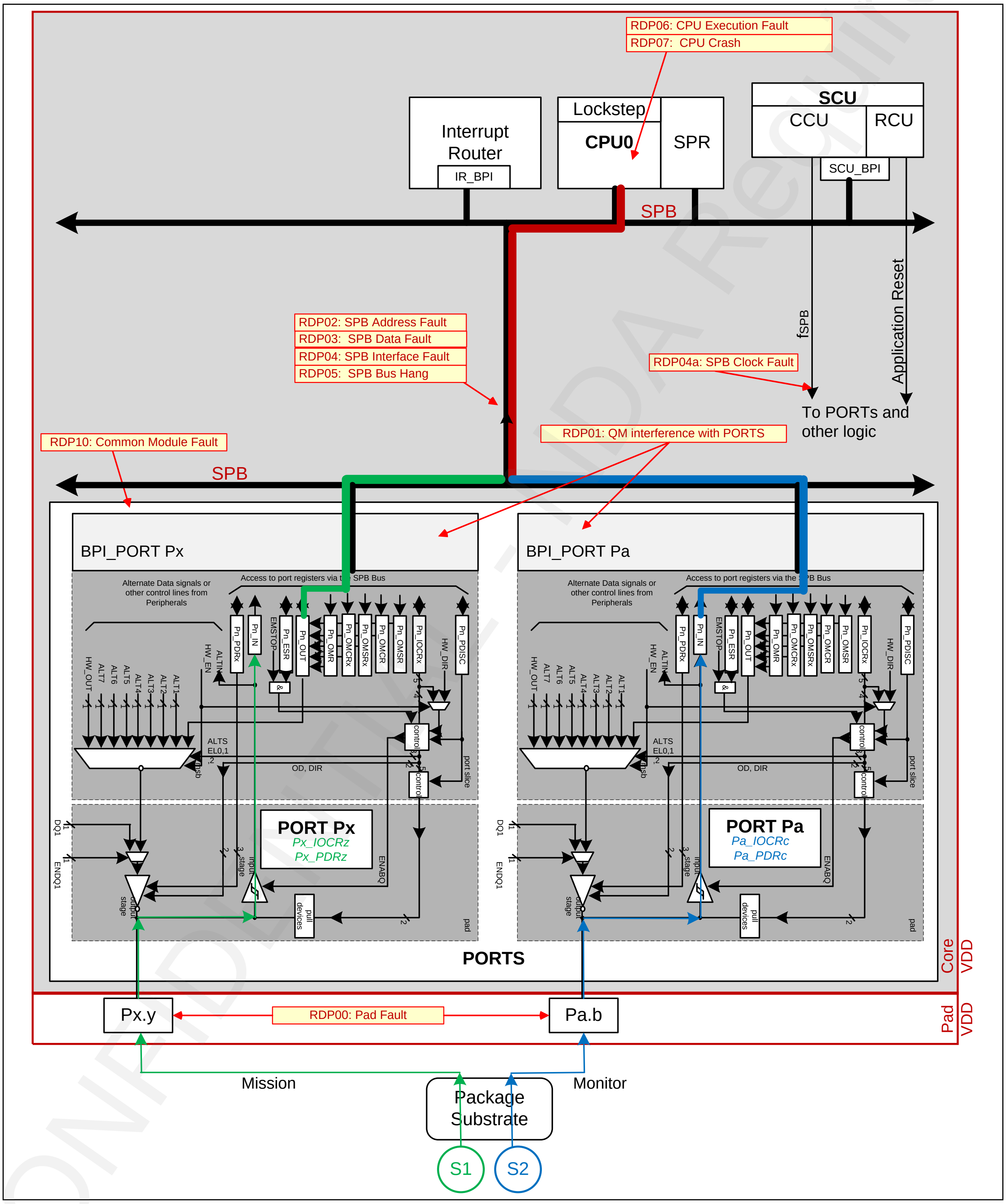

Figure 16 Basic Architecture & Potential Faults - Redundant General Purpose Inputs

4.26.7 General Purpose Output Read-Back

The purpose of this section is to explain the underlying considerations that need to be taken into account by the system integrator defining application dependent safety measures.

Figure 11**Figure 17** depicts the hardware of the chosen application specific use case of a specific single general purpose output "Px.y" that is monitored by application software over another specific single general purpose input "Pa.b" in absence of any additional application dependent safety measures..

The actuation channel that is depicted in green in this hardware diagram produces a mission signal "S1". This channel is henceforth referred to as the S1 channel. The configuration and control registers required for this part of the processing are shown in green text.

The monitoring channel that is depicted in blue in this hardware diagram processes a related or identical signal "S2". This channel is henceforth referred to as the S2 channel. The configuration control registers required for this part of the processing are shown in blue text.

The basic assumption is that any faults in the S1 channel can be detected by a combination of simple measures:

1. Comparison of S1 results against the S2 monitor results within the Fault Tolerant Time of the application
2. System level plausibility checks

[SM_AURIX_GPO_DFI_1]The AURIX microcontrollers do not provide full intrinsic independence between these channels. The registers and paths marked in red text in **Figure 17** are global or shared data-paths which could be used by both channels. These remaining potential causes of dependent failures between the two channels, also called dependent failure initiators in this document, shall be considered by the system integrator.[/req]

The potential dependent failure initiators for this scenario are divided into two classes:

- Dependent Failure Initiators applicable to basic device operation, described in section **7.2**
- Dependent Failure Initiators applicable to redundant peripheral scenarios identified as RDPxyz in **Figure 17**, and in section **7.3** where they are described

Note: Because Ports are used in all application dependent parts application scenarios, there are no dependent failure initiators specific to this scenario.

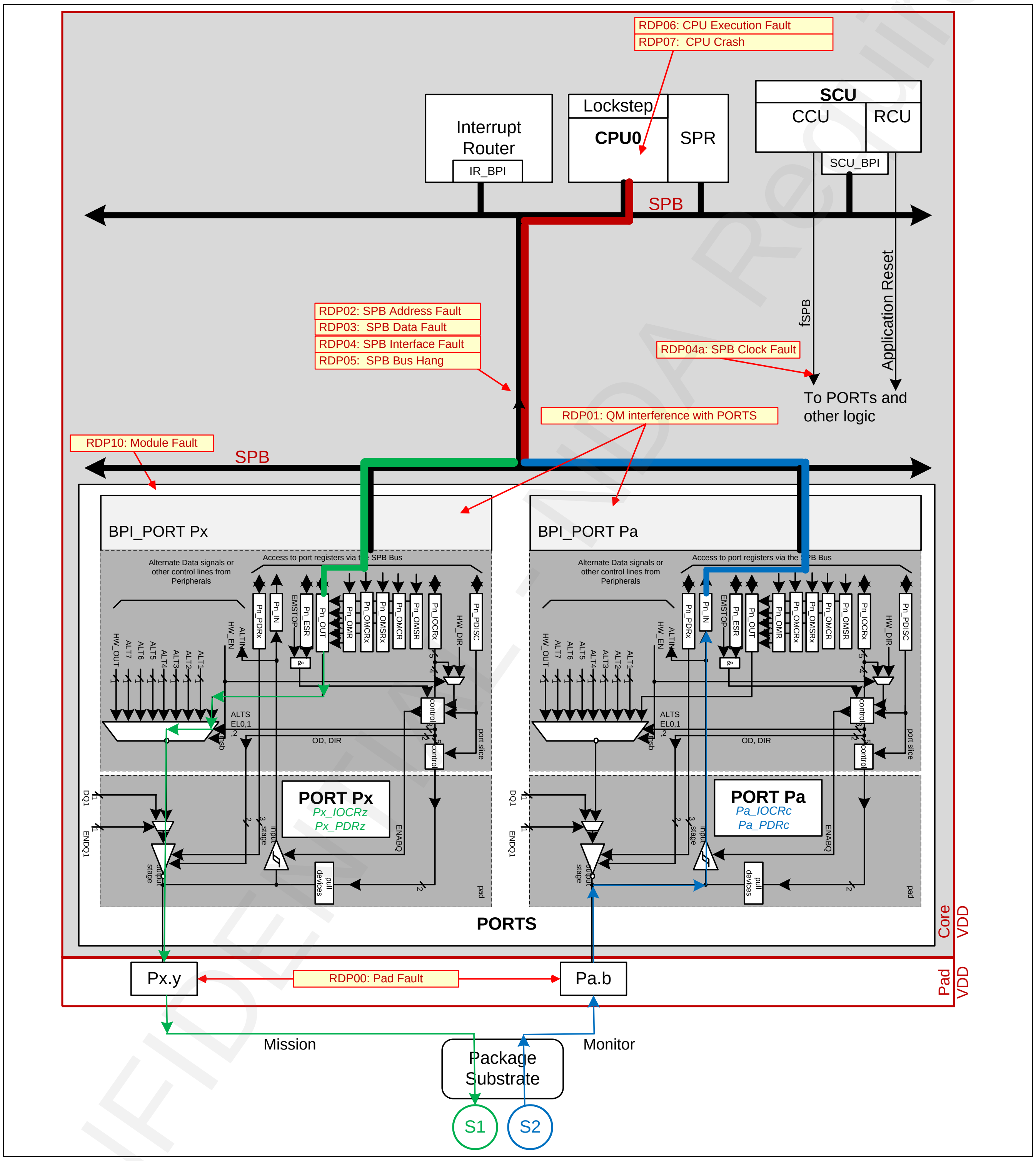

Figure 17 Basic Architecture & Potential Faults - General Purpose Output Read-back

Note: .GPIO:InternalLoopback]*It is also possible to directly read-back the level of a pin configured as digital output per software by reading the corresponding port input register. This is called GPIO internal loopback. Generally, from a system perspective, the external feedback concept makes more sense as it also covers faults in other components of the system.*

4.26.8 Hardware Safety Mechanisms

[SM_AURIX_DIO_1]The following hardware safety mechanisms shall be used:

- **SM1[HW].GTM:CLKMON**
- **SM1[HW].SPB:CLKMON**

[/req]

[SM_AURIX_DIO_2]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].GTM:AP**
- **SM1[HW].CCU6:AP**
- **SM1[HW].PORT:AP**[/req]

4.26.9 Input Output Monitor (IOM)

The Input Output Monitor (IOM) is a smart I/O comparison module which can be used to monitor digital signals. It provides 16 channels, where each channel consists of the signal to be monitored and a reference signal. The IOM is able to monitor the following signals:

- .GTM:IOM]GTM timer outputs (TOM/ATOM)
- CCU6 timer outputs
- GPIO inputs

The section *Input Output Monitor (IOM)* in **[3]** provides a list of GTM and CCU6 timer outputs as well as GPIO inputs of which each can be used as either a monitor or a reference signal.

The each IOM channel can be configured to either:

- Compare the monitor signal against the reference signal, with a configurable acceptance window with respect to rising and falling edges
- Monitor the period and/or duty cycle of a PWM signal
- Monitor the setup and/or hold time of a digital signal

In case the IOM detects a failure in a monitored signal, a local event is generated. The IOM Event Combiner Module (ECM) can combine the 16 local events to generate the IOM SMU alarm from a single local event or a combination of multiple local events.

For the GPIO inputs to the IOM, It is also possible to connect the pin to a GTM TIM input listed in Table 15 and Table 16.

Attention: ***All the pins may not be available in all package variants, please refer to the respective device datasheet [4].***

Table 15 IOM mapping to GPIO port and GTM input (TC29x/TC27x/TC26x)

IOM port naming	GPIO Port	GTM input
Iom_pin_i(0)	P33.0	TIN22
Iom_pin_i(1)	P33.1	TIN23
Iom_pin_i(2)	P33.2	TIN24

IOM port naming	GPIO Port	GTM input
Iom_pin_i(3)	P33.3	TIN25
Iom_pin_i(4)	P33.4	TIN26
Iom_pin_i(5)	P33.5	TIN27
Iom_pin_i(6)	P33.6	TIN28
Iom_pin_i(7)	P33.7	TIN29
Iom_pin_i(8)	P33.8	TIN30
Iom_pin_i(9)	P33.9	TIN31
Iom_pin_i(10)	P33.10	TIN32
Iom_pin_i(11)	P33.11	TIN33
Iom_pin_i(12)	P33.12	TIN34
Iom_pin_i(13)	P20.12	TIN68
Iom_pin_i(14)	P20.13	TIN69
Iom_pin_i(15)	P20.14	TIN70

Table 16 IOM mapping to GPIO port and GTM input (TC23x/TC22x/TC21x)

IOM port naming	GPIO Port	GTM input
Iom_pin_i(0)	P33.0	TIN22
Iom_pin_i(1)	P33.1	TIN23
Iom_pin_i(2)	P33.2	TIN24
Iom_pin_i(3)	P33.3	TIN25
Iom_pin_i(4)	P33.4	TIN26
Iom_pin_i(5)	P33.5	TIN27
Iom_pin_i(6)	P33.6	TIN28
Iom_pin_i(7)	P33.7	TIN29
Iom_pin_i(8)	P33.8	TIN30
Iom_pin_i(9)	P33.9	TIN31
Iom_pin_i(10)	P33.10	TIN32
Iom_pin_i(11)	P33.11	TIN33
Iom_pin_i(12)	P33.12	TIN34
Iom_pin_i(13)	P20.12	-
Iom_pin_i(14)	P20.13	-
Iom_pin_i(15)	P20.14	-

4.26.9.1 IOM access protection

[SM_AURIX_IOM_1]To avoid the corruption of the IOM configuration, the access to the IOM configuration registers is protected. The following safety mechanisms shall be used to ensure freedom from interference:

- **SM1[HW].IOM:AP**[/req]

4.26.9.2 Testing of the IOM

[SM_AURIX_IOM_2]When used to monitor safety related I/Os, The IOM contributes to the latent fault metric. The following tests shall be executed once at startup:

- **SM2[AoU].IOM:TEST**
- **SM2[AoU].IOM:AP**[/req]

4.27 CRC Engines

The AURIX microcontroller provides several hardware CRC engines for efficient CRC calculation.
The following CRC engine may be used in the context of system level safety mechanisms:

- **SM1[HW].FCE:CRC32**
- **SM1[HW].FCE:CRC16**
- **SM1[HW].FCE:CRC8**
- **SM1[HW].CPU:CRC32**
- **SM1[HW].DMA.DATA:CRC32**
- **SM1[HW].DMA.ADDR:CRC32**[/req]

Additionally, the FCE implements the generic access enable protection safety mechanism:

- **SM1[HW].FCE:AP**

4.28 Parts Supporting Advanced Driver Assistance Systems (ADAS)

Some of the parts and sub-parts in AURIX are especially targeting the ADAS application domain. This is:

1. The ADAS Extended Memory SRAM (EMEM)
2. The Fast Fourier Transform (FFT) accelerator
3. The Camera Interface (CIF)
4. Inter-Integrated Circuit Module (I2C)

Note: EMEM, FFT, and CIF modules are only present in specific ADAS devices.

4.28.1 Monitoring Concept

There is no generic monitoring concept allocated to these peripherals. They inherit generic safety requirements related to the microcontroller architecture. There is no safety requirement dedicated to the acquisition of information for ADAS functions according to the product targets.

4.28.2 Hardware Safety Mechanisms

Following safety mechanisms, implemented in hardware, may be used for the EMEM:

- **SM1[HW].SRAM:ECC**
- **SM1[HW].SRAM:EDC**

[SM_AURIX_ADAS_1]The following safety mechanisms shall be used to ensure freedom from interference when different applications and software elements with different ASIL requirements are executing on the same microcontroller:

- **SM1[HW].LMU:AP**
- **SM1[HW].I2C:AP** [/req]

4.28.3 Limitations specific to EMEM and CIF

[SM_AURIX_ADAS_2]Several generic hardware safety mechanisms are not available for EMEM and CIF. It is assumed that the following failure modes are controlled by safety mechanisms in software, under the responsibility of the integrator:

- Random hardware faults in other hardware sub-parts that cause a cascading failure corrupting the content of the CIF registers. Rationale: no hardware support is provided to control interferences to the CIF registers
- Random hardware faults in other hardware sub-parts that cause a cascading failure corrupting the content of the EMEM. Rationale: no hardware support is provided to control interferences to the EMEM
- Addressing faults inside the EMEM that are not controlled by ECC. Rationale: the SRAM address monitor safety mechanism **SM1[HW].SRAM:ADDRMON** is not implemented in the EMEM
- Failure modes of the EMEM ECC logic. Rationale: unlike the LMU SRAM, no hardware support is provided to control the failure modes of the EMEM ECC logic [/req]

4.29 HSM and OCDS

This is a section about the Hardware Security Module (HSM) and the On-Chip Debug Support (OCDS).

4.29.1 Monitoring Concept

There is no generic monitoring concept allocated to these peripherals as they are not intended to be used in safety applications. They inherit generic safety requirements related to the microcontroller architecture.

The OCDS Debug and test modes have the potential to transparently change the behavior of a module, that is why those modes can only be activate by write sequences, this is several write accesses, to different registers, are necessary to active debug and test modes.

Additionally the OCDS and HSM have they own Master Identifier on the on-chip bus system, therefore any unintended access from these modules to other modules can be controlled by the safety mechanisms supporting the coexistence of element, see section 4.2.

5 Safety Mechanisms

The section provides information on the safety mechanisms for each functional part of the AURIX microcontroller architecture and state the assumptions related to the use of each safety mechanism. The technical details of each safety mechanism can be found in the User's Manual [3] for the microcontroller used.

Each safety mechanism is specified here in the following format:

Name of the Safety Mechanism

ID	One or more identifiers. The same identifiers are used in the FMEDA.
Description	A description
References	References to further technical details in the user documentation
Error Signaling	Error signaling types are: • SMU alarm • Interrupt service request • CPU trap • Application software error handler *Note: Some errors result in more than one type of error signaling.* *Note: Consult the User's Manual [3] to check that a specific SMU alarm is available for the specific device. SMU alarms that are not available are described as 'Reserved' in the User's Manual [3].*
Error Detection time / Test Execution Time	Gives an indication of the time needed by the safety mechanism to detect an error. For a hardware safety mechanism, an error is considered detected when the corresponding SMU alarm is raised. The time necessary to propagate the alarm signal to the SMU, is not taken into account. ***Attention: The values given in this field for hardware safety mechanisms are theoretical worst cases based on the analysis of the safety mechanism specification.*** For safety mechanisms to be implemented by the system integrator at software or system level, this is implementation dependent.
Product Configuration	Lists the possible hardware configuration including the differences between the several microcontrollers of the AURIX family regarding the safety mechanism.
Initialization	Describes what must be done by the system integrator to initialize the safety mechanism. *Note: This describes the initialization of the safety mechanism itself, for*

	example if there are thresholds to be configured. *[SM_AURIX_ALARM_INIT_1]**What is not described but shall be done by the application is to initialize the corresponding SMU alarm and/or interrupt signals used for error signaling.**[/req]*
Tests	One or more tests necessary to detect the unavailability of a safety mechanism. These can be online monitors implemented in hardware, software tests provided in SafeTlib or system level checks to be implemented by the system integrator. The tests listed here support the latent fault metric, it is recommended to execute them at startup, but according to **[1]** they can be executed anytime within the multiple-point fault detection interval.
Limitations and/or Recommendations	Any hint which does not belong to one of the other fields

In the safety mechanisms description, N/A means that the field is Not Applicable for this specific safety mechanism.

5.1 Hardware Safety Mechanisms

5.1.1 Lockstep CPU

ID	SM1[HW].CPU:LOCKSTEP
Description	A lockstep CPU detects permanent and transient failures of the CPU by the way of hardware redundancy, using two independent CPU instances called the master CPU and the checker CPU. The two CPUs operate in a lockstep manner: They use the same input data, execute the same operation, and all the functional outputs of the master CPU and the checker CPU are compared on a cycle by cycle basis using a hardware comparator. The logic covered by this safety mechanism is called the area of duplication, see Section 4.1.1 Safe Computation. The lockstep operation has no effect on the software execution. Additional measures are implemented in hardware to mitigate common cause faults between the redundant CPU instances.
References	[3] Lockstep Comparator Logic (LCL) [3] Lockstep Comparator Logic (LCL) > Lockstep Monitoring [3] System Control Units > Lockstep CPU Configuration
Error Signaling	SMU alarm ALM0[0] CPU0 Lockstep Comparator Error ALM1[0] CPU1 Lockstep Comparator Error
Error Detection Time	Up to 20 clock cycles (f_{CPUx})
Product Configuration	The AURIX family of microcontroller provides devices with different configurations regarding the availability of lockstep cores. See [3] CPU Subsystem > AURIX Family CPU Configuration. For each available lockstep CPU it is possible to disable the SM1[HW].CPU:LOCKSTEP safety mechanism at every cold PORST. The configuration is controlled by the Startup Software using the BMI. The system integrator is responsible to configure the BMI according to the

	needs of the application. The configuration cannot be changed after the boot phase.
Initialization	No initialization required
Tests	Startup tests: • **SM2[AoU].CPU:LOCKSTEP.ALARM_TEST** • **SM2[AoU].CPU.TRAP:TEST**
Limitations and/or recommendations	N/A

5.1.2 Self-Checking Lockstep Comparator

ID	SM2[HW].CPU:LOCKSTEP.SCC
Description	AURIX implements a hardware built-in self-test of the CPU lockstep comparator: • Stuck-at faults in the comparator logic are detected thanks to the special design of the comparator elements that delivers a dual inverted output. A stuck-at fault in the comparator logic results in identical outputs, causing a lockstep alarm. • A continuously running hardware-based test injects a fault at every input of the comparator in a sequential manner. The complete self-test cycle repeats automatically every 16384 clock cycles. 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, see **SM1[HW].CPU:LOCKSTEP**.
References	[3] Lockstep Comparator Logic (LCL) > Lockstep Self Test
Error Signaling	Same as **SM1[HW].CPU:LOCKSTEP**
Error Detection Time	Up to 16384 clock cycles (f_{CPUx})
Product Configuration	Same as **SM1[HW].CPU:LOCKSTEP**
Initialization	No initialization required.
Tests	N/A
Limitations and/or recommendations	N/A

5.1.3 Internal Monitor for the External Supply Voltage VEXT

ID	SM1[HW].ExtVREG:MON
Description	VEXT (3.3 V / 5V external supply) is monitored by the AURIX to detect under-voltage and over-voltage on VEXT.
References	[3] System Control Unit > Power Supply and Control

	[3] System Control Unit > Voltage Monitoring [3] System Control Unit > EVR Control Registers
Error Signaling	SMU alarm ALM3[15] SCU/EVR External Supply under voltage ALM3[16] SCU/EVR External Supply over voltage
Error Detection Time	The monitored voltage is sampled every 1.8 µs
Product Configuration	N/A
Initialization	This safety mechanism is always active, but the alarm must be initialized in the SMU by the application. The voltage threshold can be configured by software in registers EVROVMON and EVRUVMON. Default threshold values are device dependent and are documented in the respective user manual [3].
Tests	[SM_AURIX_EXTVREG_TEST]If the Internal Monitor for the External Supply Voltage VEXT is used as part of the application safety concept, it shall be tested at startup with: • **SM2[AoU].ExtVREG:MON**[/req]
Limitations and/or recommendations	The operation of the embedded voltage monitors is independent from the power supply scheme. Due to its dependencies to the external supply voltage VEXT, this safety mechanism may only be used in combination with an external over-voltage monitor as described in the section 3.8.1 Power Supply and External Voltage Monitor. It is recommended to use this safety mechanism, with a threshold below the external overvoltage monitor threshold, if the application requires preparing for the shut-down that is expected if the external voltage monitor detects an overvoltage. For example the system integrator may want to store log data.

5.1.4 EVR Voltage Monitor

ID	SM1[HW].EVR13:MON, SM1[HW].EVR33:MON
Description	The internal voltage monitors detects under and over-voltage on VDD (1.3V) and VDDP3 (3.3V). This is intended to detect the failures of the embedded voltage regulators. They can also be used to monitor externally provided VDD and VDDP3 if the embedded voltage regulators are not used.
References	[3] System Control Unit > Power Supply and Control [3] System Control Unit > Voltage Monitoring [3] System Control Unit > Power Supply and Control > Supply Mode and Topology Selection
Error Signaling	SMU alarm ALM3[11] SCU/EVR EVR 1.3V digital under voltage ALM3[12] SCU/EVR EVR 1.3V digital over voltage ALM3[13] SCU/EVR EVR 3.3V under voltage ALM3[14] SCU/EVR EVR 3.3V over voltage
Error Detection Time	The monitored voltage is sampled every 1.8 µs
Product Configuration	The choice of the supply scheme at startup is based on the latched status of HWCFG[0:2] pins before PORST release and is indicated by PMSWSTAT.HWCFGEVR status flags
Initialization	This safety mechanism is always active, but the alarm must be initialized in the SMU by the application. The voltage threshold can be configured by software in registers EVROVMON and EVRUVMON.
Tests	Startup test: • **SM2[AoU].EVR:MON**
Limitations and/or recommendations	For the initialization of the voltage monitors, It is recommended to use the default thresholds values. [SM_AURIX_EVR_3]EVR13 shall not be used in Linear Drop Out (LDO) mode with external pass device for safety applications. Rationale: the EVR monitor does not provide sufficient safety performance for this mode. *Note: This assumption of use only applies for devices of the TC29x, TC27x, and TC26x series, and the LDO mode with external pass device on EVR13 may be used for all other devices.* [/req] [SM_AURIX_EVR_5]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. Rationale: voltage spikes shorter than the voltage monitoring period would not be detected and could lead to damages.[/req]

5.1.5 EVR Bandgap BIST

ID	SM1[HW].BANDGAP:MON
Description	An independent bandgap is used as a reference to monitor the EVR bandgap used by the embedded voltage regulators and the voltage monitors. A hardware built-in self-test makes a plausibility check between the two independent bandgaps after any reset. The result of the test is stored in EVRSTAT.BGPROK register. This built-in self-test does not generate an SMU alarm but triggers a warm PORST.
References	[3] System Control Unit > Power Supply and Control
Error Signaling	Application software error handler
Error Detection Time	Error is detected before starting the software execution, as part of the reset sequence
Product Configuration	N/A
Initialization	N/A
Tests	N/A
Limitations and/or recommendations	This safety mechanism is used by **SM2[AoU].EVR:MON**

5.1.6 Clock Frequency Monitors

ID	SM1[HW].VADC:CLKMON, SM1[HW].GTM:CLKMON, SM1[HW].STM:CLKMON, SM1[HW].SPB:CLKMON, SM1[HW].SRI:CLKMON
Description	AURIX provides clock frequency monitors for the following, internally generated, clocks: f_{STM} f_{SRI} f_{SPB} f_{GTM} f_{ADC} (TC27x B-Step only, other devices use f_{SPB} as clock source for the VADC) Each of these clocks is monitored by its own counter, incrementing proportionally to the monitored clock during a fixed monitoring period. The internally generated 100 MHz back-up clock, f_{BACK}, is used as diverse clock source, independent from the system PLL, to generate the monitoring period by incrementing a reference counter. An error is detected if the reference counter has an overflow and the monitor counter of the dedicated clock is either below the lower limit (clock

	too slow) or greater as the upper limit (clock too fast).
References	[3] System Control Unit > Clock Monitors
Error Signaling	SMU alarm ALM3[3] SCU/CGU Clock monitoring: Out of range frequency STM ALM3[6] SCU/CGU Clock monitoring: Out of range frequency SRI ALM3[7] SCU/CGU Clock monitoring: Out of range frequency SPB ALM3[8] SCU/CGU Clock monitoring: Out of range frequency GTM ALM3[9] SCU/CGU Clock monitoring: Out of range frequency ADC
Error Detection Time	up to 5.2 µs
Product Configuration	N/A
Initialization	This safety mechanism is not enabled by default; It must be configured by the application software to become active. To support a flexible configuration of the clock system, the upper and lower limits can be adapted, by selecting the target frequency and frequency divider for each clock.
Tests	Startup test: • **SM2[AoU].CLK:CLKMON**
Limitations and/or recommendations	[SM_AURIX_DFA_CLK_1]AURIX Clock System can use either an oscillator circuit or the internally generated Back-up Clock as the clock source. For safety related applications, it is assumed the oscillator circuit is used as clock source. Rationale: To avoid common cause failures, as the back-up clock is also used by the clock monitors, the SMU logic, and for the generation of the FSP signal.[/req] [SM_AURIX_DFA_CLK_2]The SMU shall be configured to react to a clock monitor error with a system reset (internal reaction) and/or FSP activation (external reaction). The port emergency stop feature can also be used, together with or instead of the FSP. Rationale: it cannot be guaranteed that interrupt and NMI operate correctly when the clock monitors detect the clock frequency is above the upper limit threshold.[/req] The minimum deviation for which a clock error is guaranteed to be detected depends on the configuration (target frequency) of the individual clock monitors, and is documented in the device user's manual [3] (Target Trimmed Check Limits). Any deviation of +/- 10% of the nominal frequency is guaranteed to be detected.

5.1.7 System PLL Frequency Monitor

ID	SM1[HW].SystemPLL:CLKMON
Description	AURIX provides a frequency monitor for the system PLL clock, f_{PLL}. It is similar to the other Clock Frequency Monitors, with the difference that if

	a wrong frequency is detected, the Clocking and Clock Control Unit (CCU) switches to the back-up clock. This is called the Clock Emergency Behavior.
References	[3] System Control Unit > Clock Monitors [3] System Control Unit > Clock Emergency Behavior
Error Signaling	SMU alarm ALM3[5] SCU/CGU Clock monitoring: Out of range frequency System PLL
Error Detection Time	up to 5.2 µs
Product Configuration	N/A
Initialization	To support a flexible configuration of the clock system, the upper and lower limits can be adapted by software, by selecting the target frequency and frequency divider for this clock.
Tests	Startup test: • **SM2[AoU].CLK:CLKMON**
Limitations and/or recommendations	

5.1.8 System PLL Loss-of-Lock Monitor

ID	SM1[HW].SystemPLL:LOCKMON
Description	The system PLL (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.
References	[3] System Control Unit > Phase-Locked Loop (PLL) Module [3] System Control Unit > Clock Emergency Behavior
Error Signaling	SMU alarm ALM3[1] SCU/CGU System PLL VCO Loss-of-Lock Event
Error Detection Time	up to 200µs
Product Configuration	N/A
Initialization	Always active, no initialization needed.
Tests	N/A
Limitations and/or recommendations	SM1[HW].SystemPLL:CLKMON also signals an error in case of PLL loss of lock. Therefore, the VCO Loss-of-lock Event alarm generated by this mechanism is redundant and does not necessarily need to be tested; it provides additional information about the cause of the error.

5.1.9 ERAY PLL Frequency Monitor

ID	SM1[HW].ErayPLL:CLKMON
Description	AURIX provides a frequency monitor for the E-Ray clock, f_{ERAY}. It is similar to the other Clock Frequency Monitors.
References	[3] System Control Unit > Clock Monitors
Error Signaling	SMU alarm ALM3[4] SCU/CGU Clock monitoring: Out of range frequency PLL_ERAY
Error Detection Time	up to 5.2 µs
Product Configuration	N/A
Initialization	To support a flexible configuration of the clock system, the upper and lower limits can be adapted by software, by selecting the target frequency and frequency divider for this clock.
Tests	Startup test: • **SM2[AoU].CLK:CLKMON**
Limitations and/or recommendations	N/A

5.1.10 E-Ray PLL Loss-of-Lock Monitor

ID	SM1[HW].ErayPLL:LOCKMON
Description	The E-Ray PLL (PLL_ERAY) 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.
References	[3] System Control Unit > ERAY Phase-Locked Loop (PLL_ERAY) Module
Error Signaling	SMU alarm ALM3[2] SCU/CGU PLL_ERAY VCO Loss-of-Lock Event
Error Detection Time	up to 200µs
Product Configuration	N/A
Initialization	Always active, no initialization needed.
Tests	N/A
Limitations and/or recommendations	SM1[HW].ErayPLL:CLKMON also signals an error in case of PLL loss of lock. Therefore, the VCO Loss-of-lock Event alarm generated by this mechanism is redundant and does not necessarily need to be tested; it provides additional information about the cause of the error.

5.1.11 SRAM Error Correction Code (ECC)

ID	SM1[HW].SRAM:ECC, SM1[HW].SRAM:EDC
Description	The SRAM implements SECDED (Single Error Correction Double Error Detection) ECC with a Hamming distance of 4, correcting single bit upsets and detecting double bit upsets. One exception is the PTAG SRAM ECC that generates an uncorrectable error for single and double-bit errors.
References	[3] Memory Test Unit (MTU) > ECC Implementation
Error Signaling	**Uncorrectable errors :** CPU Trap PIE Program Integrity Error DIE Data Integrity Error PSE Program Fetch Synchronous Error DSE Data Access Synchronous Error SMU alarm ALM0[3] CPU0 PCACHE TAG SRAM ECC uncorrectable error ALM0[7] CPU0 Unified PCACHE/PSPR ECC uncorrectable error ALM0[11] CPU0 Unified DCACHE/DSPR ECC uncorrectable error ALM1[3] CPU1 PCACHE TAG SRAM ECC uncorrectable error ALM1[7] CPU1 Unified PCACHE/PSPR ECC uncorrectable error ALM1[11] CPU1 Unified DCACHE/DSPR ECC uncorrectable error ALM1[15] CPU1 DCACHE TAG SRAM ECC uncorrectable error ALM2[18] LMU SRAM ECC uncorrectable error ALM4[1] GTM SRAMs non correctable error ALM4[5] CAN SRAM uncorrected error ALM4[9] FLEXRAY SRAM ECC uncorrectable error ALM4[13] EMEM SRAM ECC uncorrectable error ALM4[17] Misc SRAMs SRAM ECC uncorrectable error ALM6[3] CPU2 PCACHE TAG SRAM ECC uncorrectable error ALM6[7] CPU2 Unified PCACHE/PSPR ECC uncorrectable error ALM6[11] CPU2 Unified DCACHE/DSPR ECC uncorrectable error ALM6[15] CPU2 DCACHE TAG SRAM ECC uncorrectable error ALM3[30] SRI Shared Resource Interconnect, bus error **Correctable errors :** SMU alarm ALM0[6] CPU0 Unified PCACHE/PSPR ECC single bit correction ALM0[10] CPU0 Unified DCACHE/DSPR ECC single bit correction ALM1[6] CPU1 Unified PCACHE/PSPR ECC single bit correction ALM1[10] CPU1 Unified DCACHE/DSPR ECC single bit correction ALM1[14] CPU1 DCACHE TAG SRAM ECC correction ALM2[17] LMU SRAM ECC single bit correction notification ALM4[0] GTM SRAMs single bit correction ALM4[4] CAN SRAM single bit correction ALM4[8] FLEXRAY SRAM ECC single bit correction ALM4[12] EMEM SRAM ECC single bit correction ALM4[16] Misc SRAMs SRAM ECC single bit correction ALM6[6] CPU2 Unified PCACHE/PSPR ECC single bit correction ALM6[10] CPU2 Unified DCACHE/DSPR ECC single bit correction ALM6[14] CPU2 DCACHE TAG SRAM ECC correction
Error Detection Time	up to 10 clock cycles (f_{SPB})
Product Configuration	N/A

Initialization	• The error correction is active after reset. • The MTU module must be enabled by the system integrator or no SRAM alarm will be forwarded to the SMU (MTU_CLC). By default, after any Power-On Reset, system reset and application reset, the MTU is disabled • [SM_AURIX_SRAM_INIT]After Cold PORST, the SRAM content shall be initialized. The initialization can be done either by the application software or automatically by the SSW according to the PROCOND configuration.[/req]
Tests	Startup tests for CPU and LMU SRAMs: • **SM2[AoU].SRAM:ECC** Startup tests for peripheral SRAMs, if required: • **SM2[AoU].PSRAM:ECC**
Limitations and/or recommendations	• Please consider the information in section 6 About Error Reaction and Fault Tolerance. • Multiple SMU alarms related to correctable and uncorrectable events may occur simultaneously in reaction to some faults of the SRAM. • It is recommended to write the RDBFL registers of each SRAM instance with an invalid data pattern of data and ECC, for example All-0, before starting the safety related application software. This is to avoid the silent corruption of safety related data during application runtime, due to a failure mode of the MBIST for which the content of the RDBFL registers is written accidentally in memory. • [SM_AURIX_SRAM_ALM]CPU traps generated in case of uncorrectable SRAM error (PSE, DSE, PIE, DIE) shall be handled by the application software as dangerous failures. Rationale: in one corner case, uncorrectable error alarms may not be forwarded to the SMU, instead only traps are raised[/req]

5.1.12 SRAM Address Monitor

ID	SM1[HW].SRAM:ADDRMON
Description	The Address Monitor detects addressing failures caused by permanent or transient faults in the SRAM's logic. The Address Monitor monitors faults that lead or behave like a wrong address access to the SRAM hard macro. The address monitor monitors both read and write accesses. If the SRAM does not receive an address (no access fault) the address monitor raises per default the address error SMU alarm.
References	[3] Introduction > Safety Concept Overview
Error Signaling	SMU alarm Depending on the SRAM module where the fault occurs, one of the following SMU alarm will be raised: ALM0[4] CPU0 PCACHE TAG SRAM address error ALM0[8] CPU0 Unified PCACHE/PSPR address error ALM0[12] CPU0 Unified DCACHE/DSPR address error ALM0[16] CPU0 DCACHE TAG SRAM address error ALM1[4] CPU1 PCACHE TAG SRAM address error ALM1[8] CPU1 Unified PCACHE/PSPR address error

	ALM1[12] CPU1 Unified DCACHE/DSPR address error ALM1[16] CPU1 DCACHE TAG SRAM address error ALM6[4] CPU2 PCACHE TAG SRAM address error ALM6[8] CPU2 Unified PCACHE/PSPR address error ALM6[12] CPU2 Unified DCACHE/DSPR address error ALM6[16] CPU2 DCACHE TAG SRAM address error
Error Detection Time	up to 10 clock cycles (f_{SPB})
Product Configuration	DCACHE TAG SRAM address error alarms are only available for TC1.6P cores, as TC1.6E cores do not implement data cache. See [3] CPU Subsystem > AURIX Family CPU Configurations.
Initialization	• The Address Monitor is active after reset • The MTU module must be enabled by the system integrator or no SRAM alarm will be forwarded to the SMU (MTU_CLC). By default, after any Power-On Reset, system reset and application reset, the MTU is disabled.
Tests	Startup tests for the CPU and LMU SRAMS: • **SM2[AoU].SRAM:ADDRMON** Startup tests for peripheral SRAMs, if required: • **SM2[AoU].PSRAM:ADDRMON**
Limitations and/or recommendations	N/A

5.1.13 Lockstep SRAM ECC Monitor

ID	SM1[HW].SRAM.ECC:LOCKSTEP
Description	The Error Correction Code (ECC) logic for the CPU side memories is located in the area of duplication, therefore the failure modes of the correction logic are controlled by the CPU lockstep safety mechanism.
References	[3] CPU Subsystem > Lock Step Implementation
Error Signaling	SMU alarm ALM0[0] CPU0 Lockstep Comparator Error ALM1[0] CPU1 Lockstep Comparator Error
Error Detection Time	This is the same as **SM1[HW].CPU:LOCKSTEP**
Product Configuration	This is the same as **SM1[HW].CPU:LOCKSTEP**
Initialization	This is the same as **SM1[HW].CPU:LOCKSTEP**
Tests	This is the same as **SM1[HW].CPU:LOCKSTEP**
Limitations and/or recommendations	N/A

5.1.14 LMU SRAM ECC Monitor

ID	SM1[HW].LMU.SRAM.ECC:MONITOR
Description	The failure modes of the LMU SRAM error correction logic are controlled by a redundant and diverse ECC monitor. It covers the LMU read data path.
References	[3] Local Memory Unit (LMU) > Error Detection and Signaling > ECC check failure
Error Signaling	SMU alarm ALM2[15] LMU ECC Error
Error Detection Time	Up to 20 clock cycles (fSRI)
Product Configuration	N/A
Initialization	• The error correction is active after reset. • The MTU module must be enabled by the system integrator or no SRAM alarm will be forwarded to the SMU (MTU_CLC). By default, after any Power-On Reset, system reset and application reset, the MTU is disabled.
Tests	Startup test: • **SM2[AoU].LMU.SRAM.ECC:MONITOR**
Limitations and/or recommendations	N/A

5.1.15 SRAM Error Tracking

ID	SM1[HW].SRAM:Monitor
Description	The SRAM error tracking feature provides, for each memory controller instance, up to 5 error tracking registers (ETRRx) where the addresses of data having caused a correctable, uncorrectable, or address error event are logged. An address is logged only once, which means, further error events caused by data at the same address are not tracked. When all the error tracking registers of a memory controller instance are used and a new error event happens, a 'SRAM monitor address buffer overflow' alarm is triggered to the SMU. It can be combined with the SMU alarms or the error detection status bits in the ECC Detection Register (ECCD), to make it possible to determine which type of error happened among correctable error, uncorrectable error, and address error. If more than one type of error occurs and is logged, it is not possible to determine which error happened to which address.
References	[3] Memory Test Unit (MTU) > Memory Controllers > Error Tracking Register(s) [3] Memory Test Unit (MTU) > Memory Controllers > ECC Safety Register [3] Memory Test Unit (MTU) > Memory Controllers > ECC Detection

	Register [3] Memory Test Unit (MTU) > Implementation Section > Configurations - Memory Controller
Error Signaling	SMU alarm ALM0[13] CPU0 Unified DCACHE/DSPR address buffer overflow ALM0[5] CPU0 PCACHE TAG address buffer overflow ALM0[9] CPU0 Unified PCACHE/PSPR address buffer overflow ALM1[13] CPU1 Unified DCACHE/DSPR address buffer overflow ALM1[17] CPU1 DCACHE TAG SRAM address buffer overflow ALM1[5] CPU1 PCACHE TAG SRAM address buffer overflow ALM1[9] CPU1 Unified PCACHE/PSPR address buffer overflow ALM2[20] LMU SRAM Address buffer overflow ALM4[11] FLEXRAY SRAM address buffer overflow ALM4[15] EMEM SRAM address buffer overflow ALM4[19] Misc SRAMs SRAM address buffer overflow ALM4[7] CAN SRAM address buffer overflow ALM6[13] CPU2 Unified DCACHE/DSPR address buffer overflow ALM6[17] CPU2 DCACHE TAG SRAM Address buffer overflow ALM6[5] CPU2 PCACHE TAG SRAM address buffer overflow ALM6[9] CPU2 Unified PCACHE/PSPR address buffer overflow
Error Detection Time	up to 10 clock cycles (f_{SPB})
Product Configuration	The number of error tracking registers for a memory controller instance depends on the size of the SRAM it is controlling and was defined based on reliability targets, it can be found in [3] with the configuration parameters of the memory controllers.
Initialization	• Individually for each SRAM instance in the ECC Safety Register ECCS • The MTU module must be enabled by the system integrator or no SRAM alarm will be forwarded to the SMU (MTU_CLC). By default, after any Power-On Reset, system reset and application reset, the MTU is disabled.
Tests	Startup tests for CPU and LMU SRAMS: • **SM2[AoU].SRAM:Monitor** Startup tests for peripheral SRAMs, if required: • **SM2[AoU].PSRAM:Monitor**
Limitations and/or recommendations	Please also consider section 6 About Error Reaction and Fault Tolerance.

5.1.16 PFLASH Error Correction Code (ECC)

ID	SM1[HW].PFLASH:ECC
Description	The data in PFLASH is stored with Error Correcting Codes (ECC) in order to protect against data corruption. Data in PFLASH uses an ECC code with DEC-TED (Double Error Correction, Triple Error Detection) capabilities. Each block of 256 data bits is accompanied with a set of ECC bits. [SM_AURIX_PMU_1:1]The PFLASH implements two algorithms for ECC calculation:

	a legacy and a safety ECC. It is assumed that the safety ECC algorithm is used. Rationale: In this mode all-0 or all-1 are invalid code words. An invalid code raises a non-correctable error SMU alarm.[/req]
References	[3] Program Memory Unit (PMU) > Data Integrity and Safety
Error Signaling	SMU alarm ALM2[2] PMU PFLASH ECC single bit correction notification ALM2[3] PMU PFLASH ECC double bit correction notification ALM2[4] PMU PFLASH ECC non correctable multiple bit/addressing error
Error Detection Time	up to 10 f_{SRI} clock cycle, when a corrupted data is read from PFLASH
Product Configuration	N/A
Initialization	No initialization required. The ECC logic is operational after reset
Tests	Startup tests: • **SM2[AoU].PFLASH:ECC**
Limitations and/or recommendations	It is recommended to react only on non-correctable error alarm, as described in Section 6.3

5.1.17 PFLASH Address Error Detection

ID	SM1[HW].PFLASH:ADDRMON
Description	The PFLASH implements address error detection by including the address bits into the computation of the ECC.
References	[3] Program Memory Unit (PMU) > Data Integrity and Safety
Error Signaling	SMU alarm ALM2[4] PMU PFLASH ECC non correctable multiple bit ALM2[5] PMU PFLASH Addressing error ALM3[30] SRI Shared Resource Interconnect, bus error
Error Detection Time	up to 10 f_{SRI} clock cycle, when a corrupted data is read from PFLASH
Product Configuration	N/A
Initialization	No initialization required. The ECC logic is operational after reset
Tests	Startup tests: • **SM2[AoU].PFLASH:ADDRMON**
Limitations and/or recommendations	N/A

5.1.18 PFLASH ECC Logic Monitor

ID	SM1[HW].PFLASH.ECC:Monitor
Description	The ECC decoder has the ability to corrupt data by performing a wrong correction. To mitigate this issue the PFLASH ECC Logic Monitor continuously monitors the PFLASH ECC logic and checks that the corrected word and the ECC bit belongs to the valid codeword vector space.
References	[3] Program Memory Unit (PMU) > Signaling to the Safety Management Unit
Error Signaling	SMU alarm ALM2[7] PMU PFLASH ECC monitor error
Error Detection Time	up to 10 f_{SRI} clock cycle, when a corrupted data is read from PFLASH
Product Configuration	N/A
Initialization	No initialization required. The ECC monitor is operational after reset
Tests	[SM_AURIX_PMU_1:4]Startup tests: • **SM2[AoU].PFLASH:ECC**[/req]
Limitations and/or recommendations	N/A

5.1.19 PFLASH Redundant Error Detection Logic

ID	SM2[HW].PFLASH.EDC:CMP
Description	The error detection logic used by SM1[HW].PFLASH:ECC is implemented redundantly with an hardware comparator. This is a safety mechanism to support the latent fault metric, detecting failures of the error detection logic which cannot be covered by a software test.
References	[3] Program Memory Unit (PMU) > Signaling to the Safety Management Unit
Error Signaling	ALM2[8] PMU PFLASH EDC comparator error
Error Detection Time	Up to 10 f_{SRI} clock cycles, when a corrupted data is read from PFLASH and the two instances of the error detection logic does not provide the same result while reading data from PFLASH
Product Configuration	N/A
Initialization	No initialization required. The error detection logic is operational after reset
Tests	Startup tests: • **SM2[AoU].PFLASH.EDC:CMP**
Limitations and/or recommendations	N/A

5.1.20 PFLASH Error Tracking

ID	SM1[HW].PFLASH:Monitor
Description	The PFLASH provides a safety mechanism to monitor the number of data blocks that caused correctable errors. The safety mechanism enables the application to count the number of individual event as well as to reconstruct the system address. The same address is stored only once. The safety mechanism is implemented by the Corrected Bits Address Buffer (CBAB) and generates an alarm when the buffer is full. CBAB behaves like a FIFO and the oldest entry is accessed through a register called CBABTOP. Additionally the CBABSTAT register can be read to determine the filling level, and the CBABCFG register can be used to configure the type of faults to be captured.
References	[3] Program Memory Unit (PMU) > FLASH > Data Integrity and Safety [3] Program Memory Unit (PMU) > FLASH> Reset and Startup
Error Signaling	SMU alarm ALM2[6] PMU address buffer full
Error Detection Time	Up to 10 clock cycles (f_{SPB})
Product Configuration	Each CBAB buffer has 10 entries and there is one CBAB buffer for every Flash bank.
Initialization	The PFLASH Monitor is disabled by default. It can be initialized through the register CBABCFG. The CBAB registers are not affected by a system or application reset.
Tests	Startup tests: • **SM2[AoU].PFLASH:Monitor**
Limitations and/or recommendations	Please also consider section 6 About Error Reaction and Fault Tolerance.

5.1.21 SRI Address Error Detection

ID	SM1[HW].SRI.ADDR:EDC, SM1[HW].SRI:EDC
Description	Each SRI transaction consists of: • one request phase, also called address phase • one or multiple data phases The corruption of the address phase signals due to permanent and transient faults is detected by an Error Detection Code (EDC). If an SRI agent detects an error it triggers an alarm. In parallel it may or may not complete the transaction
References	[3] On-Chip System Buses and Bus Bridges > SRI Crossbar (Xbar_SRI) > Functional Overview > SRI ECC Error Handling
Error Signaling	SRI Bus Error SMU Alarm ALM3[30] Shared Resource Interconnect, bus error
Error Detection	Up to 20 clock cycles (f_{SRI}), when a corrupted signal is handled by an SRI

Time	slave
Product Configuration	N/A
Initialization	No initialization required. The error detection logic is operational after reset
Tests	Startup tests: • **SM2[AoU].SRI:EDC**
Limitations and/or recommendations	N/A

5.1.22 SRI Data Error Detection

ID	SM1[HW].SRI.DATA:EDC, SM1[HW].SRI:EDC
Description	Each SRI transaction consists of: • one request phase, also called address phase • one or multiple data phases The corruption of the data phase signals due to permanent and transient faults is detected by an Error Detection Code (EDC). If an SRI agent detects an error during a write access, it triggers an alarm. In parallel it may or may not complete the transaction. If an SRI agent detects an error during a read access, the transaction continues to completion. In parallel to the alarm triggering, CPUs will not use the data and will trap, while other masters such as DMA set an internal error state. The signaling of the fault differs for the different classes of SRI modules: • CPU's master and slave interfaces • Non-CPU agent's master and slave interfaces (for example DMA's)
References	'[3] On-Chip System Buses and Bus Bridges > SRI Crossbar (Xbar_SRI) > Functional Overview > SRI ECC Error Handling [2] 6.3.5 System Bus and Peripheral Errors (Trap Class 4)
Error Signaling	• CPU's master and slave interfaces CPU Trap PSE Program Fetch Synchronous Error DSE Data Access Synchronous Error DAE Data Access Asynchronous Error SMU alarm ALM0[18] CPU0 CPU Instruction Fetch SRI Interface EDC Error ALM0[19] CPU0 CPU Data SRI Interface (Load/Store) EDC Error ALM1[18] CPU1 CPU Instruction Fetch SRI Interface EDC Error ALM1[19] CPU1 CPU Data SRI Interface (Load/Store) EDC Error ALM6[18] CPU2 CPU Data SRI Interface (Load/Store) EDC Error ALM6[19] CPU2 CPU Data SRI Interface (Load/Store) EDC Error • master and slave interfaces other than CPU's (for example DMA) SMU alarm ALM2[21] SRI non-CPU SRI agents: ECC Address phase error ALM2[22] SRI non-CPU SRI agents: ECC Write phase error ALM2[23] SRI non-CPU SRI agents: ECC Read phase error

Error Detection Time	Up to 20 clock cycles (f_{SRI}), when corrupted signal is received by a SRI agent
Product Configuration	N/A
Initialization	No initialization required. The error detection logic is operational after reset
Tests	Startup tests: • **SM2[AoU].SRI:EDC**
Limitations and/or recommendations	N/A

5.1.23 SRI Error Handler

ID	SM1[HW].SRI:EH, SM1[HW].SRI:EH.ES, SM1[HW].SRI:EH.TID
Description	SRI protocol errors are detected by the SRI Crossbar bus and are signaled with an interrupt and an SMU alarm. The XBar_SRI module additionally implements error tracking registers providing diagnostic information. This safety mechanisms detect the following errors: • access level is incorrect (user/supervisor) • unmapped address (within the slave address space) access from an SRI master • unsupported op-code (the SRI op-code defines for a SRI transaction the number of data phases, the addressing mode in case of a multi beat transaction and valid bytes in case of a single data transaction) • reserved op-code • starvation of masters • transaction ID errors
References	[3] On-Chip System Buses and Bus Bridges > SRI Crossbar (XBar_SRI) > SRI Error Conditions [3] On-Chip System Buses and Bus Bridges > SRI Crossbar (XBar_SRI) > SRI Transaction ID Error Conditions [3] On-Chip System Buses and Bus Bridges > SRI Crossbar (XBar_SRI) > Functional Overview > Default Slave [3] On-Chip System Buses and Bus Bridges > SRI Crossbar (XBar_SRI) > Functional Overview > Error Tracking Capability
Error Signaling	Interrupt service request XBAR_SRI Service Request SMU alarm ALM3[30] SRI Shared Resource Interconnect, bus error
Error Detection Time	Detected during the transaction. Faulty transaction is terminated in the next cycle with SRI error acknowledge.
Product Configuration	N/A
Initialization	All the errors are captured by default but the system integrator can enable/disable the capture and the interrupt generation for specific type of error in registers XBAR_ARBCONx and XBAR_IDINTSAT. The SMU alarm

	generation is always active, but the corresponding alarm must be configured in the SMU.
Tests	Startup tests: • **SM2[AoU].SRI:EH**
Limitations and/or recommendations	N/A

5.1.24 SPB Error Handler

ID	SM1[HW].SPB:EH, SM1[HW].SPB:TIMEOUT
Description	SPB protocol errors are detected by the System Peripheral Bus Control Unit (SBCU) and are signaled with an interrupt and an SMU alarm. The SBCU additionally implements error tracking registers providing diagnostic. The following error cases are detected: • Slave acknowledged the transaction with an error • Un-implemented address: no slave respond to a transaction • Time-out: A slave does not respond to a transaction request within a certain time window. The number of bus clock cycles that can elapse until a bus time-out is defined by bit field SBCU_CON.TOUT
References	[3] On-Chip System Buses and Bus Bridges > FPI Bus Error Handling
Error Signaling	Interrupt service request SPB Service Request SMU alarm ALM3[31] System Peripheral Bus, bus error
Error Detection Time	• Error acknowledge, Un-implemented address: detected during the transaction. Faulty transaction are aborted • Timeout: configurable
Product Configuration	N/A
Initialization	The timeout value can be configured in register SBCU_CON
Tests	Startup tests: • **SM2[AoU].SPB:EH** • **SM2[AoU].SPB:TIMEOUT** • **SM2[AoU].SBCU:InitCheck**
Limitations and/or recommendations	N/A

5.1.25 IR Error Detection Code

ID	SM1[HW].IR:EDC
Description	The Interrupt Router (IR) implements an Error Detection Code providing an

	end-to-end protection of the data path from Service Request Node (SRN), through Interrupt Control Unit (ICU), all the way to the Interrupt Service Provider (CPU or DMA).
References	[3] Interrupt Router (IR) > ECC Encoding (ECC) [3] Interrupt Router (IR) > Interrupt Control Unit (ICU)
Error Signaling	SMU alarm ALM2[25] IR Interrupt Monitoring: EDC error
Error Detection Time	Error is detected when the IR receives the interrupt acknowledge back from the interrupt service provider (CPU, DMA)
Product Configuration	N/A
Initialization	Active by default
Tests	Startup Test: • **SM2[AoU].IR:EDC**
Limitations and/or recommendations	• Following failure modes are covered by this safety mechanism: o Activation of disabled SRN o Incorrect SRN request to a wrong interrupt service provider o Corruption of SRN configuration information o SRN aliasing • For the following Failure Modes which are not covered by the safety mechanism, system level safety mechanisms are necessary, as listed in section 4.20: o Wrong Interrupt Arbitration o Incorrect input trigger (too few, to many, no trigger) • This safety mechanism is a passive mechanism; on detection of an error, the faulty service request is not stopped by the hardware. Therefore, It is recommended to configure the SMU internal action in case of ALM2[25] to be a reset or an NMI: o reset will stop any interrupt processing such interrupt service routine and DMA transfers which might have been started by mistake because of an interrupt router failure. o NMI will also preempt any interrupt service routine, but will not stop the DMA transfers; application software should consider this.

5.1.26 DMA Internal Data Path Error Detection Code

ID	SM1[HW].DMA:EDC
Description	The DMA implements an Error Detection Code (EDC) on the internal data path between SPB and SRI domain. EDC extensions are generated when the read data is stored in the Move Engine Read Register. Data corruption is checked by an EDC decoder just before writing the data to the source over the SRI bus. Within the SRI domain, the DMA uses the SRI Error Detection Code safety

	mechanism to monitor its internal hardware path.
References	[3] Direct Memory Access (DMA) > Data Integrity
Error Signaling	SMU alarm ALM2[23] SRI EDC Read Phase Error Interrupt service request DMA Error Service Request
Error Detection Time	Error is detected before the DMA move engine provides a corrupted data to the bus
Product Configuration	N/A
Initialization	No initialization required. This safety mechanism is active after reset
Tests	Covered by **SM2[AoU].SRI:EDC**
Limitations and/or recommendations	N/A

5.1.27 DMA Timestamp

ID	SM1[HW].DMA:TIMESTAMP
Description	The DMA supports the generation of a 32 bit time stamp from a continuously running timer. If this safety mechanism is enabled, the current counter value is appended by a DMA write move to the next word aligned address in the destination data sequence. This mechanism can be used by the application software to detect failures of the DMA module such as "No DMA transfer", "unintended DMA transfer", and "Wrong DMA transfer". It can also be used to monitor the freshness of transmitted data and detect, for example, that inputs capture is delayed.
References	[3] Direct Memory Access (DMA) > Flow Control
Error Signaling	N/A
Error Detection Time	N/A
Product Configuration	N/A
Initialization	This safety mechanism must be enabled by the application software for every channel independently
Tests	Startup Test: • **SM2[AoU].DMA:TIMESTAMP**
Limitations and/or recommendations	The DMA Timestamp is not supported the following operation modes of the DMA: • Double Buffering • Conditional Linked List • Pattern Detection In Destination Circular Buffer mode, the DMA Timestamp is supported only with the following settings:

	• ADICRz.DMF = 001B (Address offset is 2 x CHCFGRz.CHDW) • AND CHCFGRz.CHDW = 010B (32-bit data width for moves, SDTW) • ADICRz.INCD = 1B (increment of DMA destination address)

5.1.28 Flexible CRC Engine (FCE)

ID	SM1[HW].FCE:CRC32, SM1[HW].FCE:CRC16, SM1[HW].FCE:CRC8
Description	The Flexible CRC Engine (FCE) implements the following CRC polynomials: • IEEE 802.3 CRC32 Ethernet polynomial • CCITT CRC16 polynomial • SAE J1850 CRC8 polynomial Additionally the FCE can be configured to automatically check the CRC result over a configurable message length. If the result differs from the expected one, an interrupt is generated.
References	[3] Flexible CRC Engine (FCE)
Error Signaling	Application software error handler
Error Detection Time	N/A
Product Configuration	Only available in TC29x, TC27x and TC26x
Initialization	[SM_AURIX_FCE_1]The CRC initial value shall be initialized in the CRC Engine Initialization Register CRCm before feeding the data to the FCE input register.[/req]
Tests	Startup Test: • **SM2[AoU].FCE:TEST**
Limitations and/or recommendations	N/A

5.1.29 CPU CRC Instruction

ID	SM1[HW].CPU:CRC32
Description	The TC1.6E and TC1.6P CPUs provides a parallel CRC 32-bit engine which implement the IEEE 802.3 polynomial: P(x) = x32 +x26 +x23 +x22 +x16 +x12 +x11 +x10 +x8 +x7 +x5 +x4 +x2 +x+1 It can be used by the application software through the CRC32 instruction
References	[2] Instruction Set > CPU Instructions
Error Signaling	Application software error handler
Error Detection Time	N/A
Product Configuration	N/A
Initialization	N/A

Tests	For Lockstep CPU: a failure of the CRC32 CPU instruction is detected by the safety mechanism SM1[HW].CPU:LOCKSTEP, and no startup test is necessary [SM_AURIX_CPUCRC32_1]For Non-Lockstep CPU - Startup Test: if it is used, this CRC32 CPU instruction shall be tested by computing a CRC32 over a known data range. [/req]
Limitations and/or recommendations	N/A

5.1.30 DMA Data CRC

ID	SM1[HW].DMA.DATA:CRC32
Description	The DMA provides a parallel CRC 32-bit engine which implements the IEEE 802.3 polynomial: P(x) = x32 +x26 +x23 +x22 +x16 +x12 +x11 +x10 +x8 + x7 +x5 +x4 +x2 +x+1. The DMA engine computes a CRC32 over the data read from the source. This CRC is stored in the register DMA_RDCRCRz, where z is the channel number, and can be used to detect the corruption of the data at the destination. The verification of the CRC is done by the application. Depending on the DMA channel mode, the CRC is accumulated over a DMA transaction or over all DMA transactions of a DMA linked list.
References	[3] Direct Memory Access (DMA) > DMA Channel Functionality
Error Signaling	Application software error handler
Error Detection Time	N/A
Product Configuration	N/A
Initialization	The CRC computation is always active
Tests	Startup Test: • **SM2[AoU].DMA.DATA:CRC32**
Limitations and/or recommendations	N/A

5.1.31 DMA Address CRC

ID	SM1[HW].DMA.ADDR:CRC32
Description	The DMA implements a parallel CRC 32-bit engine using the IEEE 802.3 polynomial:

	P(x) = x32 +x26 +x23 +x22 +x16 +x12 +x11 +x10 +x8 + x7 +x5 +x4 +x2 +x+1. The DMA engine computes a CRC over the source and destination addresses. This CRC is stored in the register DMA_SDCRCRz, where z is the channel number, and can be used to detect the corruption of the source or destination addresses. The verification of the CRC is done by the application. Depending on the DMA channel mode, the CRC is accumulated over a DMA transaction or over all DMA transactions of a DMA linked list.
References	[3] Direct Memory Access (DMA) > DMA Channel Functionality
Error Signaling	Application software error handler
Error Detection Time	N/A
Product Configuration	N/A
Initialization	The CRC computation is always active
Tests	Startup Test: • **SM2[AoU].DMA.ADDR:CRC32**
Limitations and/or recommendations	If the DMA channel is configured for Conditional Linked List, the SDCRC register is used as an address pointer. The DMA address CRC safety mechanism is not available. If the DMA channel is configured for: • DMA Double Source Buffering (Software Switch Only) • DMA Double Source Buffering (Automatic Hardware and Software Switch) • DMA Double Destination Buffering (Software Switch Only) • DMA Double Destination Buffering (Automatic Hardware and Software Switch) It is expected that the DMA shall not be able to store a DMA address CRC for each of the double buffers.

5.1.32 SMU Configuration Lock

ID	SM1[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.
References	[3] Safety Management Unit (SMU) > Functional Description > Register Properties > Register Write Protection
Error Signaling	None; this safety mechanism silently block write accesses to SMU configuration registers
Error Detection Time	N/A
Product Configuration	N/A

Initialization	The SMU lock is activated by software, over the SMU command interface
Tests	Startup Test: • **SM2[AoU].SMU.LOCK:TEST**
Limitations and/or recommendations	N/A

5.1.33 SMU Recovery Timer

ID	SM2[HW].SMU:RT
Description	The SMU implements two Recovery Timers, RT0 and RT1, to monitor the response time of a NMI or interrupt triggered by an alarm. If the RT is not serviced before it times out, it issues an internal SMU alarm.
References	[3] Safety Management Unit (SMU) > Recovery Timer
Error Signaling	SMU alarm ALM2[29] SMU Recovery Timer 0 Timeout ALM2[30] SMU Recovery Timer 1 Timeout
Error Detection Time	depends on the timeout value configured by the application
Product Configuration	N/A
Initialization	• RT0 and RT1 can be enabled / disabled by the application (RT0 is enabled by default as it is required for the operation of the CPU watchdogs) • The recovery time can be configured by the application software • The application can map up to 4 alarms to one recovery timer
Tests	Startup Test: • **SM2[AoU].SMU.RT:TEST**
Limitations and/or recommendations	N/A

5.1.34 CPU Memory Protection

ID	SM1[HW].CPU.CODE:MPU, SM1[HW].CPU.DATA:MPU
Description	The CPU Memory Protection Unit (MPU) is a hardware mechanism that protects user-specified memory ranges from unauthorized read, write, or instruction fetch accesses by the software running on the CPU. The MPU provides range-based protection of instruction and data memory. A protection range is a contiguous part of the address space for which access permissions are specified. A protection range is defined by the lower and upper boundary. A complete set of protection range registers is called a protection set. Each protection set then defines data access and instruction-fetch permissions. Hardware manages changing the protection set on a task context switch. Each task is assigned code execution ranges, for application code and shared code (libraries), and data ranges for local data, constants (read only), global data and local stack.
References	[2] Memory Protection System [3] CPU Subsystem > CPU Memory Protection Registers
Error Signaling	CPU trap MPR Memory Protection Read MPW Memory Protection Write MPX Memory Protection Execution
Error Detection Time	The unauthorized access prohibited and trap processing commences in the next cycle.
Product Configuration	All the microcontrollers of the AURIX family provide 4 protection sets per CPU. Each protection set support up to 16 data protection ranges and up to 8 code protection ranges.
Initialization	The memory protection is disabled after reset and no protection ranges are defined. The application must initialize the protection ranges and permissions adapted to the software architecture. Dynamic reconfiguration of the protection ranges and permissions is supported.
Tests	For Lockstep and Non-Lockstep CPU - Startup Test: • **SM2[AoU].CPU.MPU:TEST** For Non-Lockstep CPU, an additional software is necessary: • **SM2[AoU].CPU.MPU:InitCheck**
Limitations and/or recommendations	N/A

5.1.35 Bus Memory Protection (CPU)

ID	SM1[HW].CPU.BUS:MPU
Description	The Bus Memory Protection of the CPU is a hardware mechanism that protects user-specified memory ranges of the CPU's local memories (DSPR, PSPR) from unauthorized write accesses through the SRI slave interface.

	For example, with this mechanism, the system integrator can restrict write access from CPU0 to CPU1 DSPR to a dedicated memory region used as communication buffer between the two cores. Similarly this mechanism can restrict the write access from DMA channels. For each memory range one or more master can be authorized. Used together with SM1[HW].CPU:STI and SM1[HW].DMA:DMTI, this mechanism support complex bus memory protection schemes: • Only specifically identified software (safe software) is allowed to write to a region • Only specifically identified DMA channels are allowed to write to a region
References	[3] CPU Subsystem > Safety Memory Protection
Error Signaling	SMU alarm ALM0[1] CPU0 MPU error (Registers, DSPR, PSPR) ALM1[1] CPU1 MPU error (Registers, DSPR, PSPR) ALM6[1] CPU2 MPU error (Registers, DSPR, PSPR)
Error Detection Time	Up to 20 clock cycles (f_{CPUx}), when an unauthorized access is attempted
Product Configuration	All the microcontrollers of the AURIX family support up to 8 regions per CPU local memory (PSPR and DSPR together).
Initialization	After reset, the Bus memory protection allows access to the whole addressable space, to all masters. The application must initialize the protection ranges and permissions adapted to the software architecture.
Tests	For Lockstep and Non Lockstep CPU - Startup Test: • **SM2[AoU].CPU.BUS.MPU:TEST** For Non Lockstep CPU, see also **SM2[AoU].CPU.BUS.MPU:InitCheck**
Limitations and/or recommendations	N/A

5.1.36 Bus Memory Protection (LMU)

ID	SM1[HW].LMU.BUS:MPU
Description	The Bus Memory Protection of the LMU is a hardware mechanism protecting user-specified memory ranges of the LMU's shared memory from unauthorized write accesses through the SRI slave interface. For example, with this mechanism, the system integrator can restrict write access from CPU0 to LMU SRAM to a dedicated memory region used as communication buffer between CPU0 and another core. Similarly this mechanism can restrict the write access from DMA channels. For each memory range one or more master can be authorized. Used together with SM1[HW].CPU:STI and SM1[HW].DMA:DMTI, this mechanism support complex bus memory protection schemes: • Only specifically identified software (safe software) is allowed to write to a region • Only specifically identified DMA channels are allowed to write to a

	region
References	[3] Local Memory Unit (LMU) > Memory Protection
Error Signaling	SMU alarm ALM2[16] LMU Configuration and Memory Access Protection error
Error Detection Time	Up to 20 clock cycles (f_{SRI})
Product Configuration	Only available on devices implementing the shared LMU SRAM (TC27x, TC29x). These devices support up to 8 regions.
Initialization	After reset, the Bus memory protection allows access to the whole addressable space, to all masters. The application must initialize the protection ranges and permissions adapted to the software architecture.
Tests	See **SM2[AoU].LMU.MPU:TEST** and **SM2[AoU].LMU.MPU:InitCheck**
Limitations and/or recommendations	N/A

5.1.37 CPU Privilege Levels

ID	SM1[HW].CPU:SV, SM1[HW].CPU:USER0, SM1[HW].CPU:USER1
Description	The CPUs implements privilege levels. Each task is allocated a mode, depending on the task's function: • User-0 Mode: Used for tasks that may not access peripheral modules Tasks in this mode cannot enable or disable interrupts. • User-1 Mode: Used for tasks that may access common, unprotected peripherals. Typically this would be a read or write access to serial port, a read access to timer, and most I/O status registers. User-1 tasks may enable and disable interrupts (default behavior). This ability may be removed if desired by configuration of the system control register SYSCON appropriately. • Supervisor Mode: Permits read/write access to system registers and all peripheral modules. Tasks in this mode may disable interrupts. The privilege mode is controlled by the I/O privilege bit field PSW.IO. See also: **SM1[HW].Register:SV**
References	[2] Architecture Overview > Protection System
Error Signaling	CPU trap PRIV Privileged Instruction MPP Memory Protection Peripheral Access DSE Data Access Synchronous Error DAE Data Access Asynchronous Error
Error Detection Time	PRIV, MPP, DSE are synchronous traps and the instruction/operation is prohibited and the trap handler commences in the next cycle. DAE is asynchronous to the instruction flow - when the trap handler is commenced, the store instruction which violated the required privilege will already have been executed.
Product	N/A

Configuration	
Initialization	N/A
Tests	covered by • **SM2[AoU].SRI:EH**
Limitations and/or recommendations	Interrupt and trap routines start by default in supervisor mode. If required their privilege level can be changed to a user mode.

5.1.38 Register Privilege Level

ID	SM1[HW].Register:SV
Description	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. See also: **SM1[HW].CPU:SV**
References	[2] Architecture Overview > Protection System [3] Introduction > Register Access Modes
Error Signaling	SMU alarm ALM3[30] SRI Shared Resource Interconnect, bus error ALM3[31] SPB System Peripheral Bus, bus error
Error Detection Time	Up to 10 clock cycles (f_{SPB}), when an unauthorized access is attempted
Product Configuration	N/A
Initialization	N/A
Tests	Startup tests: • **SM2[AoU].SRI:EH**
Limitations and/or recommendations	N/A

5.1.39 CPU Register Access Protection

ID	SM1[HW].CPU:AP
Description	All CPUs implements a register access protection for their registers. With this safety mechanism the CPU CSFRs can only be written by allowed bus masters. Write accesses from forbidden masters are blocked and cause a bus error.
References	[3] CPU Subsystem > Safety Register Protection
Error Signaling	SMU Alarm ALM0[1] CPU0 MPU error (Registers, DSPR, PSPR) ALM1[1] CPU1 MPU error (Registers, DSPR, PSPR) ALM6[1] CPU2 MPU error (Registers, DSPR, PSPR)
Error Detection Time	Up to 10 clock cycles (f_{CPU}), when an unauthorized access is attempted
Product Configuration	N/A
Initialization	After reset, the register access protection allows access to CPU CSFR from all the SRI masters. The application must select the authorized master by initializing the ACCEN0 register. This register is Safety Endinit protected (see SM1[HW].SWDT:SE).
Tests	Startup tests: • **SM2[AoU].CPU.AP:TEST**
Limitations and/or recommendations	N/A

5.1.40 LMU Register Access Protection

ID	SM1[HW].LMU:AP
Description	The LMU implements a register access protection for its registers. With this safety mechanism the LMU registers can only be written by selected bus masters. Write accesses from forbidden masters are blocked and cause a bus error. In devices implementing a FFT module, this safety mechanism also protects the FFT registers.
References	[3] Local Memory Unit (LMU) > LMU Register Protection
Error Signaling	SRI Bus Error SMU Alarm ALM3[30] SRI Shared Resource Interconnect, bus error and, ALM2[16] Register Access Protection error
Error Detection Time	The unauthorized access is blocked
Product Configuration	N/A
Initialization	After reset, the register access protection allows access to LMU registers from all masters. The application must select the authorized master by initializing the ACCEN0 register. This register is Safety Endinit protected (see SM1[HW].SWDT:SE).
Tests	See **SM2[AoU].LMU.AP:TEST**
Limitations and/or recommendations	N/A

5.1.41 SPB Register Access Protection

ID	SM1[HW].CAN:AP, SM1[HW].IR:AP, SM1[HW].QSPI:AP, SM1[HW].SCU:AP, SM1[HW].SMU:AP, SM1[HW].VADC:AP, SM1[HW].SBCU:AP, SM1[HW].STM:AP, SM1[HW].FCE:AP, SM1[HW].IOM:AP, SM1[HW].GTM:AP, SM1[HW].CCU6:AP, SM1[HW].ERAY:AP,SM1[HW].PORT:AP, SM1[HW].DSADC:AP, SM1[HW].ASCLIN:AP, SM1[HW].HSSL:AP, SM1[HW].PSI5:AP, SM1[HW].PSI5-S:AP, SM1[HW].SENT:AP, SM1[HW].Ethernet:AP, SM1[HW].MTU:AP, SM1[HW].I2C:AP
Description	The modules connected on the SPB implements a register access protection. With this safety mechanism all the configuration registers can only be written by selected masters. Write accesses from forbidden masters are blocked and cause a bus error.
References	[3] look for Register Access Protection (ACCEN1/0), Access Protection Registers, Register Access Control [3] Controller Area Network Controller (MultiCAN+) > MultiCAN+ Module External Registers [3] FlexRay™ Protocol Controller (E-Ray) > E-Ray Module Implementation [3] Interrupt Router (IR) > Register Access Protection (ACCEN1/0)

	[3] Queued Synchronous Peripheral Interface (QSPI) > BPI_FPI Module Registers [3] System Control Unit > SCU Access Restriction Registers [3] Safety Management Unit (SMU) > System Registers description [3] On-Chip System Buses and Bus Bridges > System Peripheral Bus > FPI Bus Control Unit (SBCU) > FPI Bus Error Handling
Error Signaling	FPI Bus Error SMU Alarm ALM3[31] System Peripheral Bus, bus error Note: some modules, such as IR, additionally triggers ALM3[22] Access Enable Error. This alarm can be ignored if ALM3[31] is handled.
Error Detection Time	The unauthorized access is blocked
Product Configuration	N/A
Initialization	[SM_AURIX_AP_1]After reset, the register access protection allows access to the registers from all the bus masters. The application must select the authorized master by initializing the ACCEN0 register.[/req]
Tests	Startup tests: • **SM2[AoU].Register:AP**
Limitations and/or recommendations	[SM_AURIX_ADC_2:1]For the VADC module, the register access protection covers all the registers that belong to all VADC instances. Additional application-level measures are required to verify the integrity of the redundant channels when monitoring concepts such as described in Redundant Acquisition Using two VADC Groups 4.25.1 is used. Based on the VADC architecture it is not possible to separate the non-safety related channels from channels supporting a safety function, therefore special care shall be taken at software level to control possible interference between safety related software and non-safety related software when accessing ADC resources.[/req]

5.1.42 SRI Register Access Protection

ID	SM1[HW].SRI:AP, SM1[HW].PMU:AP
Description	The SRI configuration registers and SRI slaves implements a register access protection. With this safety mechanism all the configuration registers can only be written by allowed masters. Write accesses from forbidden masters are blocked and cause a bus error.
References	[3] On-Chip System Buses and Bus Bridges > SRI Crossbar (Xbar_SRI) > Functional Overview > Register Access Protection [3] Program Memory Unit (PMU) > Master Specific Access Control
Error Signaling	SRI Bus Error SMU Alarm ALM3[30] Shared Resource Interconnect, bus error
Error Detection Time	The unauthorized access is blocked
Product Configuration	N/A

Initialization	After reset, the register access protection allows access to SRI registers from all the SRI masters. The application must select the authorized master by initializing the ACCEN0 register. This register is Safety Endinit protected (see SM1[HW].SWDT:SE).
Tests	[SM_AURIX_SRI_2:1]Startup tests: • **SM2[AoU].SRI:AP**[/req]
Limitations and/or recommendations	N/A

5.1.43 DMA Register Access Protection

ID	SM1[HW].DMA:AP
Description	For the DMA module, the register access protection is configured by partition. Each DMA channel including its transaction control set is assigned to one hardware resource partition. This assignment in configurable by software. Each partition has its own set of access enable registers, enabling the system integrator to build a partitioning scheme adapted to the application. For example, one could assign one partition for QM related DMA channels, one for ASIL A/B related channels and one for ASIL C/D related channels.
References	[3] Direct Memory Access (DMA) > Functional Safety Features > Access Protection [3] Direct Memory Access (DMA) > DMA Access Protection Registers
Error Signaling	SRI Bus Error SMU Alarm ALM3[30] Shared Resource Interconnect, bus error
Error Detection Time	The unauthorized access is blocked
Product Configuration	All the DMA modules of the AURIX family support 4 hardware resource partitions.
Initialization	After reset, all DMA channels are allocated to hardware resource partition 0, and the register access protection allows access from all the SRI masters. The application must allocate DMA channels to hardware resource partitions and select the authorized master by initializing the ACCEN0 register. This register is Safety Endinit protected (see SM1[HW].SWDT:SE).
Tests	[SM_AURIX_DMA_2:1]Startup tests: • **SM2[AoU].DMA:AP**[/req]
Limitations and/or recommendations	N/A

5.1.44 DMA Master TAG Identifier

ID	SM1[HW].DMA:DMTI

Description	Each DMA channel is assigned to one hardware resource partition. Each hardware resource partition supports a unique master tag identifier which allows the access protection system to distinguish between the hardware resources requesting a write access. This feature is used in the context of the register access protection and/or bus MPU.
References	[3] Direct Memory Access (DMA) > Access Protection
Error Signaling	N/A
Error Detection Time	N/A
Product Configuration	All the DMA modules of the AURIX family support 4 hardware resource partitions.
Initialization	By default, the DMA channels are allocated to the hardware partition 0. If the application uses this mechanism it must allocate each channel to a partition depending on the software architecture.
Tests	covered by the tests of the register access protection and Bus MPU
Limitations and/or recommendations	N/A

5.1.45 DMA Privilege Level

ID	SM1[HW].DMA:MSEL
Description	Some modules implement registers that are accessible only by software in supervisor mode. To allow such registers to be written by DMA channels, the DMA channels are also configured to function with User Mode or Supervisor Mode. Each DMA channel is assigned to one hardware resource partition. Each hardware resource partition can be configured so that the channels allocated to this partition access resources in User Mode or in Supervisor Mode. This feature is used in the context of the register supervisor protection **SM1[HW].Register:SV**.
References	[3] Direct Memory Access (DMA) > Access Protection
Error Signaling	N/A
Error Detection Time	N/A
Product Configuration	All the DMA modules of the AURIX family support 4 hardware resource partitions.
Initialization	By default, the DMA channels are allocated to the hardware partition 0. If the application uses this mechanism it must allocate each channel to a partition depending on the software architecture.
Tests	covered by the tests of **SM1[HW].Register:SV**.
Limitations and/or recommendations	N/A

5.1.46 Endinit and Safety Endinit Protection

ID	SM1[HW].WDT0:CE0, SM1[HW].WDT1:CE1, SM1[HW].WDT2:CE2, SM1[HW].SWDT:SE, SM1[HW].WDT:CE
Description	There are a number of registers in the device that are usually programmed only once during the initialization sequence of the system or application. Modification of these registers during a normal application run can have a severe impact on the overall operation of modules or the entire system. Such registers are protected by the Endinit mechanism. Any attempt to write an Endinit protected register without deactivating the protection is discarded, the register value is not changed, and a bus error is generated. The duration of the Endinit protection deactivation is also controlled and monitored by the watchdog timers; if the Endinit is not terminated by software, a timeout event is issued and forwarded to the SMU watchdog timeout alarm. When the Endinit protection is deactivated all the Endinit-protected registers can be written to. Each watchdog provides an Endinit signals, some registers are protected by the signal from a specific watchdog. CPU0 Endinit is used by CPU0 as an additional protection level to control write accesses to its registers. The list of CPU0 registers protected by CPU0 Endinit is specified in the User Manual. Similarly CPU1 and CPU2, if available, have their own Endinit signal. The Global Endinit signal is the or of all the CPU Endinit signals. The Safety Endinit signal is controlled over the Safety Watchdog and protects access to configuration registers of safety mechanisms.
References	[3] System Control Unit > The Endinit Functions [3] System Control Unit > Determining WDT Periods
Error Signaling	SMU alarm ALM3[30] Shared Resource Interconnect, bus error ALM3[31] System Peripheral Bus, bus error
Error Detection Time	Depends on configuration of the watchdog counter frequency and of fSPB.
Product Configuration	All the devices of the AURIX family implement one watchdog instance per CPU, plus one instance as safety watchdog.
Initialization	This safety mechanism is always active. No initialization by the application software is required. The watchdog counter frequency can be configured by the application.
Tests	Covered by: • **SM2[AoU].WDT:TEST**
Limitations and/or recommendations	The usage of the Endinit protection interferes with the watchdog password function which can be used for program flow monitoring. If a watchdog is used at runtime for program flow monitoring it should not be used, at runtime, to activate the Endinit signal. In multicore devices, It should be ensured by the application software that only one core performs Safety ENDINIT access at a time, because if the

	proper sequence to access the safety watchdog is violated, the SMU alarm ALM3[17] is triggered.

5.1.47 Safety Task Identifier

ID	SM1[HW].CPU:STI
Description	CPU Safety Task Identifier. Specifies if the current task should be identified as a Safe Task. The Safety Task Identifier is used by the CPU hardware to select an alternate master identifier when a load/store instruction accesses a SoC-level resource. Note: the Safety Task Identifier does not apply to local DSPR but applies to PSPR accesses as loads and stores to the PSPR occur over the SRI and so the master tag is present to the bus MPU and access protection mechanisms. The Safety Task Identifier is used in the context of the register access protection and/or bus MPU. In each CPU, the usage of the safety task identifier is controlled by the safety task identifier 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 ID associated with that task.
References	[2] General Purpose and System Registers > Program State Information Registers [3] CPU Subsystem > SRI Safe Data Master tag
Error Signaling	N/A
Error Detection Time	N/A
Product Configuration	N/A
Initialization	N/A
Tests	Covered by **SM2[AoU].CPU.AP:TEST**
Limitations and/or recommendations	N/A

5.1.48 CPU Temporal Protection

ID	SM1[HW].CPU:TPS
Description	To guard against task runtime overrun each CPU implements a temporal protection system. This temporal protection system consists of three independent decrementing counters (TPS_TIMERn) which generates a trap on decrement to zero. The operating system can use the feature to detect time budget overruns in the system.
References	[2] Temporal Protection System [3] CPU Subsystem > Temporal Protection Registers
Error Signaling	CPU trap TAE Temporal Asynchronous Error

Error Detection Time	N/A
Product Configuration	N/A
Initialization	N/A
Tests	No dedicated startup test needed as the temporal protection system is included in the area of duplication of the lockstep CPU.
Limitations and/or recommendations	N/A

5.1.49 Watchdog Timers

ID	SM1[HW].WDT, SM1[HW].SWDT
Description	The device implements one dedicated watchdog timer WDTx for each CPU and one additional system safety watchdog SWDT. They can be used by the application to implement a safety mechanism monitoring the execution of the software. See SM1[AoU].WDT:Timeout. To service these watchdogs a password must be written in the register WDTxCON0. If the application software does not service a watchdog with the correct password, at the correct time, the watchdog generates an SMU alarm. The watchdogs can be configured to use a static or a pseudo-random sequence password. • .WDT:SPM,SM1[HW].SWDT:SPM]In static password mode, the password can be changed by software. To perform this action the software writes back the current password with specific bits inverted. Note: this mode can be used to implement a software-controlled password sequencing. • .WDT:APS,SM1[HW].SWDT:APS]The automatic pseudo-random password sequencing mode is based on a 14-bit Fibonacci LFSR (Linear Feedback Shift Register). When the application software services the watchdog, the next password is automatically updated. The initial password value can be set by software. This feature can be used by the application to implement a safety mechanism monitoring the program flow. See SM1[AoU].WDT:PFM.
References	[3] System Control Unit > Watchdog Timers
Error Signaling	SMU alarm ALM3[17] SCU/WDTS Safety Watchdog time-out ALM3[18] SCU/WDTCPU0 Watchdog CPU0 time-out ALM3[19] SCU/WDTCPU1 Watchdog CPU1 time-out ALM3[20] SCU/WDTCPU2 Watchdog CPU2 time-out ALM3[21] SCU/WDT Watchdog time-out. This alarm is a logical OR over all watchdog time-out alarms

Error Detection Time	N/A
Product Configuration	All the devices of the AURIX family implements one watchdog instance per CPU, plus one instance as safety watchdog.
Initialization	N/A
Tests	Startup tests: • **SM2[AoU].WDT:TEST** • **SM2[AoU].WDT:InitCheck**
Limitations and/or recommendations	The usage of the Endinit protection interferes with the watchdog password function which can be used for program flow monitoring. If a watchdog is used at runtime for program flow monitoring it should not be used, at runtime, to activate the Endinit signal.

5.1.50 Register Monitor (Safety Flip-Flops)

ID	SM1[HW].Register:SFF, SM1[HW].Register:FTSFF
Description	Some registers are implemented by using redundant flip-flops with comparator logic providing an alarm to the SMU. This safety mechanism detects permanent and transient errors. Registers which control the reset, the clock, the test mode, or the activation of safety mechanisms are monitored by this safety mechanism in following modules: SMU, SCU, and MTU. Additionally some special configuration registers of each SRAM instance are protected with fault tolerant redundant safety flip-flops. This safety mechanism does not provide an alarm to the SMU but it makes each bit of the registers robust against the failure of one of the redundant flip flops (2 out of 3 voting). This safety mechanism, identified as SM1[HW].Register:FTSFF, is not implemented in TC27x B-Step
References	[3] Safety Management Unit (SMU) > Safety Flip-Flops
Error Signaling	SMU alarm ALM3[27] Registers Register Monitor error detection
Error Detection Time	up to 1 clock cycle (f_{SPB})
Product Configuration	N/A
Initialization	This safety mechanism is always active, but the alarm must be configured in the SMU by the application
Tests	Startup test: • **SM2[AoU].Register:SFF**
Limitations and/or	N/A

recommendations	

5.1.51 Illegal Op-code Detection

ID	SM1[HW].CPU.TRAP:IOPC
Description	The TriCore CPU detects Invalid opcodes due to corrupt or illegal CPU instructions and raises a trap. This mechanism is only effective for primary and secondary opcode fields of 32 and 16bit instructions; it is not effective for immediate value fields or register/memory pointer fields.
References	[2] Trap System
Error Signaling	CPU Trap IOPC - Illegal Opcode
Error Detection Time	The error is detected before the execution of an illegal or corrupt CPU instruction and the trap handler commences in the next cycle
Product Configuration	N/A
Initialization	N/A
Tests	Startup tests • **SM2[AoU].CPU.TRAP:TEST**
Limitations and/or recommendations	N/A

5.1.52 LMU Data Path Error Detection Code

ID	SM1[HW].LMU.DP:EDC
Description	This safety mechanism protects both the read, and write paths through the LMU, from the SRI interface to the LMU SRAM.
References	[3] Local Memory Unit (LMU) > Error Detection and Signaling > Internal ECC Error [3] Local Memory Unit (LMU) > Error Detection and Signaling > Internal SRAM Read Error
Error Signaling	SMU alarm ALM2[22] SRI non-CPU SRI agents: ECC Write phase error ALM2[23] SRI non-CPU SRI agents: ECC Read phase error
Error Detection Time	Up to 20 clock cycles (f_{SRI})
Product	The LMU SRAM is only available in TC27x and TC29x series.

Configuration	
Initialization	N/A
Tests	Startup tests: • **SM2[AoU].SRI:EDC**
Limitations and/or recommendations	N/A

5.2 Assumptions of Use: Software-Based Tests

5.2.1 Lockstep CPU Comparator Alarm Test

ID	SM2[AoU].CPU:LOCKSTEP.ALARM_TEST
Description	[SM_AURIX_CPU_1:1]It shall be tested that the Lockstep Comparison Logic (LCL) can activate the corresponding SMU alarm. Fault injection is supported by hardware and can be triggered by writing in the LCLTEST register. This is part of the tests for **SM1[HW].CPU:LOCKSTEP**. [/req]
References	[3] System Control Unit > System Control Unit (SCU) > Lockstep CPU Configuration > Logic Monitor Control Registers
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.2 CPU trap Test

ID	SM2[AoU].CPU.TRAP:TEST
Description	[SM_AURIX_CPU_1:2]The CPU trap system shall be tested. The test is performed by generating a trap of each trap class that is tested and checking that the correct trap handler gets invoked with the correct trap identification number.[/req]
References	[2] Trap System
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent

Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.3 Voltage Monitors Test

ID	SM2[AoU].EVR:MON, SM2[AoU].EVR33:MON, SM2[AoU].EVR13:MON, SM2[AoU].ExtVREG:MON
Description	[SM_AURIX_EVR_1:1]The safety related EVR Voltage Monitors shall be tested. Each voltage monitor can be tested with software configuring too high or too low threshold values to force under-voltage and overvoltage detection. The following voltage monitors can be tested independently: • **SM1[HW].EVR13:MON** • **SM1[HW].EVR33:MON** • **SM1[HW].ExtVREG:MON** Additionally, the result of the EVR bandgap hardware built-in self-test, **SM1[HW].BANDGAP:MON**, shall be checked. [/req]
References	**SM1[HW].EVR13:MON** **SM1[HW].EVR33:MON** **SM1[HW].ExtVREG:MON**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.4 Clock Monitor Test

ID	SM2[AoU].CLK:CLKMON
Description	[SM_AURIX_CCU_1:1]All the safety related clock monitors shall be tested. The test configures the clock dividers and the target frequency in a way that the clock monitors detect an error. [/req]
References	5.1.6 Clock Frequency Monitors 5.1.7 System PLL Frequency Monitor 5.1.9 ERAY PLL Frequency Monitor

Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	Specific conditions for testing the system PLL frequency monitor, **SM1[HW].SystemPLL:CLKMON** : • CCUCON0.CLKSEL=1 has to be configured • It is recommended to configure f_{PLL} to 100 MHz before executing this test, in order to avoid too high current jumps caused by the automatic switch to the 100 MHz backup clock when the system PLL frequency monitor detects an error (clock emergency behavior).

5.2.5 SRAM ECC Test

ID	SM2[AoU].SRAM:ECC
Description	[SM_AURIX_CPU_3:1]The ECC safety mechanisms of each safety related SRAM instance of CPU and LMU shall be tested. It shall be tested that, for the following error conditions, the corresponding alarm is raised in the SMU: • Uncorrectable error • Address buffer overflow[/req] The test consists of injecting bit failure(s) in a memory location of the considered SRAM instance, using the memory controller: using the ECC bit Mapping Mode in the ECC Safety Register, the ECC functionality can be disabled while a data is written to a memory location. Activating the ECC again and reading back the data causes an ECC error. This can be used to inject single, double or multiple bit faults.
References	**SM1[HW].SRAM:ECC** **SM1[HW].SRAM:EDC**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.6 LMU ECC Monitor Test

ID	SM2[AoU].LMU.SRAM.ECC:MONITOR
Description	[SM_AURIX_LMU_1:1]The LMU ECC Monitor safety mechanism shall be tested if the LMU SRAM is used for safety related data.[/req]
References	**SM1[HW].LMU.SRAM.ECC:MONITOR**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.7 SRAM Address Monitor Test

ID	SM2[AoU].SRAM:ADDRMON
Description	[SM_AURIX_CPU_3:2]The Address Monitor of each safety related SRAM instance of CPU and LMU shall be tested. It shall be tested that, for the following error conditions, the corresponding alarm is raised in the SMU: • Address error The address error can be tested by using a special test mode controlled by the SFLE bit in the ECC Safety Register.[/req]
References	**SM1[HW].SRAM:ADDRMON**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.8 SRAM Error Tracking Test

ID	SM2[AoU].SRAM:Monitor
Description	[SM_AURIX_CPU_3:3]If the SRAM ECC is used for monitoring of a safety related SRAM instance, the ECC Error Tracking mechanism of this instance shall

	be tested. It shall be at least tested that, when the error tracking buffer overflows, the corresponding alarm is raised in the SMU: • Address buffer overflow[/req] The test consists of injecting bit failure(s) in a memory location of the considered SRAM instance, using the memory controller: using the ECC bit Mapping Mode in the ECC Safety Register, the ECC functionality can be disabled while a data is written to a memory location. Activating the ECC again and reading back the data causes an ECC error. This can be used to inject multiple correctable errors that are logged in the ECC Error Tracking buffer.
References	**SM1[HW].SRAM:Monitor**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.9 PFLASH Error Correction Code (ECC) Test

ID	SM2[AoU].PFLASH:ECC
Description	[SM_AURIX_PMU_1:2]The error reporting capabilities of the safety mechanisms **SM1[HW].PFLASH:ECC** and **SM1[HW].PFLASH.ECC:Monitor** shall be tested[/req]
References	**SM1[HW].PFLASH:ECC** **SM1[HW].PFLASH.ECC:Monitor** [3] Program Memory Unit (PMU) > Application Hints > Testing ECC Alarms and Error Flags [3] Program Memory Unit (PMU) > Application hints > Startup Test of ECC Logic [3] Program Memory Unit (PMU) > Application Hints > General Advice for Startup Tests
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	[SM_AURIX_PMU_1:6]During the PFLASH Error Correction Code Test it is necessary to disable the ECC correction; during that time CPUs shall not execute from PFLASH.[/req]

5.2.10 PFLASH Address Error Detection Test

ID	SM2[AoU].PFLASH:ADDRMON
Description	[SM_AURIX_PMU_1:3]The error reporting capabilities of the safety mechanism **SM1[HW].PFLASH:ADDRMON** shall be tested.[/req]
References	**SM1[HW].PFLASH:ADDRMON** [3] Program Memory Unit (PMU) > Application Hints > Testing ECC Alarms and Error Flags [3] Program Memory Unit (PMU) > Application Hints > General Advice for Startup Tests
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	It is sufficient to test that the FLASH ECC can trigger ALM2[4] PMU PFLASH ECC non correctable multiple bit error alarm. ALM2[5] PMU PFLASH Addressing error, is more complex to trigger with software and is always triggered together with ALM2[4].

5.2.11 PFLASH ECC Error Detection Logic Comparator Test

ID	SM2[AoU].PFLASH.EDC:CMP
Description	[SM_AURIX_PMU_1:5] The error reporting capabilities of the safety mechanism **SM2[HW].PFLASH.EDC:CMP** shall be tested. [/req]
References	**SM2[HW].PFLASH.EDC:CMP** [3] Program Memory Unit (PMU) > Application Hints > Testing the “EDC Comparator” [3] Program Memory Unit (PMU) > Application Hints > General Advice for Startup Tests
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.12 PFLASH Error Tracking Test

ID	SM2[AoU].PFLASH:Monitor
Description	[SM_AURIX_PMU_1:7]It shall be tested that the PFLASH Error Tracking Logic, is able to track correctable single bit errors, correctable double bit errors and uncorrectable errors in the CBAB / UBAB FIFOs.[/req]
References	**SM1[HW].PFLASH:Monitor**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.13 SRI Error Detection Test

ID	SM2[AoU].SRI:EDC
Description	[SM_AURIX_SRI_1:1]The error detection capabilities of **SM1[HW].SRI.ADDR:EDC** and **SM1[HW].SRI.DATA:EDC** shall be tested with software injecting errors in the SRI communication during address and read phases. Errors can be injected from each CPU using the SRI Error Generation register SEGEN. If the LMU shared memory LMURAM is used to store safety related data, the LMU internal data path shall also be tested.[/req]
References	[3] CPU Subsystem > Safety Features **SM1[HW].SRI.ADDR:EDC** **SM1[HW].SRI.DATA:EDC**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.14 SRI Error Handler Test

ID	SM2[AoU].SRI:EH
Description	[SM_AURIX_SRI_1:3]The SRI Error Handler shall be tested by software, including:

	• Detection of incorrect access level: User Mode 0 and 1 • Detection of an access to an address that is not mapped to any SRI-device • Detection of an unsupported / invalid opcode • Detection of an invalid transaction id [/req]
References	**SM1[HW].SRI:EH** [3] CPU Subsystem > Safety Features
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.15 SPB Error Handler Test

ID	SM2[AoU].SPB:EH
Description	[SM_AURIX_SPB_1:3]The Error signaling capabilities of the SPB Error Handler shall be tested. This is covered by other tests such as **SM2[AoU].SPB:TIMEOUT**, and **SM2[AoU].Register:AP**. [/req]
References	**SM1[HW].SPB:EH**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.2.16 SPB Timeout Test

ID	SM2[AoU].SPB:TIMEOUT

Description	[SM_AURIX_SPB_1:1]The SPB Timeout safety mechanism shall be tested. This can be done by configuring SBCU_CON.TOUT, then trying to access one peripheral register, such as the SPB transaction times out and the SPB Service Request is generated.[/req]
References	**SM1[HW].SPB:Timeout**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.2.17 Register Monitor Test

ID	SM2[AoU].Register:SFF
Description	[SM_AURIX_SFF_TEST]The register monitor **SM1[HW].Register:SFF** shall be tested. The test is supported by a hardware self-test started and monitored using the registers SMU_RMCTL, SMU_RMEF,and SMU_RMSTS.[/req]
References	[3] Safety Management Unit (SMU) > SMU Special Safety Registers: Register Monitor
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.18 Register Access Protection Test

ID	SM2[AoU].Register:AP, SM2[AoU].STM:AP, SM2[AoU].DMA:AP, SM2[AoU].IR:AP, SM2[AoU].SBCU:AP, SM2[AoU].SMU:AP, SM2[AoU].FCE:AP, SM2[AoU].IOM:AP, SM2[AoU].PORT:AP, SM2[AoU].QSPI:AP, SM2[AoU].MSC:AP, SM2[AoU].CAN:AP, SM2[AoU].SENT:AP, SM2[AoU].ERAY:AP, SM2[AoU].GTM:AP, SM2[AoU].CCU6:AP, SM2[AoU].VADC:AP, SM2[AoU].DSADC:AP, SM2[AoU].MTU:AP, SM2[AoU].SCU:AP, SM2[AoU].Ethernet:AP,

	SM2[AoU].HSSL:AP, SM2[AoU].PSI5:AP, SM2[AoU].PSI5-S:AP, SM2[AoU].SRI:AP, SM2[AoU].I2C:AP, SM2[AoU].PMU:AP
Description	[SM_AURIX_SPB_2:1]The SPB Register Access Protection and SRI Register Access Protection shall be tested For each module for which it is used. This includes testing of write accesses to at least one protected register while write access is disabled, and check that the corresponding alarm is set. [/req]
References	5.1.41 SPB Register Access Protection 5.1.42 SRI Register Access Protection
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.19 CPU Memory Protection Test

ID	SM2[AoU].CPU.MPU:TEST
Description	[SM_AURIX_CPU_2:1]If it is used, the CPU memory protection unit shall be tested with the following test cases: • Read accesses to a protected memory region are blocked and a MPR trap is generated • Write accesses to a protected memory region are blocked and a MPW trap is generated • Read accesses to a memory region which are not covered by any Data Protection Range are blocked and a MPR trap is generated • Write accesses to a memory region which are not covered by any Data Protection Range are blocked and a MPW trap is generated • Read accesses to an unprotected memory region are not blocked and no trap is generated • Write accesses to an unprotected memory region are not blocked and no trap is generated • Program fetches from a protected memory region are blocked and a MPX trap is generated • Program fetches from an unprotected memory region are not blocked and no trap is generated[/req]
References	**SM1[HW].CPU.DATA:MPU** **SM1[HW].CPU.CODE:MPU**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.20 CPU Bus MPU Test

ID	SM2[AoU].CPU.BUS.MPU:TEST
Description	[SM_AURIX_CPU_2:2]If it is used, the CPU bus memory protection shall be tested. This includes testing of write accesses to one memory region while write access is disabled, and check that the corresponding SMU alarm is raised. Note: in TC29x, TC27x, TC26x each CPU has its own bus memory

	protection and the test need to be done for each one of them. [/req]
References	SM1[HW].CPU.BUS:MPU
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	Implementation hints: • When writing variables in the CPUx PSPR from CPUx itself, store accesses are issued from the CPUx Data Memory Interface (DMI) to its Program Memory Interface (PMI). These accesses go over the SRI bus checking the Bus MPU of CPUx. The normal as well as the safe task identifier - see SM1[HW].CPU:STI - can be used to test authorized and non-authorized access. • In a multicore microcontroller, one CPU can test the bus memory protection of other CPU issuing write access to either DSPR or PSPR of the other CPU

5.2.21 CPU Register Access Protection Test

ID	SM2[AoU].CPU.AP:TEST
Description	[SM_AURIX_2:3]CPU register access protection shall be tested for each CPU for which it is used. This includes testing of disabled write access to at least one CPU register, and check that the corresponding SMU alarm is raised. [/req]
References	SM1[HW].CPU:AP
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.22 AURIX Watchdogs test

ID	SM2[AoU].WDT:TEST
Description	[SM_AURIX_WDT_TEST]If one of the following features of the AURIX internal watchdogs are used by the application safety concept, it shall be tested: • Timeout monitoring • ENDINIT protection • Password protection • Automatic password sequencing[/req]
References	**SM1[HW].WDT** **SM1[HW].SWDT** **SM1[HW].WDT:CE**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.23 Reset Stable State Check

ID	SM2[AoU].RESET:SSCheck
Description	[SM_AURIX_RCU_2:1]After a reset, it shall be checked whether the CPU(s) reached a stable state before reset, meaning that all ongoing write transactions have been completed. The RSTCON2.CSSx (x=0-2) bits indicate whether each CPU successfully flushed its write buffers and reached an idle state before the previous reset. If any of the CSS bits is zero after a reset it shall be assumed that the SRAM content have been corrupted by the reset.[/req]
References	4.8 Reset Control Unit (RCU) [3] System Control Unit > Reset Control Unit (RCU) > Reset Operation
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	The RSTCON2 register, as well as the RSTSTAT register are not protected by hardware mechanisms, therefore it is recommended that the application

	software do an additional plausibility check of any information provided by these registers that can affect the execution of the safety related startup software flow.

5.2.24 SBCU Configuration Check

ID	SM2[AoU].SBCU:InitCheck
Description	[SM_AURIX_SPB_1:2]The application shall verify the correct configuration of the bus control unit (SBCU_CON, SBCU_PRIOH, and SBCU_PRIOL) after reset and SW initialization (initial setup and every subsequent configuration changes).[/req]
References	[3] On-Chip System Buses and Bus Bridges > System Bus Control Unit Registers
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.2.25 SMU Fault Signaling Protocol Test

ID	SM2[AoU].SMU.FSP:TEST
Description	[SM_AURIX_SMU_2:1]The correct operation of the Fault Signaling Protocol (FSP) on the error pin shall be tested at startup. It is assumed this test is implemented by the system integrator at system-level, as this is part of testing the complete safe state activation path.[/req]
References	[3] Safety Management Unit (SMU) > Fault Signaling Protocol (FSP)
Error Signaling	Implementation dependent
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.26 SMU Initialization Check

ID	SM2[AoU].SMU:InitCheck
Description	[SM_AURIX_SMU_3:2]The SMU register configuration shall be verified by reading back the configuration registers and comparing with the expected value after every configuration changes. The following SMU configuration registers have to be considered: • SMU_FSP • SMU_AGC • SMU_RTC • SMU_RTAC0 • SMU_RTAC1 • SMU_AG0CFx (x=0-2) • SMU_AG1CFx (x=0-2) • SMU_AG2CFx (x=0-2) • SMU_AG3CFx (x=0-2) • SMU_AG4CFx (x=0-2) • SMU_AG5CFx (x=0-2) • SMU_AG6CFx (x=0-2) • SMU_AG0FSP • SMU_AG1FSP • SMU_AG2FSP • SMU_AG3FSP • SMU_AG4FSP • SMU_AG5FSP • SMU_AG6FSP • SMU_RMCTL • SMU_PCTL.PCS, SMU_PCTL.HWEN, SMU_PCTL.HWDIR Additionally it is recommended to verify the reset configuration register in SCU by reading back the following registers and comparing with the expected value after every configuration changes: • ARSTDIS • RSTCON [/req]
References	[3] Safety Management Unit (SMU)
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.27 SMU Configuration Lock Test

ID	SM2[AoU].SMU.LOCK:TEST
Description	[SM_AURIX_SMU_3:3]The application shall check the SMU configuration lock mechanism after performing the SMU lock sequence: 1. After locking the SMU configuration, attempt to over-write one SMU alarm configuration register 2. Check that the configuration register was not changed[/req]
References	**SM1[HW].SMU:LOCK**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	This test must be executed after the SMU configuration lock by the application. The SMU must be in the RUN state.

5.2.28 SMU Alarms Test

ID	SM2[AoU].SMU.ALARM:TEST
Description	[SM_AURIX_SMU_3:4]All SMU alarms and pre-alarms which are relevant for the application shall be tested: it shall be tested that the safety mechanism can trigger the alarm and that the SMU alarm is correctly indicated in the SMU_AGx register. [/req]
References	[3] Safety Management Unit (SMU)
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.29 SMU Test

ID	SM2[AoU].SMU:TEST

Description	[SM_AURIX_SMU_TEST_1]The system integrator shall test the SMU alarm handling and triggering of the possible reactions: • Interrupt Request • NMI • Reset • Idle • Port Emergency Stop Only the reactions that are used by the application need to be tested. This can be done by using the SMU software alarms, configured to react as expected by the test. For NMI, Reset, Idle and Port Emergency Stop, the request is processed by the System Control Unit (SCU). It is important that the correct processing of the request is also tested as part of the SMU test - for example that NMI goes all the way from SMU to the CPU.[/req]
References	[3] Safety Management Unit (SMU)
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	Additionally it is recommended to

5.2.30 SMU Recovery Timer Test

ID	SM2[AoU].SMU.RT:TEST
Description	[SM_AURIX_SMU_TEST_2]If the SMU recovery timers are used by the application, they shall be tested. [/req] A recovery timer alarm can be caused by software: 1. trigger a software alarm previously configured to trigger and interrupt and to start RTx (x=0-1) 2. the corresponding ISR does not service RTx RTx will timeout and generate ALM2[29] (x=0), or ALM2[30] (x=1)
References	[3] Safety Management Unit (SMU)
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A

Limitations and/or recommendations	N/A

5.2.31 Non-Lockstep CPU MPU Initialization Check

ID	SM2[AoU].CPU.MPU:InitCheck
Description	[SM_AURIX_CPU_MPU_INIT_1]To detect permanent failures in the CPU MPU configuration registers of non-lockstep CPUs, the system integrator shall implement a software level safety mechanism verifying the correct CPU MPU configuration (for code and data memory protection), after initial software initialization and every subsequent configuration change. The safety mechanism shall consist of reading the relevant configuration registers and comparing the read value to the expected value.[/req]
References	**SM1[HW].CPU.DATA:MPU** **SM1[HW].CPU.CODE:MPU**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	This is only necessary for Non-Lockstep CPU

5.2.32 Non-Lockstep CPU BUS MPU Initialization Check

ID	SM2[AoU].CPU.BUS.MPU:InitCheck
Description	[SM_AURIX_CPU_MPU_INIT_2] To detect permanent failures in the CPU Bus Memory Protection configuration registers of non-lockstep CPUs, the system integrator shall implement a software level safety mechanism verifying the correct CPU Bus Memory Protection configuration, after initial software initialization and every subsequent configuration change. The safety mechanism shall consist of reading the relevant configuration registers and comparing the read value to the expected value.[/req]
References	**SM1[HW].CPU.BUS:MPU**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent

Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	This is only necessary for Non-Lockstep CPU

5.2.33 LMU BUS MPU Initialization Check

ID	SM2[AoU].LMU.MPU:InitCheck
Description	[SM_AURIX_LMU_MPU_INIT_1]To detect permanent failures in the configuration registers of the memory protection of the LMU, the system integrator shall implement a software level safety mechanism verifying the correct Memory Protection configuration, after initial software initialization and every subsequent configuration change. The safety mechanism shall consist of reading the relevant configuration registers and comparing the read value to the expected value.[/req]
References	**SM1[HW].LMU.BUS:MPU**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.34 Watchdog Timer Initialization Check

ID	SM2[AoU].WDT:InitCheck
Description	[SM_AURIX_WDT_TEST:1]To detect permanent failures in the configuration registers of the watchdog Timers, the system integrator shall implement a software level safety mechanism verifying the correct watchdog configuration, after initial software initialization and every subsequent configuration change. This include at least following registers: • WDTxCON0.REL • WDTxCON0.PW

	• WDTxCON1.IR0,IR1 • WDTxCON1.DR,UR • WDTxCON1.TCTR • WDTxCON1.PAR • WDTxCON1.TCR [/req]
References	**SM1[HW].WDT** **SM1[HW].SWDT**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.35 Interrupt Router Error Detection Code Test

ID	SM2[AoU].IR:EDC
Description	[SM_AURIX_IR_1:1]If the EDC mechanism of the Interrupt Router, **SM1[HW].IR:EDC**, is used, it shall be tested by the system integrator that it can trigger an alarm in case of corrupted data in the Service Request Node (SRN). For a given SRN, a fault is injected by writing the ECC field of the corresponding Service Request Control Register (SRC) with an 8-bit or 16-bit write access. Rationale: to test the detection and reporting capabilities of the safety mechanism. [/req]
References	[3] Interrupt Router (IR) > Service Request Node (SRN) > Service Request Control Register (SRC)
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.36 DMA Timestamp Test

ID	SM2[AoU].DMA:TIMESTAMP
Description	[SM_AURIX_DMA_TEST_1]If the DMA timestamp mechanism is used, it shall be tested by the system integrator. One way to test is to execute successive transactions and check the timestamp is counting up. [/req]
References	**SM1[HW].DMA:TIMESTAMP**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.37 Flexible CRC Engine (FCE) Test

ID	SM2[AoU].FCE:TEST
Description	[SM_AURIX_FCE_TEST_1]The CRC engines of the Flexible CRC Engine (FCE) which are used by the application shall be tested by the system integrator. This can be done by computing a CRC over known data and check the result with the expected CRC stored in Flash. Alternatively, for the CRC32 engine, the same data could be processed by the CPU CRC32 instruction or DMA CRC32 mechanism and compared with the result of the FCE CRC engine.[/req] [SM_AURIX_FCE_TEST_2]If the automatic checksum check at the end of a message is used, it shall also be tested that the FCE interrupt is generated in case of wrong CRC result.[/req]
References	**SM1[HW].FCE:CRC32** **SM1[HW].FCE:CRC16** **SM1[HW].FCE:CRC8**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent

Tests	N/A
Limitations and/or recommendations	N/A

5.2.38 DMA Cyclic Redundancy Check Test

ID	SM2[AoU].DMA.DATA:CRC32, SM2[AoU].DMA.ADDR:CRC32
Description	[SM_AURIX_DMA_TEST_2]If the DMA CRC mechanism for data and addresses is used, it shall be tested by the system integrator. This can be done by computing a CRC over known data and check the result with the expected CRC stored in Flash. Alternatively, the same data and addresses could be processed by the CPU CRC32 instruction or FCE CRC32 engine and compared with the result of the DMA CRC mechanism. [/req]
References	**SM1[HW].DMA.DATA:CRC32** **SM1[HW].DMA.ADDR:CRC32**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.39 LMU Bus MPU Test

ID	SM2[AoU].LMU.MPU:TEST
Description	[SM_AURIX_LMU_MPU_TEST_1]If used to protect safety related data, the Bus Memory Protection of the LMU shall be tested by the system integrator. This can be done by issuing Load/Store accesses from CPUx to the LMU. This goes over the SRI bus, checking the Bus Memory Protection of the LMU. The normal as well as the safety task identifiers, **SM1[HW].CPU:STI**, can be used to test authorized and non-authorized accesses.[/req]
References	[3] Local Memory Unit (LMU) > Memory Protection **SM1[HW].LMU.BUS:MPU**

Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.40 LMU Register Access Protection Test

ID	SM2[AoU].LMU.AP:TEST
Description	[SM_AURIX_LMU_AP_TEST_1]If used for safety related reasons, the access enable protection of the LMU shall be tested by the system integrator. This can be done by writing LMU registers from a CPU. The normal as well as the safety task, **SM1[HW].CPU:STI**, can be used to test authorized and non-authorized accesses.[/req]
References	**SM1[HW].LMU:AP**
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.41 IOM test

ID	SM2[AoU].IOM:TEST
Description	[SM_AURIX_IOM_TEST_1]If used to monitor safety related signals, the IOM shall be tested by the system integrator. The test shall check that,when the IOM is configured and a fault is injected in IOM Logic Analyzer Module (LAM), the corresponding SMU alarm is raised.[/req]
References	4.26.9.2 Testing of the IOM

Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.42 Peripheral SRAM ECC Test

ID	SM2[AoU].PSRAM:ECC
Description	[SM_AURIX_PSRAM_1]If the application safety concept relies on the SRAM ECC safety mechanisms for SRAM instances in peripheral modules, the ECC safety mechanisms of each of these instances shall be tested. It shall be at least tested that, for the following error conditions, the corresponding alarm is raised in the SMU: • Uncorrectable error • Address buffer overflow[/req] The test consists of injecting bit failure(s) in a memory location of the considered SRAM instance, using the memory controller: using the ECC bit Mapping Mode in the ECC Safety Register, the ECC functionality can be disabled while a data is written to a memory location. Activating the ECC again and reading back the data causes an ECC error. This can be used to inject single, double or multiple bit faults.
References	**SM1[HW].SRAM:ECC** **SM1[HW].SRAM:EDC** [3]Memory Test Unit (MTU) > ECC Safety Register
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.43 SRAM Address Monitor Test

ID	SM2[AoU].PSRAM:ADDRMON
Description	[SM_AURIX_PSRAM_1:3]If the SRAM Address Monitor is used for monitoring of a safety related peripheral SRAM instance, the Address Monitor safety mechanisms of this instance shall be tested. It shall be at least tested that, for the following error conditions, the corresponding alarm is raised in the SMU: • Address error The address error can be tested by using a special test mode controlled by the SFLE bit in the ECC Safety Register.[/req]
References	**SM1[HW].SRAM:ADDRMON** [3]Memory Test Unit (MTU) > ECC Safety Register
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.44 Peripheral SRAM Error Tracking Test

ID	SM2[AoU].PSRAM:Monitor
Description	[SM_AURIX_PSRAM_1:2]If the SRAM ECC is used for monitoring of a safety related peripheral SRAM instance, the ECC Error Tracking mechanism of this instance shall be tested. It shall be tested that, when the error tracking buffer overflows, the corresponding alarm is raised in the SMU: • Address buffer overflow[/req] The test consists of injecting bit failure(s) in a memory location of the considered SRAM instance, using the memory controller: using the ECC bit Mapping Mode in the ECC Safety Register, the ECC functionality can be disabled while a data is written to a memory location. Activating the ECC again and reading back the data causes an ECC error. This can be used to inject multiple correctable errors that are logged in the ECC Error Tracking buffer.
References	[3] Memory Test Unit (MTU) > ECC Safety Register **SM1[HW].SRAM:Monitor**
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent

Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.2.45 EVR Configuration Check

ID	SM1[AoU].EVR:CFGMON
Description	[SM_AURIX_EVR_4:1]Random hardware faults while latching the hardware configuration pins during supply ramp-up have the potential to corrupt the EVR configuration. For that purpose a software test of the EVR intended configuration shall be executed at startup: • It needs to be checked that the latched hardware configuration in the PMSWSTAT.HWCFG register matches the intended Hardware configuration in the application • it needs to be checked that the current EVR mode and state as reflected in the EVRSTAT.EVR13, EVRSTAT.EVR33, EVRSTAT.EXTPASS13, EVRSTAT.EXTPASS33 registers match the intended EVR modes in the application Additionally, a permanent overvoltage on VEXT, VDDP3, or VDD, already present before the SMU is configured, will not trigger the corresponding alarm once the SMU is configured. Therefore it shall be checked by software, at startup, that the following registers are not set: EVRSTAT.OV13, EVRSTAT.OV33, EVRSTAT.OVSWD.[/req]
References	[3] System Control Unit > Power Supply and Control 5.1.4 EVR Voltage Monitor
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.3 Assumptions of Use: Software-Based Runtime Monitoring

5.3.1 Non-Lockstep CPU Soft Error Monitoring

ID	SM1[AoU].CPU:SoftErrorMon
Description	[SM_AURIX_NLCPU_2:1]On non-lockstep CPUs, transient faults such as soft errors shall be controlled by system level measures. Such measures include, for example, software plausibility checks and trap handling.[/req]
References	N/A
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	This is only necessary for safety related application software executed on a non-lockstep CPU. [SM_AURIX_NLCPU_2:2]As usage of a non-lockstep CPU is usually only considered for safety related software supporting safety goals up to ASIL B, full redundant software execution might not be necessary to reach the hardware metric targets. This shall be analyzed by the system integrator.[/req]

5.3.2 Non-Lockstep CPU Software Based Self-Test

ID	SM1[AoU].CPU:SBST
Description	[SM_AURIX_NLCPU_1]If safety related software is executed on non-Lockstep CPUs, the software-based self-Test (SBST) provided by Infineon for Tricore CPUs shall be executed cyclically, at least once in the diagnostic test interval. The test is non-destructive and is designed to preserve the state of the CPU; therefore it can be used cyclically at runtime. This is a measure against permanent faults, to support the single point fault metric.[/req]
References	4.5 Non-Lockstep CPU 3.9.2 SafeTlib
Error Signaling	Application software error handler
Test Execution Time	Implementation dependent

Product Configuration	The AURIX family or microcontroller provides devices with different configurations regarding the availability of lockstep cores. See [3] CPU Subsystem > AURIX Family CPU Configuration.
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	The SBST covers only the core functionalities of the CPU. The exact scope of the SBST is provided in the corresponding device FMEDA. The SBST cannot provide coverage against transient faults

5.3.3 SPB Test Pattern

ID	SM1[AoU].SPB:TEST
Description	[SM_AURIX_SPB_3:1]The application shall implement a cyclic test of the SPB to detect possible permanent faults related to address, data, and control signal lines of the SPB.[/req] Safety Mechanisms implemented to monitor the correct behavior of safety related peripherals also provide a high coverage of the SPB. This is, for example, End-to-End safe communication protocol and redundant, diverse I/O or communication channels. For safety related configuration registers, it may be necessary to check the content after write - this is application dependent. Note that SPB may have to be considered as a possible source of common cause failure in the dependent failures analysis at the system level. Common cause failures due to the SPB are fairly covered by data diversity in case of redundant channels (for example redundant analog acquisitions with data diversity) Alternatively, when no other mechanism is available, cyclically reading and checking static registers of safety related peripherals such as MODID, ACCEN0 and ACCEN1 provides a low coverage for SPB failures.
References	none
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.3.4 System Timer Plausibility Check

ID	SM1[AoU].STM:Plausibility
Description	[SM_AURIX_STM_3:1]The System Timer (STM) is not monitored by hardware. Failures of the STM include wrong or no interrupt generation and wrong content for the STM counter value. These failures shall be controlled by the system integrator with system level safety measures. Depending on the STM usage, timeout monitoring (5.3.5) and/or program flow monitoring (5.3.6) of the software executing on a specific CPU could be sufficient. Alternatively, a plausibility check using an independent timer, such as another STM or the CPU Temporal Protection System could be implemented in software.[/req]
References	N/A
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	N/A
Limitations and/or recommendations	N/A

5.3.5 Timeout Monitoring

ID	SM1[AoU].WDT:TIMEOUT
Description	[SM_AURIX_WDT_1:1]Hardware fault of the microcontroller as well as software faults which delay the execution of a program may endanger the overall system timing. To prevent this, a temporal protection system shall be implemented.[/req]
References	N/A
Error Signaling	Implementation dependent
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or	N/A

recommendations	

5.3.6 Program Flow Monitoring

ID	SM1[AoU].WDT:PFM
Description	[SM_AURIX_WDT_1:2]Hardware fault of the microcontroller as well as software faults may corrupt the sequence and timing of the individual program sections. To prevent this, Program Flow Monitoring (logical and temporal monitoring), shall be implemented.[/req]
References	N/A
Error Signaling	Implementation dependent
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.3.7 Monitoring with External Watchdog

ID	SM1[AoU].ExtWDT
Description	Besides environmental conditions that would cause the AURIX to fail completely or partially without being able to signal an error, the external window watchdog can also detect internal failure of the microcontroller leading to severely delayed or stopped execution of the complete microcontroller such as failures of the power management controller.
References	Section 3.8.3
Error Signaling	Implementation dependent
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent

Limitations and/or recommendations	N/A

5.3.8 Interrupt Rate Monitor

ID	SM1[AoU].IR:Monitor
Description	[SM_AURIX_IR_3:1]The system integrator shall implement software-level safety mechanisms to detect omission or unexpected triggering of service requests, based on known application constraints and expected interrupt rates. Additionally it is recommended to verify by software the configuration information written to Service Request Nodes (SRCx [15:0]) after initial software initialization and every subsequent configuration changes to detect any fault in the write path.[/req]
References	[3] Interrupt Router (IR) > Interrupt Router SRC Registers
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	

5.3.9 Interrupt Source Check

ID	SM1[AoU].IR:SrcCheck
Description	[SM_AURIX_IR_3:2]Interrupt plausibility check, to detect unexpected triggering of service requests or call of a wrong ISR, based on additional information from the interrupt source. For example, the ISR shall check additional status information from the interrupt source (for example in peripheral module) and abort execution, if the interrupt service request cannot be confirmed. [/req]
References	[3] Interrupt Router (IR) > Interrupt Router SRC Registers [3] System Control Units > NMI Trap Generation
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product	N/A

Configuration	
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	• It is recommended to forward all interrupts not used in the interrupt vector table to a generic handler managing these unintended event. • It is recommended that the Service Request Nodes (SRN) of the interrupt router which are not used by the application are kept disabled - this is the default state after reset

5.3.10 DMA Monitor

ID	SM1[AoU].DMA:Monitor
Description	[SM_AURIX_DMA_3:1]The application software shall verify the integrity of data transferred by DMA. The functional failure modes to consider for the DMA are listed below. There are many possible usage of the DMA and this is the responsibility of the application to cover the failures of the DMA; the detection method listed below informs about typical system level measures as well as specific AURIX features which may be used by the application for this purpose. **Incorrect message at destination** caused by • The DMA internal data path corrupting the data, or • The DMA selecting a wrong destination address, or • The DMA selecting a wrong source address. **DMA transfer without a request** caused by • Unintended transfer caused by DMA, or • Unintended transfer caused by the unintended triggering of the corresponding interrupt in the interrupt router. **Requested message not transferred (or transfer notification missing)** caused by • The DMA not executing the transfer, or • The DMA not triggering the channel transfer interrupt, or • Interrupt router not processing the channel transfer interrupt correctly **Message loss** caused by • Delayed DMA transfer: new transaction request for the channel while a request was already pending for this channel **Message corruption** caused by • Delayed DMA transfer: message gets overwritten at source after the transfer started but before the transfer finishes[/req]
References	N/A
Error Signaling	Implementation dependent
Error Detection	Implementation dependent

Time	
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	**Exemplary safety measures (informative only)**: • **Data corruption** could be detected with safety mechanisms in software such as ○ end-to-end protection of the message including CRC over the data, or ○ redundant messages. • **Wrong destination address** could be detected by the combination of ○ bus memory protection (5.1.35, 5.1.36), and ○ other measures such as software safety mechanisms against data corruption, or checking the DMA automatically generated address CRC (5.1.31). • **Wrong source address** could be detected with safety mechanisms in software such as ○ end-to-end protection of the message, or ○ redundant messages, or ○ checking the DMA automatically generated address CRC (5.1.31). • **DMA transfer without a request** could be detected with safety mechanisms in software such as ○ end-to-end protection of the message including message counter, or ○ interrupt source check per software in the DMA channel interrupt service routine. See 5.3.9. • **Requested message not transferred (or transfer notification missing)** could be detected with safety mechanisms in software checking periodically the correct timing properties of the application execution. • **Message loss** caused by delayed DMA transfer could be detected by safety mechanisms in software such as ○ end-to-end protection of the message including message counter, or ○ checking the DMA transaction request lost error notification, or ○ checking periodically the correct timing properties of the application execution. • **Message corruption** caused by delayed DMA transfer could be detected by safety mechanisms in software such as ○ end-to-end protection of the message including CRC over the complete message, or ○ redundant messages, or ○ checking periodically the correct timing properties of the application execution.

5.3.11 End-to-End Communication Protection

ID	SM1[AoU].CAN:SafeProtocol, SM1[AoU].ERAY:SafeProtocol, SM1[AoU].ETH:SafeProtocol, SM1[AoU].HSSL:SafeProtocol
Description	[SM_AURIX_SAFE_COMM_1]The application shall implement a safe communication protocol on CAN, E-Ray, Ethernet, HSSL[/req]

References	4.1.3 Safe Communication [3] Controller Area Network Controller (MultiCAN+) > CAN Basics [3] FlexRay™ Protocol Controller (E-Ray) > Functional Description [3] High Speed Serial Link (HSSL)
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	The CAN, ERay and Ethernet modules implement an internal loop-back mode which can be used to in-system test of the module without access to the external bus

5.3.12 QSPI Protocol Protection

ID	SM1[AoU].QSPI:SafeProtocol
Description	[SM_AURIX_QSPI_3:1]A system level safety mechanism shall be implemented to monitor communication protocols such as SPI. The QSPI module of AURIX shall be seen as black boxes. It is recommended to use an End-to-End safe communication protocol. [/req]
References	None
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	The QSPI module implements an internal loop-back mode which can be used to in-system test of the module without access to the external bus

5.3.13 Other Communication Protocols Protection

ID	SM1[AoU].ASCLIN:SafeProtocol, SM1[AoU].SENT:SafeProtocol, SM1[AoU].PSI5:SafeProtocol
Description	A system level safety mechanism shall be implemented to monitor communication protocols such as SENT, ASCLIN and PSI5. The SENT, ASCLIN and PSI5 modules of AURIX shall be seen as black boxes. It is recommended to use an End-to-End safe communication protocol.
References	None
Error Signaling	Application software error handler
Error Detection	Implementation dependent

Time	
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.3.14 LMU SRAM Data Path Test

ID	SM1[AoU].LMU.DP:TEST
Description	[SM_AURIX_LMU_SRAM_1]If necessary to reach the safety metric targets, a safety mechanism covering permanent faults of the LMU SRAM data path shall be implemented in software by the system integrator and executed cyclically, at least once every diagnostic test interval. The test consists of the following sequence: 1. Write 8 different 8-bit values to sequential addresses in LMU SRAM starting with a 64-bit aligned address. Data pattern (64-bit): 0x1122334455667788 2. Perform a 64-bit read and compare against expected value. 3. Write 4 different 16-bit values to sequential addresses in LMU SRAM starting with a 64-bit aligned address. Data pattern (64-bit): 0xEEDDCCBBAA998877 4. Perform a 64-bit read and compare against expected value 5. Write 2 different 32-bit values to sequential addresses in LMU SRAM starting with a 64-bit aligned address. Data pattern (64-bit): 0xA5A5A5A55A5A5A5A 6. Perform a 64-bit read and compare against expected value [/req]
References	[3] Local Memory Unit (LMU) [7] Device FMEDA
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

5.3.15 LMU SRAM Data Interference

ID	SM1[AoU].LMU:Interference

Description	[SM_AURIX_LMU_SRAM_2]In order to prevent other neighboring IPs to interfere with the LMU SRAM the system integrator shall implement a software safety mechanism which consist of the following sequence: 1. Clear Watchdog EndInit 2. Write the value 0x00000001 in at the address 0xF8700900 3. Read back 0xF8700900 and check that read value is equal to 0x00000002 4. Set Watchdog EndInit **[/req]**
References	[3] Local Memory Unit (LMU) [7] Device FMEDA
Error Signaling	Application software error handler
Error Detection Time	Implementation dependent
Product Configuration	N/A
Initialization	Implementation dependent
Tests	Implementation dependent
Limitations and/or recommendations	N/A

6 Error Reaction and Fault Tolerance Considerations

The main focus of the safety mechanisms available in the AURIX microcontroller is to signal the presence of an error. Although the reaction to an error is application dependent, this section gives hints on which kind of reaction may be adapted and how to possibly recover from an error in the microcontroller's vital parts.

***Attention:** It is assumed the system integrator understands the microcontroller's failure mechanisms and, ultimately, it is the system integrator responsibility to define and implement an error reaction strategy adapted to the considered application.*

6.1 Generic Error Reaction

The safety mechanisms by themselves are in general not able to distinguish whether the error was caused by a transient or permanent fault. In specific cases it therefore can make sense to attempt an error recovery, for example by initiating a system reset. The SMU can be configured to request a system reset when an alarm is activated. Alternatively, the SMU can be configured to trigger a CPU interrupt or an NMI, giving control to the software which can request a system reset after executing some error handling.

In the same time, the Fails Safe Protocol (FSP) could be activated to inform the system about an error in the AURIX. After the system reset, the application software should determine whether the fault is permanent or not. If the reset was caused by a transient fault, the software can transition the FSP back to the fault free state.

6.2 Correctable SRAM errors

All safety related instances within AURIX are covered by an error correcting code, which is able to either correct single bit errors or detects double bit errors in one word. Single correctable error events are logged by the SRAM Error Tracking mechanism **SM1[HW].SRAM:Monitor** which can log up to five different addresses of faulty words, depending on the size of the SRAM instance, and trigger a SMU alarm in case of address buffer overflow.

Due to this mechanism, single correctable error events can be ignored to increase the fault tolerance of the system without impacting the safety.

[SM_AURIX_26]In order to comply with the reliability targets of the microcontroller the number of permanent cell errors in the SRAM that are correctable with dedicated Error Correction Codes techniques is assumed to be under certain limits. Therefore the application shall monitor the SRAMS as follow:

- For each SRAM used in a safety application, correctable error events shall be logged in the error tracking buffer, **SM1[HW].SRAM:Monitor**, and the corresponding alarm shall be ignored.
- The application software shall react on overflow of the error tracking buffer, and consider that the microcontroller is not operational anymore. The error tracking buffer overflow event is indicated through an SMU alarm.
- For SRAM instances not used as part of a safety application, single-error events shall not be logged.[/req]

6.3 Correctable PFLASH errors

For PFLASH, the AURIX implements an enhanced error correcting code, correcting single and double bit errors and detecting triple bit errors. This safety mechanism is referenced in the document as **SM1[HW].PFLASH:ECC**. Moreover the PMU implements error tracking buffers where the addresses of PFLASH wordlines which caused a correctable error are logged in Correctable Bit Address Buffers (CBABs), this mechanism is called the PFLASH monitor and is referenced in this document as **SM1[HW].PFLASH:Monitor**.

[SM_AURIX_27]In order to comply with the reliability targets of the microcontroller the number of permanent cell errors in the PLASH that are correctable with dedicated Error Correction Codes techniques is assumed to be under certain limits. Therefore the application shall monitor the PFLASH as follows:

- For each PFLASH bank used in a safety application, single-bit error events shall not be logged, only double-bit error events shall be logged in the CBAB. If the number of double-bit error events is strictly greater than two within a PFLASH bank, the application software shall react and consider that the microcontroller is not operational anymore. [/req]
- The application shall react on each non-correctable error event

Note: [SM_AURIX_28]For non-safety related PFLASH banks, the application shall ignore all corrected single and double bit errors [/req]

After each PORST or after a system/application reset requested because of a PFLASH related error, it is recommended to compute a CRC over the PFLASH content; besides checking the integrity of the PFLASH content, doing so will trigger events for all the permanent double bit errors which can be counted by application software using the PFLASH monitor. This can be realized, for example, with one of the CRC32 hardware accelerators listed in section 4.27.

Note: [SM_AURIX_29]The CBABs generate an alarm when they are full; this is when 10 entries are logged. To realize the above requirements, the filling level of the CBAB shall be monitored by the application software.[/req]

Note: The CBABs are not cleared after a system reset; if the PFLASH Monitor is used at startup as described in the previous paragraph, they should be cleared before executing the CRC test.

Note: By default after PORST, no error are logged in the CBAB; if the PFLASH Monitor is used at startup as described in this paragraph, it need to be configured by the application before executing the CRC test.

7 Dependent Failures Considerations

7.1 Introduction

Stress conditions have the potential to affect the intended operation of the microcontroller SEooC and lead to coincident errors through coupling mechanisms. Coincident errors that affect safety related hardware elements that implement independent safety requirements have the potential to violate the safety goals at system level. According to the ISO26262-9 several root causes can lead to dependent failures. Software systematic failures are not in the scope of this document. The following information is informative only and intended to support the dependent failure analysis of the system integrator.

The root causes (also called stress situations) that need to be considered can be classified into:

- Internal Stress:
  - Ageing
  - Internal disturbance caused by excessive activity leading to voltage drops, increased temperature
- Environmental Factors:
  - Electromagnetic interference
  - Electrical overstress
  - Temperature cycling
  - Soldering stress
  - Thermo mechanical stress
  - Radiation
  - External Voltage noise (ripple, overvoltage,...)
- Random Hardware failures from shared hardware elements:
  - Test structures
  - Clock structures
  - Reset structures
  - Supply
  - Shared logic
  - Shared storage elements
  - Shared configuration registers
  - Random Hardware failures related to the interfaces (wiring) of the given redundant hardware elements

The result from a stress condition is a multiple fault occurrence that affects several given hardware elements assumed to be independent. The following effects are usually considered:

- Latchup
- Timing Fault
- Bit Flip
- Crosstalk between signals
- Cross-influence between signals (aggressor victim model)

Each fault occurrence may have different behavior in the time domain. The following time behaviors are considered:

- The fault may be permanent until a power off-on cycle takes place, for example in the case of latchup
- The fault manifests depending on a precise workload situation
- The fault is workload-independent but depends on the characteristics of the stress/disturbance: example a timing fault depends on the duration and intensity of a voltage spike

ISO26262:9 lists the following root causes which are beyond the scope of this document and may be addressed in other documents:

- Development faults
  - development faults are addressed by the design process
- Manufacturing faults
  - manufacturing faults are addressed by the design process and the production tests
- Installation and repair faults
  - must be considered by the system integrator at system level

Typical measures to mitigate dependent failures include:

- physical separation of logical elements: this is achieved by defining layout regions during the placement of the hardware,
- functional diversity: using diverse modules as redundant channels, for example GTM and CCU6, or ADC and digital communication interface, highly reduces the risks that a failure causes both elements to fail at the same time and with the same effect,
- signal diversity: when using homogenous hardware redundancy, for example using two VADC channels as redundant channels, the acquired signals should have some sort of diversity that makes it possible to detect dependent failures of the redundant hardware elements. Examples of techniques for correlating analog sensor signals, including diversity measures such as opposite slopes, different gains, and out-of-range regions are provided in ISO26262-5:2011 Annex D.

7.2 Dependent Failure Initiators applicable to Basic Device Operation

There are faults, such as faults of the core voltages or of the clock system, which have with a global effect on the AURIX and can affect both mission and monitoring logic and/or prevent the device from reporting alarms to the system. These dependent failure initiators applicable to basic device operation are listed in **Table 17**.

Table 17 Dependent Failure Initiators applicable to Basic Device Operation

ID	Dependent Failure Initiator
DFI_REF_ID_6: supply	Power supply elements, including power distribution network, common voltage regulators, common references (bandgaps, bias generators and related network)
DFI_REF_ID_3: clock structures	Common clocks (including PLL, clock trees, clock enable signals, etc.)
DFI_REF_ID_4: test structures	Common test logic including signal scan chains: test structure supporting production tests, controlled by the Test Control Unit (TCU), Common debug logic including debug routing network and

ID	Dependent Failure Initiator
	trace signals (OCDS, MCDS)
DFI_REF_ID_7: reset structures	Common reset logic including reset signals, controlled by the Reset Control Unit (RCU)

7.3 Dependent Failure Initiators applicable to Redundant Peripheral Scenarios

There are potential dependent failure initiators which are applicable to any application scenario in which two (or more) redundant peripheral modules or sub-modules are used to provide data for software comparison in the processor. These dependent failure initiators are listed in **Table 18**.

Table 18 Dependent Failure Initiators applicable to Redundant Peripheral Scenarios

ID	Dependent Failure Initiator
RDP00: Pad Fault	Pad or Package stuck-at, short or open affecting both S1 and S2 paths
RDP01: QM Interference with PORTS	Interference from co-existent QM software reconfigures pad/port settings
RDP02: SPB Address Fault	Single bit stuck-at, short or open in SPB address causing S1 and S2 result register aliasing
RDP03: SPB Data Fault	Single bit stuck-at, short or open fault in SPB datapath or slave select mux causing similar data error in both S1 and S2 results
RDP04: SPB Interface Fault	Fault in SPB Bus Master/arbiter or comparison memory slave etc affecting result reads from both S1 and S2 modules
RDP04a: SPB Clock Fault	Fault in SPB clock affecting result reads from both S1 and S2 modules
RDP05: SPB Bus Hang	Stuck-at, short or open fault with major impact on SPB protocol preventing any result reads
RDP06: CPU Execution Fault	Stuck-at, short or open fault within CPU causing wrong computation and leading to incorrect result comparison between S1 and S2 results
RDP07: CPU Crash	Stuck-at, short or open fault causing major CPU crash preventing comparison of S1 and S2 reults
RDP08: Application Service Request Fault	Stuck-at, short or open causing missed application Service Request
RDP09: QM Interference with Application Service Request Routing	Interference from co-existent QM software changes routing of interrupts/DMA requests
RDP10: Common Module Fault	Other internal Common Cause functional fault affecting both S1 and S2 channels
RDP10a: Register Fault	Fault in common control register with impact on both S1 and S2, or blocked write affecting independent control register writes to both S1 and S2 register sets

Attention: The EMC related common cause failures are not covered by the Application Dependent Scenario description VADC-VADC. Indeed, EMC can disturb redundant

analog measurements and have the same impact as RDP10; dedicated counter-measures on system level have to be taken into account by the integrator.

Correct EMC behavior can only be measured within target system of the system integrator. Therefor no EMC tests are performed by Infineon for this analog scenario. In general Infineon performs EMI immunity tests based on recognized industrial standards (IEC 62132) to show correct functionality of the on-chip hardware safety mechanisms. No hardware safety mechanism is used in the VADC/VADC application scenario.

To ensure correct functioning of the VADC-VADC scenario in the system integrator environment, the integrator can take counter measures on system level:

1. Test and validation according to applicable standards

2. The effect of EMI may be controlled/detected by measures such as data diversity and input filtering

7.4 Recommendations against Dependent Failures in Package

Guidelines for the selection of signal balls/pins for redundant input/output channels to mitigate dependent failures in the package are given, for all package variants, in a separate document named “AP32318 AURIX^{TM} Recommendations against Common Cause Failures in package”

8 Hardware Metrics

To support the estimation of the hardware architectural metrics, Infineon provides an FMEDA tool [7], adapted to each device of the AURIX family. When using the FMEDA, please take the following information into account:

[SM_AURIX_23]The system integrator is responsible for the implementation and the diagnostic coverage of the safety mechanisms identified as assumptions of use in the FMEDA (SM1[AoU]* and SM2[AoU]*). The diagnostic coverage indicated in the FMEDA for the assumptions of use, if any, are based on the informal estimations from [1] ISO26262-5 Annex D “Evaluation of the Diagnostic Coverage”, and shall be adapted by the system integrator.[/req]

The safety measures indicated for the application dependent parts, if any, are examples or placeholders only and are not formally identified in this manual.

Some safety hardware safety mechanisms have different diagnostic coverage for different failure modes. This is indicated in FMEDA by appending the tag [FMC=…] to the safety mechanism. For example SM1[HW].SRAM:ECC [FMC=MBE] is used with the SRAM ECC coverage for multiple bit errors. In this safety manual, only the safety mechanism is documented, not the different failure mode coverages, therefore the reader only finds SM1[HW].SRAM:ECC.

[SM_AURIX_31]System integrators using bare die products shall consider the alpha activity of the material set for encapsulation and interconnect, and scale the soft error failure rate given by AURIX FMEDA [7] accordingly if necessary. Rationale: the AURIX FMEDA considers the alpha activity caused by the mold compound of AURIX packages which is specified to be less than 10 counts per hour per square meter (10 cts/(hour.m^2)). [/req] w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG