UM1840
User manual

STM32F4 Series safety manual

Introduction

This document describes how to use the microcontrollers of the STM32F4 Series in the context of a safety-related system, specifying the user's responsibilities for installation and operation in order to reach the targeted safety integrity level.

This manual applies to the microcontrollers of the STM32F4 Series and to the X-CUBE-STL part number.

System designers can avoid going into the details of the functional safety standards application to the STM32F4 Series by following the indications reported in this manual.

This manual is written in compliance with IEC 61508. It indicates how to use the microcontrollers of the STM32F4 Series in the context of other functional safety standards such as safety of machinery directives in ISO 13849.

The safety analysis summarized in this manual takes into account the variation in terms of memory size, internal peripheral number and package among the different part numbers of the Arm® Cortex®-M4 based STM32F4 Series microcontrollers.

This manual has to be read along with the technical documentation for the related part numbers (such as reference manuals and datasheets) available on *www.st.com*.

Contents

List of tables

List of figures

1 About this document

1.1 Purpose and scope

This document describes how to use the Arm®[a] Cortex®-M4 based STM32F4 Series in the context of a safety-related system, specifying the user's responsibilities for installation and operation, in order to reach the desired safety integrity level.

This document is useful to system designers willing evaluate the safety of their solution embedding one or more STM32F4 Series microcontroller(s).

1.2 Terms and abbreviations

Abbreviations related to STM32F4 Series hardware modules (like DMA, GPIO etc.) are the same than the ones used in STM32F4 Series technical documentation. See *Table 1* below for a list of acronyms used in this document.

Table 1. Terms and abbreviations

Acronym	Definition
CCF	Common cause failure
CM	Continuous mode
COTS	Commercial off-the-shelf
CoU	Conditions of use
CPU	Central processing unit
CRC	Cyclic redundancy check
DC	Diagnostic coverage
DMA	Direct memory access
DTI	Diagnostic test interval
ECM	Engine control module
ECU	Electronic control unit
EUC	Equipment under control
FIT	Failure in time
FMEA	Failure mode effect analysis
FMEDA	Failure mode effect diagnostic analysis
HD	High demand
HFT	Hardware fault tolerance
HW	Hardware

---

[a] Arm is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Table 1. Terms and abbreviations (continued)

Acronym	Definition
ITRS	International technology roadmap for semiconductors
LD	Low demand
MCU	Microcontroller unit
MTBF	Mean time between failure
MTTFd	Mean time to failure
NA	Not available
PDS(SR)	Power drive system (safety related)
PEc	Programmable electronics - core
PEd	Programmable electronics - diagnostic
PFH	Probability of failure per hour
PL	Performance level
PST	Process safety time
SFF	Safe failure fraction
SIL	Safety integrity level
SRCF	Safety-related control function
SRECS	Safety-related electrical control systems
SRP/CS	Safety-related parts of control systems
SW	Software

Read also the following definitions used within this manual:

- End user: the STM32F4 Series final user of that is in charge of integrating the MCU in a real application (for example an electronic control board).
- Application software: the actual software running on the STM32F4 Series MCUs and implementing the safety function.

1.3 Reference normative

This document is written in compliance with the IEC 61508 international norm for functional safety of electrical, electronic and programmable electronic safety-related systems.

The version used as reference is IEC 61508:1-7 © IEC:2010.

The other functional safety standards considered in this manual are the following:

- ISO 26262-1, 2, 3, 4, 5, 6, 7, 8, 9: 2011(E), ISO 26262-10: 2012(E),
- ISO 13849-1:2006, ISO 13849-2:2010,
- IEC 62061:2005/AMD1:2012
- IEC 61800-5-2:2007, ed.1.0,

Table 2 reports the mapping of this document content with respect to the requirements listed in the IEC 61508-2 Annex D.

Table 2. Mapping between this document content and IEC 61508-2 Annex D requirements

IEC 61508 requirement (part 2 annex D)	Reference
D2.1 a) a functional specification of the functions capable of being performed	*Section 3*
D2.1 b) identification of the hardware and/or software configuration of the compliant item	*Section 3.2*
D2.1 c) constraints on the use of the compliant item or assumptions on which analysis of the behavior or failure rates of the item are based	*Section 3.2*
D2.2 a) the failure modes of the compliant item due to random hardware failures, that result in a failure of the function and that are not detected by diagnostics internal to the compliant item;	*Section 3.7*
D2.2 b) for every failure mode in a), an estimated failure rate;	
D2.2 c) the failure modes of the compliant item due to random hardware failures, that result in a failure of the function and that are detected by diagnostics internal to the compliant item;	
D2.2 d) the failure modes of the diagnostics, internal to the compliant item due to random hardware failures, that result in a failure of the diagnostics to detect failures of the function;	
D2.2 e) for every failure mode in c) and d), the estimated failure rate;	
D2.2 f) for every failure mode in c) that is detected by diagnostics internal to the compliant item, the diagnostic test interval;	*Section 3.2.2*
D2.2 g) for every failure mode in c) the outputs of the compliant item initiated by the internal diagnostics;	*Section 3.6*
D2.2 h) any periodic proof test and/or maintenance requirements;	*Section 3.7*
D2.2 i) for those failure modes, in respect of a specified function, that are capable of being detected by external diagnostics, sufficient information must be provided to facilitate the development of an external diagnostics capability.	
D2.2 j) the hardware fault tolerance;	*Section 3*
D2.2 k) the classification as type A or type B of that part of the compliant item that provides the function (see 7.4.4.1.2 and 7.4.4.1.3);	

The safe failure fraction reported in this manual has been computed under the assumptions described in this document and especially according to the conditions of use described in *Section 3.7: Conditions of use*.

2 STM32F4 Series microcontrollers development process

The development process of a microelectronic device that is used in safety critical application takes into account the adequate management to reduce the probability of systematic faults introduced during the design phase.

IEC 61508:2 in Annex F (Techniques and measures for ASICs - avoidance of systematic failures) act as a guidance in tailoring the microcontroller standard design and manufacturer process to the compliance of the IEC 61508 requirements. The checklist reported in the named Annex F helps to collect all related evidences of a given real process.

2.1 STMicroelectronics standard development process

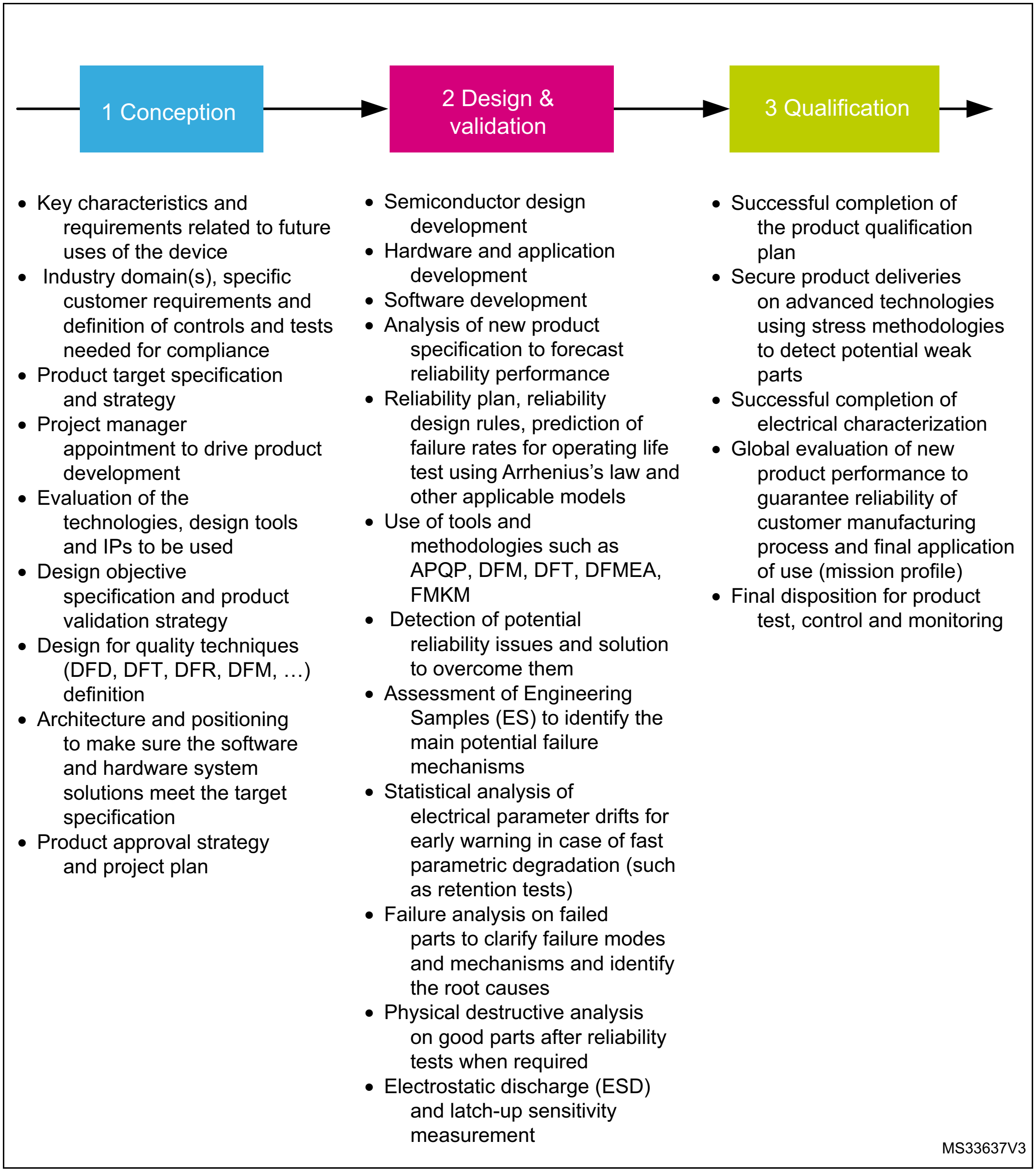
STMicroelectronics (ST) serves four industry domains:

- Standard products.
- Automotive products: ST automotive products are AEC-Q100 compliant. They are subject to specific stress testing and processing instructions in order to achieve the required quality levels and product stability.
- Automotive safety: a subset of the automotive domain. ST uses as a reference the ISO 26262 Road vehicles Functional safety standard. ST supports customer inquiries regarding product failure rates and FMEDA to support hardware system compliance to established safety goals. ST provides products that are safe in their intended use, working in cooperation with customers to understand the mission profile, adopt common methods and define countermeasures for residual risks.
- Medical products: ST complies with applicable regulations for medical products and applies due diligence in the development and validation of these products.

STMicroelectronics product development process, compliant with the ISO/TS 16949 standard, is a set of interrelated activities dedicated to transform customer specification and market or industry domain requirements into a semiconductor device and all its associated elements (package, module, sub-system, application, hardware, software and documentation), qualified respecting ST internal procedures and able to be manufactured using ST internal or subcontracted technologies.

Figure 1 presents a summary of the STMicroelectronics product-development process.

Figure 1. STMicroelectronics product development process



- Key characteristics and requirements related to future uses of the device
- Industry domain(s), specific customer requirements and definition of controls and tests needed for compliance
- Product target specification and strategy
- Project manager appointment to drive product development
- Evaluation of the technologies, design tools and IPs to be used
- Design objective specification and product validation strategy
- Design for quality techniques (DFD, DFT, DFR, DFM, …) definition
- Architecture and positioning to make sure the software and hardware system solutions meet the target specification
- Product approval strategy and project plan

- Semiconductor design development
- Hardware and application development
- Software development
- Analysis of new product specification to forecast reliability performance
- Reliability plan, reliability design rules, prediction of failure rates for operating life test using Arrhenius’s law and other applicable models
- Use of tools and methodologies such as APQP, DFM, DFT, DFMEA, FMKM
- Detection of potential reliability issues and solution to overcome them
- Assessment of Engineering Samples (ES) to identify the main potential failure mechanisms
- Statistical analysis of electrical parameter drifts for early warning in case of fast parametric degradation (such as retention tests)
- Failure analysis on failed parts to clarify failure modes and mechanisms and identify the root causes
- Physical destructive analysis on good parts after reliability tests when required
- Electrostatic discharge (ESD) and latch-up sensitivity measurement

- Successful completion of the product qualification plan
- Secure product deliveries on advanced technologies using stress methodologies to detect potential weak parts
- Successful completion of electrical characterization
- Global evaluation of new product performance to guarantee reliability of customer manufacturing process and final application of use (mission profile)
- Final disposition for product test, control and monitoring

3 Reference safety architecture

This section reports the details of the STM32F4 Series safety architecture.

3.1 Introduction

The STM32F4 Series microcontroller analyzed in this document can be used as a compliant item within different safety applications.

The aim of this section is to identify such compliant item and therefore to define the context of the analysis in terms of assumptions with respect to a reference concept definition. This concept definition includes therefore reference safety requirements as also assumptions on the design external to the defined compliant item.

As a consequence of compliant item approach, the goal is not to provide an exhaustive hazard and risk analysis of the system around the microcontroller, but rather to list the system-related information considered during the analysis. Such information include - among others - application related assumptions for dangerousness factors, frequency of failures and diagnostic coverage already guaranteed by the application.

3.2 Compliant item

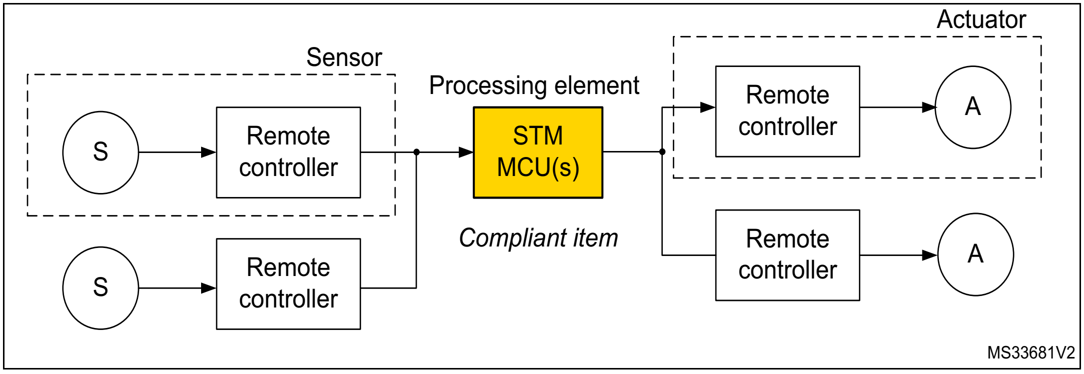
This section includes all the information related to the definition of the compliant item, including its usage in different safety architecture schemes.

3.2.1 Definition of the compliant item

According to IEC 61508:1 clause 8.2.12, a compliant item is any item (for example an element) on which a claim is being made with respect to the clauses of IEC 61508 series. With respect to its user, at the end of its development the compliant item must be described by a safety manual.

In this document, the compliant item is defined as a system including one or two STM32 microcontrollers (MCU) (see *Figure 2*). The communication bus is directly or indirectly connected to sensors and actuators.

Figure 2. Definition of the compliant item



Other components might be related to the compliant item, like the external HW components needed to guarantee either the functionality of the STM32F4 Series (external memory, clock quartz etc) or its safety (for example the external watchdog, voltage supervisors).

Defined compliant item can be classified as “element” according IEC61508-4, 3.4.5.

3.2.2 Safety functions performed by the compliant item

In essence, the compliant item architecture can be represented as composed by the following processes performing the safety function or part of it:

- Input processing elements (PEi) reading safety related data from the remote controller connected to the sensor(s) and transferring them to the following computation elements;
- Computation processing elements (PEc) performing the algorithm required by the safety function and transferring the results to the following output elements;
- Output processing elements (PEo) transferring safety related data to the remote controller connected to the actuator;
- In the case of the 1oo2 architecture, a further voting processing element (PEv) can be present;
- Processes external to the compliant item are considered to guarantee safety integrity, such as a watchdog (WDTe) and voltage monitors (VMONe).

The role of the PEv and of the external processes WDTe and VMONe is clarified in the sections where the CoU (definition of safety mechanism) are detailed:

- WDTe: refer to Independent watchdog – VSUP_SM_2, Control flow monitoring in application software – CPU_SM_1,
- VMONe: refer to Supply Voltage Monitoring – VSUP_SM_1.

In summary, STM32F4 Series microcontrollers support the implementation of end user safety functions composed by three operations:

- Safe acquisition of safety related data from input peripheral(s).
- Safe execution of application software program and safe computation of related data.
- Safe transfer of results or decisions to output peripheral(s).

Claims on the compliant item and computation of safety metrics are done with respect to these three basic operations.

According to above reported definition for implemented safety functions, the compliant item i.e. the element can be regarded as type B (as per IEC61508-2, 7.4.4.1.2 definition). Despite accurate, exhaustive and detailed failure analysis has been done for STM32F4 Series, this device has to be considered intrinsically complex and therefore type B classification is appropriate.

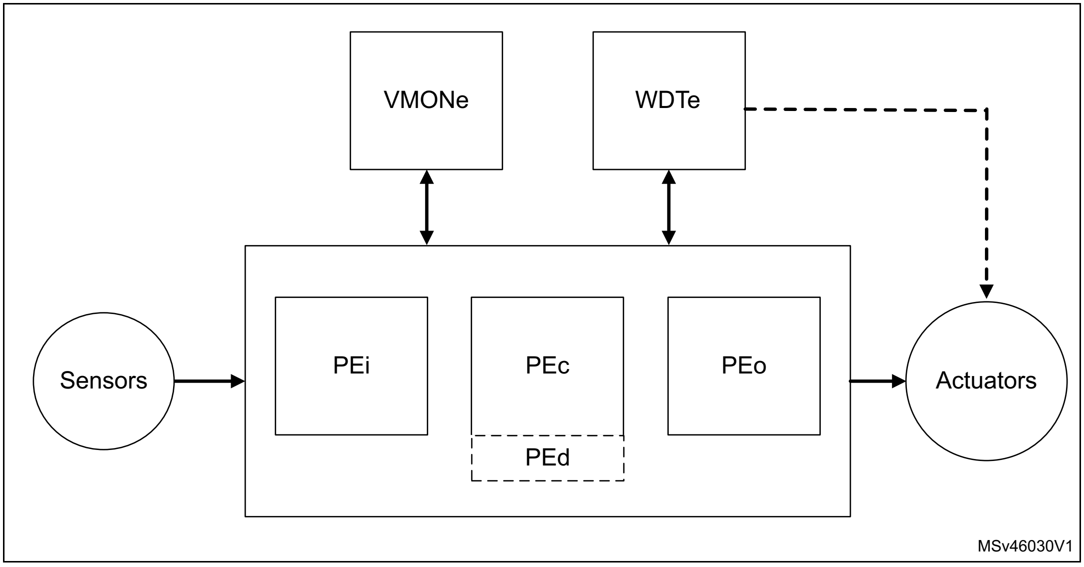
Two main safety architecture are therefore identified: 1oo1 (using one MCU) and 1oo2 (using two MCUs).

3.2.3 Reference safety architectures - 1oo1

In 1oo1 reference architecture (shown in below *Figure 3*) the safety integrity of the compliant item is guaranteed by the combination of STM32F4 Series internal processes (implemented safety mechanisms) and external processes WDTe and VMONe.

Target for 1oo1 reference architecture is SIL2.

Figure 3. 1oo1 reference architecture

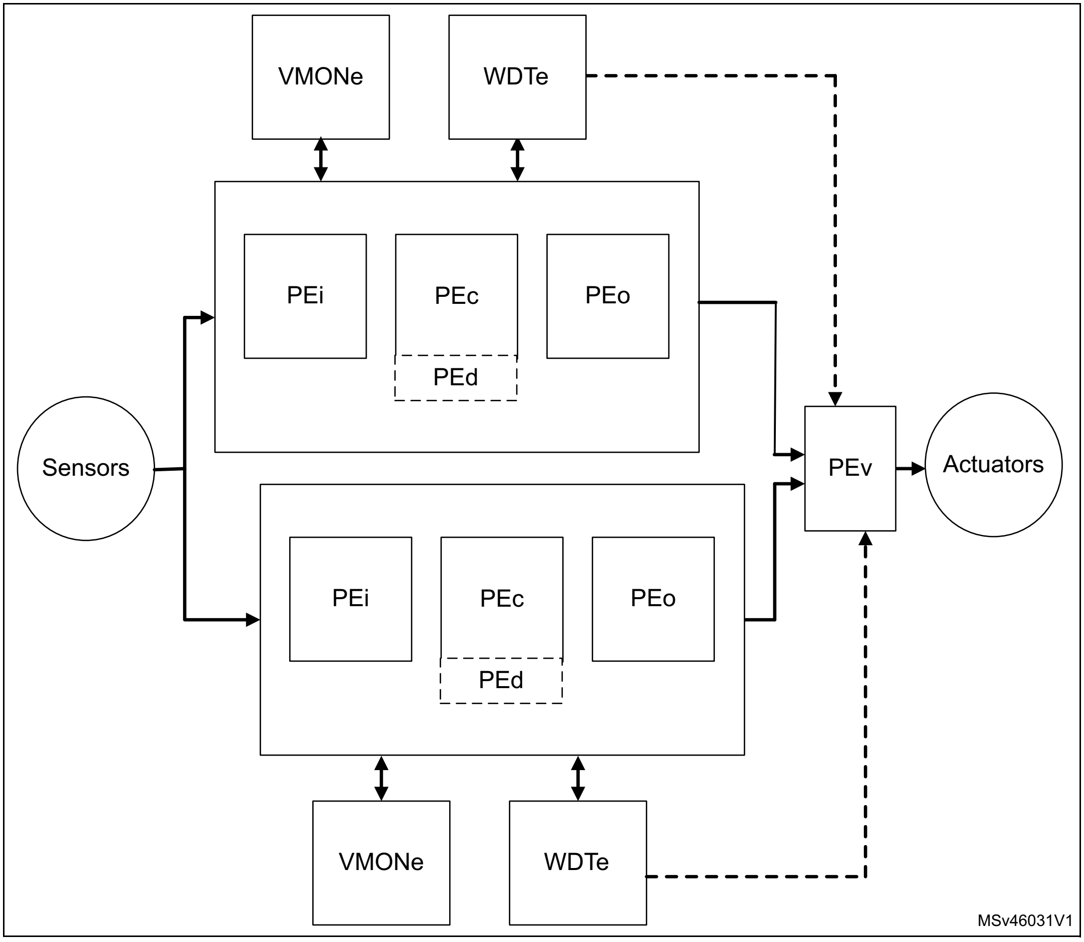


3.2.4 Reference safety architectures - 1oo2

1oo2 reference architecture (shown in below *Figure 4*) is composed by two separate channels, each of them implemented in the same way of 1oo1 reference architecture. Safety integrity of each channel is guaranteed by the combination of STM32F4 Series internal processes (implemented safety mechanisms) and external processes WDTe and VMONe. Safety integrity of overall compliant item is guaranteed by the external voter PEv allowing to claim HFT=1. Achievement of higher safety integrity levels as per IEC61508-2 Table 3 is therefore possible. Appropriate separation between the two channels (including power supply separation) should be implemented in order to avoid huge impact of common-cause failures (refer to *Section 4.2*). βD computation is anyway required.

Target for 1oo2 reference architecture is SIL3.

Figure 4. 1oo2 reference architecture



3.3 Assumed requirements

This section collects all assumptions done during the safety analysis of STM32F4 Series microcontrollers

3.3.1 Assumed safety requirements

The concept specification, the hazard and risk analysis, the overall safety requirement specification and the consequent allocation has determined the requirements for the compliant item (ASR: assumed safety requirements) listed here below.

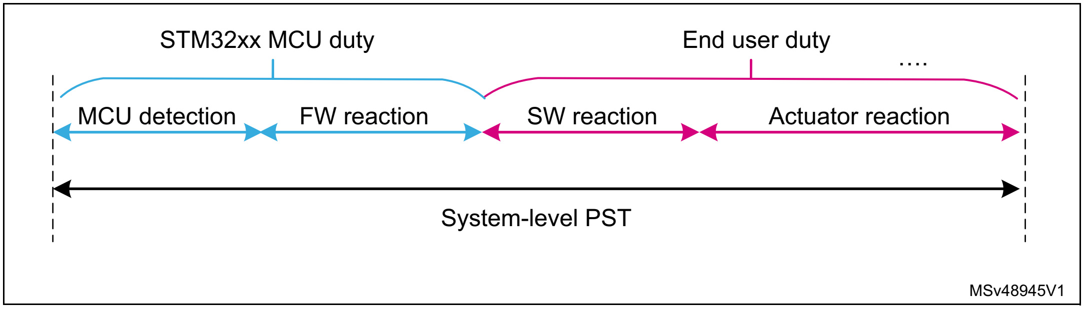
Caution: It is the end user’s responsibility to check the compliance of the final application with these assumptions.

ASR1: The compliant item can be used for four kinds of safety functions mode of operations according part 4, 3.5.16:

- A continuous mode or high-demand SIL3 safety function (CM3), or
- A low-demand SIL3 safety function (LD3), or
- A continuous mode or high-demand SIL2 safety function (CM2), or
- A low-demand SIL2 safety function (LD2).

ASR2: The compliant item is used to implement a safety function allowing a time budget of 10 ms (worst case) for the STM32 MCU to detect and react to a failure. That time corresponds to the portion of the Process Safety Time allocated to STM32F4 Series MCUs ("STM32xx MCU duty" in *Figure 5*) in error reaction chain at system level.

Figure 5. Allocation and target for STM32 PST



ASR3: The compliant item is used in a safety function that can be continuously powered-on for a time higher than 8 hours. It is assumed to not require any proof test and the lifetime of the product is considered to be not less than 10 years.

ASR4: It is assumed that only one safety function is performed or if many, all functions are classified with the same SIL and therefore they are not distinguishable in terms of their safety requirements.

ASR5: In case of multiple safety functions implementations, it is assumed that end user is responsible to guarantee their needed mutual independence.

ASR6: It is assumed that there are no "non-safety related" functions implemented in application software and coexisting with the safety functions.

ASR7: It is assumed that the implemented safety function(s) is not depending on STM32F4 MCU transition to and from a low-power state.

ASR8: The local safe state of the compliant item is the one in which either:

- SS1: the application software is informed by the presence of a fault and a reaction by the application software itself is possible
- SS2: the application software cannot be informed by the presence of a fault or the application software is not able to execute a reaction(a)

Details on safe states SS1 and SS2 are provided in following table:

---

a. The end user must take into account that random hardware failures affecting the STM32 can compromise the MCU capability of operating properly (for example failure modes affecting the program counter prevent the correct execution of software).

Table 3. SS1 and SS2 safe state details

Safe state	Condition	Compliant item action	System Transition to Safe state – 1oo1 architecture	System Transition to Safe state – 1oo2 architecture
SS1	The application software is informed by the presence of a fault and a reaction by the application software itself is possible.	Fault reporting to application software	Application software drives the overall system in his safe state	Application software in one of the two channels drives the overall system in his safe state
SS2	The application software cannot be informed by the presence of a fault or the application software is not able to execute a reaction.	Reset signal issued by WDTe	WDTe drives the overall system in his safe state (“safe shut-down”)(1)	PEv drives the overall system in his safe state

1. Safe state achievement intended here is compliant to Note on IEC61508-2, 7.4.8.1 a)

ASR9: It is assumed that the safe state defined at system level by the end user is compatible with the assumed local safe state (SS1, SS2) for the compliant item.

ASR10: The compliant item is assumed to be analyzed according to routes 1H and 1S of IEC 61508-2.[a]

ASR11: The compliant item is assumed to be regarded as type B as per IEC61508:2, 7.4.4.1.2.

ASR12: It is assumed that dual-Flash banks mass erase and reprogramming features are used during final system’s maintenance state.

3.4 Electrical specifications and environment limits

The user must not exceed the electrical specification and the environmental limits defined in the below list as reported in the STM32F4 Series user manual in order to guarantee the STM32F4 Series safety integrity:

- Absolute maximum rating,
- Capacity,
- Operating conditions.

Due to the large number of STM32F4 Series part numbers, the related user manuals and datasheets are not listed in this document; users are responsible to carefully check the above reported limits in the technical documentation on the related part number available on *www.st.com.*

---

[a] Please refer to *Section 3.5: Systematic safety integrity* and *Section 3.6: Description of hardware and software diagnostics*.

3.5 Systematic safety integrity

According to the requirements of IEC 61508 -2, 7.4.2.2, the Route 1S has been considered in the STM32F4 Series development. As clearly authorized by IEC61508-2, 7.4.6.1, STM32 MCU series can be considered a standard, mass-produced electronic integrated device – for which stringent development procedures, rigorous testing and extensive experience of use minimizes the likelihood of design faults. Anyway, an internal assessment against the compliance of STM32 MCU development flow with the techniques and measures suggested in IEC 61508-2 Annex F has been executed. The Safety Case Database (*Section 5: List of evidences*) maintains the evidences of the compliance to the norm.

3.6 Description of hardware and software diagnostics

This section lists all the safety mechanisms (hardware, software and application level) considered in the safety analysis of the microcontrollers of the STM32F4 Series. It is expected that users are familiar with the STM32F4 Series architecture, and that this document is used in conjunction with the related device datasheet, user manual and reference information. Therefore, to avoid the eventuality of mistakes and reduce the amount of information to be shown, minimum functional details are included in this document. In following descriptions the words “safety mechanism”, “method” or “requirement” are used as synonym.

Note that each part number of the STM32F4 Series owns different combinations of peripherals (for instance, some of them are not equipped with USB peripheral). To reduce the number of documents and avoid information-less repetitions, the current Safety Manual (and therefore this section) addresses the overall possible peripherals available in the targeted part numbers. Users have to select which peripherals are really available on their devices, and discard the meaningless recommendations accordingly.

The implementation guidelines reported in the following section are for reference only. The safety verification executed by ST during STM32F4 Series safety analysis and related diagnostic coverage figures reported in this manual (or its Annexes) are based on such guidelines. For the sake of clarity, safety mechanism are grouped for MCU basic functions.

Information are organized in form of tables (one for each safety mechanism). *Table 4* below presents the explanation of each field:

Table 4. Safety mechanism field explanation

SM CODE	Unique safety mechanism code/identifier used also in FMEA document. Identifiers use the scheme mmm_SM_x where mmm is a 3 or 4 letter module acronym, and “x” is an incremental number. Please note that module acronym and numbering could be not sequential and/or different from module's actual name being derived by legacy documents.
Description	Short mnemonic description
Ownership	ST : means that method is available on silicon End user: method must be implemented by the end user by application software modification, hardware solutions, or both.
Detailed implementation	Detailed implementation sometimes including notes about the safety concept behind the introduction of the safety mechanism.
Error reporting	Describes how the fault detection is reported to application software
Fault detection time	Time that the safety mechanism needs to detect the hardware failure

Table 4. Safety mechanism field explanation (continued)

Addressed fault model	Reports fault model(s) addressed by the diagnostic (Permanent, Transient, or both), and other information: If ranked for Fault avoidance: method contributes to lower the probability of occurrence of a failure If ranked for Systematic: method is conceived to mitigate systematic errors (bugs) in application software design
Dependency on MCU configuration	Reports if safety mechanism implementation or characteristics change among different part numbers belonging to STM32F4 Series
Initialization	Specific operation to be executed to activate the contribution of the safety mechanism
Periodicity	Continuous : safety mechanism is active in continuous mode Periodic: safety mechanism is executed periodically. Note that safety mechanism can be accounted for diagnostic coverage contribution only if it is executed at least one per PST On Demand: safety mechanism is activated in correspondence of a specified event (for instance, reception of a data message) Startup: safety mechanism is supposed to be executed only at power-up or during off-line maintenance periods
Test for the diagnostic	Reports specific procedure (if any and recommended) to allow on-line tests of safety mechanism efficiency
Multiple faults protection	Reports the safety mechanism(s) associated in order to correctly manage a multi-fault scenario (refer to *Section 4.1.3: Notes on multiple faults scenario*)
Recommendations and known limitations	Additional recommendations or limitations (if any) not reported in other fields

3.6.1 Arm® Cortex®-M4 CPU

Table 5. CPU_SM_0

SM CODE	CPU_SM_0
Description	Periodical core self-test software for Arm® Cortex®-M4 CPU
Ownership	End user or ST
Detailed implementation	The software test is built around well-known techniques already addressed by IEC 61508:7, A.3.2 (Self-test by software: walking bit one-channel). To reach the required values of coverage, the self-test software is specified by means of a detailed analysis of all the CPU failure modes and related failure modes distribution
Error reporting	Depending on implementation
Fault detection time	Depending on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	None
Periodicity	Periodic
Test for the diagnostic	Self-diagnostic capabilities can be embedded in the software, according the test implementation design strategy chosen. The adoption of checksum protection on results variables and defensive programming are recommended

Table 5. CPU_SM_0 (continued)

Multiple faults protection	CPU_SM_5 : external watchdog
Recommendations and known limitations	This method is the main asset in STM32F4 Series safety concept. CPU integrity is a key factor, given that the major part of defined diagnostics for MCU peripherals are software-based

Table 6. CPU_SM_1

SM CODE	CPU_SM_1
Description	Control flow monitoring in application software
Ownership	End user
Detailed implementation	A significant part of the failure distribution of CPU core for permanent faults is related to failure modes directly related to program counter loss of control or hang-up. Due to their intrinsic nature, such failure modes are not addressed by a standard software test method like SM_CPU_0. Therefore it is necessary to implement a run-time control of the application software flow, in order to monitor and detect deviation from the expected behavior due to such faults. Linking this mechanism to watchdog firing assures that severe loss of control (or, in the worst case, a program counter hang-up) is detected. The guidelines for the implementation of the method are the following: – The different internal states of the application software is well documented and described (the use of a dynamic state transition graph is encouraged). – The monitoring of the correctness of each transition between different states of the application software is implemented. – The transition through all expected states during the normal application software program loop is checked. – The function in charge of triggering the system watchdog is implemented in order to constrain the triggering (preventing the issue of CPU reset by watchdog) also to the correct execution of the above-described method for program flow monitoring. – The use of the window feature of the independent watchdog (IWDG) (or an external one) helps to implement a more robust control flow mechanism fed by a different clock source. *Note: in any case safety metrics do not depend on the kind of watchdog in use (the adoption of independent or external watchdog contributes to the mitigation of dependent failures, see Section 4.2.2: Clock)*
Error reporting	Depends on implementation
Fault detection time	Depends on implementation. Higher value is fixed by watchdog timeout interval.
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	NA
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	-

Table 7. CPU_SM_2

SM CODE	CPU_SM_2
Description	Double computation in application software
Ownership	End user
Detailed implementation	A timing redundancy for safety-related computation is considered to detect transient faults affecting the Arm® Cortex®-M4 CPU subparts devoted to mathematical computations and data access. The guidelines for the implementation of the method are the following: – The requirement needs be applied only to safety-relevant computation, which in case of wrong result could interfere with the system safety functions. Such computation must be therefore carefully identified in the original application software source code – Both mathematical operation and comparison are intended as computation. – The redundant computation for mathematical computation is implemented by using copies of the original data for second computation, and by using an equivalent formula if possible
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	End user is responsible to carefully avoid that the intervention of optimization features of the used compiler removes timing redundancies introduced according to this condition of use

Table 8. CPU_SM_3

SM CODE	CPU_SM_3
Description	Arm® Cortex®-M4 HardFault exceptions
Ownership	ST
Detailed implementation	HardFault exception raise is an intrinsic safety mechanism implemented in Arm® Cortex®-M4 core, mainly devoted to intercept systematic faults due to software limitations or error in software design (causing for example execution of undefined operations, unaligned address access). This safety mechanism is also able to detect hardware random faults inside the CPU bringing to such described abnormal operations
Error reporting	High-priority interrupt event
Fault detection time	Depending on implementation, refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	None

Table 8. CPU_SM_3 (continued)

Periodicity	Continuous
Test for the diagnostic	It is possible to write a test procedure to verify the generation of the HardFault exception; anyway, given the expected minor contribution in terms of hardware random-failure detection, such implementation is not recommended.
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	None

Table 9. CPU_SM_4

SM CODE	CPU_SM_4
Description	Stack hardening for application software
Ownership	End user
Detailed implementation	The stack hardening method is required to address faults (mainly transient) affecting CPU register bank. This method is based on source code modification, introducing information redundancy in register-passed information to called functions. The guidelines for the implementation of the method are the following: – To pass also a redundant copy of the passed parameters values (possibly inverted) and to execute a coherence check in the function. – To pass also a redundant copy of the passed pointers and to execute a coherence check in the function. – For parameters that are not protected by redundancy, to implement defensive programming techniques (plausibility check of passed values). For example enumerated fields are to be checked for consistency.
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	This method partially overlaps with defensive programming techniques required by IEC61508 for software development. Therefore in presence of application software qualified for safety integrity greater or equal to SC2, optimizations are possible

Table 10. CPU_SM_5

SM CODE	CPU_SM_5
Description	External watchdog
Ownership	End user

Table 10. CPU_SM_5 (continued)

Detailed implementation	Using an external watchdog linked to control flow monitoring method (refer to CPU_SM_1) addresses failure mode of program counter or control structures of CPU. External watchdog can be designed to be able to generate the combination of signals needed on the final system to achieve the safe state. It is recommended to carefully check the assumed requirements about system safe state reported in *Section 3.3.1*. It also contributes to dramatically reduce potential common cause failures, because the external watchdog is clocked and supplied independently from the STM32F4 Series
Error reporting	Depends on implementation
Fault detection time	Depends on implementation (watchdog timeout interval)
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	To be defined at system level (outside the scope of compliant item analysis)
Multiple faults protection	CPU_SM_1: control flow monitoring in application software
Recommendations and known limitations	In case of usage of windowed watchdog, end user must consider possible tolerance in application software execution, to avoid false error reports (affecting system availability).

Table 11. CPU_SM_6

SM CODE	CPU_SM_6
Description	Independent watchdog
Ownership	ST
Detailed implementation	Using the IDWG watchdog linked to control flow monitoring method (refer to CPU_SM_1) addresses failure mode of program counter or control structures of CPU.
Error reporting	Reset signal generation
Fault detection time	Depends on implementation (watchdog timeout interval)
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	IWDG activation. It is recommended to use the “Hardware watchdog” in Option byte settings (IWDG is automatically enabled after reset)
Periodicity	Continuous
Test for the diagnostic	WDG_SM_1: Software test for watchdog at startup
Multiple faults protection	CPU_SM_1: control flow monitoring in application software WDG_SM_0: periodical read-back of configuration registers
Recommendations and known limitations	The IWDG intervention is able to achieve a potentially “incomplete” local safe state because it can only guarantee that CPU is reset. No guarantee that application software can be still executed to generate combinations of output signals that might be needed by the external system to achieve the final safe state. If this limitation turn out in a blocking point, end user must adopt CPU_SM_5

Table 12. CPU_SM_7

SM CODE	CPU_SM_7
Description	MPU - Memory protection Unit
Ownership	ST
Detailed implementation	The CPU Memory Protection Unit is able to detect illegal access to protected memory areas, according to End user programmed criteria
Error reporting	Exception raise (MemManage)
Fault detection time	Refer to functional documentation
Addressed fault model	Systematic (software errors) Permanent and Transient (only program counter and memory access failures)
Dependency on MCU configuration	None
Initialization	MPU registers shall be programmed at start-up
Periodicity	On line
Test for the diagnostic	Not needed
Multiple faults protection	MPU_SM_0: Periodical read-back of configuration registers
Recommendations and known limitations	The use of memory partitioning and protection by MPU functions is highly recommended when multiple safety functions are implemented in application software. The MPU can be indeed used to – enforce privilege rules – separate processes – enforce access rules Hardware random-failure detection capability for MPU is restricted to well-selected failure modes, mainly affecting program counter and memory access CPU functions. The associated diagnostic coverage is therefore expected to be not relevant in the framework of STM32F4 Series safety concept

Table 13. MPU_SM_0

SM CODE	MPU_SM_0
Description	Periodical read-back of MPU configuration registers
Ownership	End user
Detailed implementation	This method must be applied to MPU configuration registers (also unused by the end user application software). Detailed information on the implementation of this method can be found in *Section 3.6.5: EXTI controller*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0

Table 13. MPU_SM_0 (continued)

Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

3.6.2 Embedded FLASH memory

Table 14. FLASH_SM_0

SM CODE	FLASH_SM_0
Description	Periodical software test for Flash memory
Ownership	End user or ST
Detailed implementation	Permanent faults affecting the system Flash memory, memory cells and address decoder, are addressed through a dedicated software test that checks the memory cell contents versus the expected value, using signature-based techniques. According to IEC 61508:2 Table A.5, the effective diagnostic coverage of such techniques depends on the width of the signature in relation to the block length of the information to be protected - therefore the signature computation method is to be carefully selected. Note that the simple signature method (IEC 61508:7 - A.4.2 Modified checksum) is inadequate as it only achieves a low value of coverage. The information block does not need to be addressed with this test as it is not used during normal operation (no data nor program fetch)
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	Flash size changes according part number
Initialization	Memory signatures must be stored in Flash as well
Periodicity	Periodic
Test for the diagnostic	Self-diagnostic capabilities can be embedded in the software, according the test implementation design strategy chosen
Multiple faults protection	CPU_SM_1: control flow monitoring in application software CPU_SM_0: periodical core self-test software
Recommendations and known limitations	This test is expected to have a relevant time duration – test integration must therefore consider the impact on application software execution. The use of internal CRC module is recommended. In principle DMA feature for data transfer can be used. *Note: Unused Flash sections can be excluded from testing*

Table 15. FLASH_SM_1

SM CODE	FLASH_SM_1
Description	Control flow monitoring in application software
Ownership	End user

Table 15. FLASH_SM_1 (continued)

Detailed implementation	Permanent and transient faults affecting the system Flash memory, memory cells and address decoder, can interfere with the access operation by the CPU, leading to wrong data or instruction fetches. Such failures can be detected by control flow monitoring techniques implemented in the application software loaded from Flash memory. For more details on the implementation, refer to description CPU_SM_1
Error reporting	Depends on implementation
Fault detection time	Depends on implementation. Higher value is fixed by watchdog timeout interval.
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	NA
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	CPU_SM_1 correct implementation supersede this requirement

Table 16. FLASH_SM_2

SM CODE	FLASH_SM_2
Description	Arm® Cortex®-M4 HardFault exceptions
Ownership	ST
Detailed implementation	Hardware random faults (both permanent and transient) affecting system Flash (memory cells, address decoder) can lead to wrong instruction codes fetches, and eventually to the intervention of the Arm® Cortex®-M4 HardFault exceptions. Refer to CPU_SM_3 for detailed description
Error reporting	Refer to CPU_SM_3
Fault detection time	Refer to CPU_SM_3
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Refer to CPU_SM_3
Periodicity	Continuous
Test for the diagnostic	Refer to CPU_SM_3
Multiple faults protection	Refer to CPU_SM_3
Recommendations and known limitations	None

Table 17. FLASH_SM_3

SM CODE	FLASH_SM_3
Description	Option byte write protection
Ownership	ST
Detailed implementation	This safety mechanism prevents unintended writes on the option byte. The use of this method is encouraged to enhance end application robustness for systematic faults
Error reporting	Write protection exception
Fault detection time	Not applicable
Addressed fault model	None (Systematic only)
Dependency on MCU configuration	None
Initialization	Not needed (enabled by default)
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	This method addresses systematic faults in software application and it have zero efficiency in addressing hardware random faults affecting the option byte value during running time. No DC value is therefore associated

Table 18. FLASH_SM_4

SM CODE	FLASH_SM_4
Description	Static data encapsulation
Ownership	End user
Detailed implementation	If static data are stored in Flash memory, encapsulation by a checksum field with encoding capability (like CRC) must be implemented. Checksum validity is checked by application software before static data consuming
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	None

Table 19. FLASH_SM_5

SM CODE	FLASH_SM_5
Description	Option byte redundancy with load verification
Ownership	ST
Detailed implementation	During option byte loading after each power-on reset, the bit-wise complementarity of the option byte and its corresponding complemented option byte is verified. Mismatches are reported as error
Error reporting	Option byte error (OPTERR) generation
Fault detection time	Not applicable
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	None (always enabled)
Periodicity	Startup
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	None

Table 20. FLASH_SM_6

SM CODE	FLASH_SM_6
Description	Flash unused area filling code
Ownership	End user
Detailed implementation	Used Flash area must be filled with deterministic data. This way in case that the program counter jumps outside the application program area due to a transient fault affecting CPU, the system evolves in a deterministic way
Error reporting	NA
Fault detection time	NA
Addressed fault model	None (Fault avoidance)
Dependency on MCU configuration	None
Initialization	NA
Periodicity	NA
Test for the diagnostic	NA
Multiple faults protection	NA
Recommendations and known limitations	Filling code can be made of NOP instructions, or an illegal code that leads to a HardFault exception raise

Table 21. FLASH_SM_8

SM CODE	FLASH_SM_8
Description	Read protection (RDP), Write protection (WRP), Proprietary code readout protection (PCROP)
Ownership	ST
Detailed implementation	Flash memory can be protected against illegal reads or erase/write by using these protection features. The combination of these techniques and the related different protection level allows End user to build an effective access protection policy.
Error reporting	Refer to functional documentation - in some cases an HardFault error is generated
Fault detection time	Refer to functional documentation
Addressed fault model	Systematic
Dependency on MCU configuration	None
Initialization	Not needed
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	Not needed
Recommendations and known limitations	Hardware random-failure detection capability for Flash access policy is restricted to well-selected marginal failure modes, mainly affecting program counter and Flash interface functions. The associated diagnostic coverage is therefore expected to be not relevant in the framework of STM32F4 Series safety concept

3.6.3 Embedded SRAM

Table 22. RAM_SM_0

SM CODE	RAM_SM_0
Description	Periodical software test for SRAM memory
Ownership	End user or ST
Detailed implementation	To enhance the coverage on SRAM data cells and to ensure adequate coverage for permanent faults affecting the address decoder it is required to execute a periodical software test on the system RAM memory. The selection of the algorithm must ensure the target SFF coverage for both the RAM cells and the address decoder. Evidences of the effectiveness of the coverage of the selected method must be also collected
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	RAM size can change according to the part number
Initialization	Depends on implementation
Periodicity	Periodic
Test for the diagnostic	Self-diagnostic capabilities can be embedded in the software, according the test implementation design strategy chosen

Table 22. RAM_SM_0 (continued)

Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Usage of a March test C- is recommended. Because the nature of this test can be destructive, RAM contents restore must be implemented. Possible interferences with interrupt-serving routines fired during test execution must be also considered (such routines can access to RAM invalid contents). Note: unused RAM section can be excluded by the testing, under end user responsibility on actual RAM usage by final application software

Table 23. RAM_SM_2

SM CODE	RAM_SM_2
Description	Stack hardening for application software
Ownership	End user
Detailed implementation	The stack hardening method is used to enhance the application software robustness to SRAM faults that affect the address decoder. The method is based on source code modification, introducing information redundancy in the stack-passed information to the called functions. Method contribution is relevant in case the combination between the final application software structure and the compiler settings requires a significant use of the stack for passing function parameters. Implementation is the same as method CPU_SM_4
Error reporting	Refer to CPU_SM_4
Fault detection time	Refer to CPU_SM_4
Addressed fault model	Refer to CPU_SM_4
Dependency on MCU configuration	Refer to CPU_SM_4
Initialization	Refer to CPU_SM_4
Periodicity	Refer to CPU_SM_4
Test for the diagnostic	Refer to CPU_SM_4
Multiple faults protection	Refer to CPU_SM_4
Recommendations and known limitations	Refer to CPU_SM_4

Table 24. RAM_SM_3

SM CODE	RAM_SM_3
Description	Information redundancy for safety-related variables in application software
Ownership	End user

Table 24. RAM_SM_3 (continued)

Detailed implementation	To address transient faults affecting SRAM controller, it is required to implement information redundancy on the safety-related system variables stored in the RAM. The guidelines for the implementation of this method are the following: – The system variables that are safety-related (in the sense that a wrong value due to a failure in reading on the RAM affects the safety functions) are well-identified and documented. – The arithmetic computation or decision based on such variables are executed twice and the two final results are compared. – Safety-related variables are stored and updated in two redundant locations, and comparison is checked before consuming data. – Enumerated fields must use non-trivial values, checked for coherence at least one time per PST – Data vectors stored in SRAM must be protected by a encoding checksum (like CRC)
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Implementation of this safety method shows a partial overlap with an already foreseen method for Cortex®-M4 (CPU_SM_1); optimizations in implementing both methods are therefore possible

Table 25. RAM_SM_4

SM CODE	RAM_SM_4
Description	Control flow monitoring in application software
Ownership	End user
Detailed implementation	In case the end user application software is executed from SRAM, permanent and transient faults affecting the memory (cells and address decoder) can interfere with the program execution. To address such failures it is needed to implement this method. For more details on the implementation, refer to description CPU_SM_1
Error reporting	Depends on implementation
Fault detection time	Depends on implementation. Higher value is fixed by watchdog timeout interval.
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous

Table 25. RAM_SM_4 (continued)

Test for the diagnostic	NA
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Needed just in case of application software execution from SRAM. CPU_SM_1 correct implementation supersedes this requirement

Table 26. RAM_SM_5

SM CODE	RAM_SM_5
Description	Periodical integrity test for application software in RAM
Ownership	End user
Detailed implementation	In case application software or diagnostic libraries are executed in RAM, it is needed to protect the integrity of the code itself against soft-error corruptions and related code mutations. This method must check the integrity of the stored code by checksum computation techniques, on a periodic basis (at least once per PST). For implementation details refer to similar method FLASH_SM_0
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Periodic
Test for the diagnostic	Self-diagnostic capabilities can be embedded in the software, according the test implementation design strategy chosen.
Multiple faults protection	CPU_SM_0: periodical core self test software CPU_SM_1: control flow monitoring in application software
Recommendations and known limitations	This method must be implemented only in case of application software or diagnostic libraries are executed from RAM

Table 27. RAM_SM_6

SM CODE	RAM_SM_6
Description	Read protection (RDP), Write protection (WRP)
Ownership	ST
Detailed implementation	SRAM2 memory can be protected against illegal reads or erase/write by using these protection features. The combination of these techniques and the related different protection level allows End user to build an effective access protection policy
Error reporting	Refer to functional documentation - in some cases an HardFault error is generated
Fault detection time	Refer to functional documentation
Addressed fault model	Systematic
Dependency on MCU configuration	SRAM2 size can vary depending on part number

Table 27. RAM_SM_6 (continued)

Initialization	Not needed
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	Not needed
Recommendations and known limitations	Hardware random-failure detection capability for SRAM2 access policy is restricted to well-selected marginal failure modes, mainly affecting program counter and SRAM2 interface functions. The associated diagnostic coverage is therefore expected to be not relevant in the framework of STM32F4 Series safety concept

3.6.4 System bus architecture

Table 28. BUS_SM_0

SM CODE	BUS_SM_0
Description	Periodical software test for interconnections
Ownership	End user
Detailed implementation	The intra-chip connection resources (Bus Matrix, AHB or APB bridges) needs to be periodically tested for permanent faults detection. Note that STM32F4 Series MCUs have no hardware safety mechanism to protect these structures. The test executes a connectivity test of these shared resources, including the testing of the arbitration mechanisms between peripherals. According to IEC 61508:2 Table A.8, A.7.4 the method is considered able to achieve high levels of coverage
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Periodic
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Implementation can be considered in large part overlapping with the widely used “Periodical read-back of configuration registers” required for several peripherals

Table 29. BUS_SM_1

SM CODE	BUS_SM_1
Description	Information redundancy in intra-chip data exchanges
Ownership	End user

Table 29. BUS_SM_1 (continued)

Detailed implementation	This method requires to add some kind of redundancy (e.g. a CRC checksum at packet level) to each data message exchanged inside the MCU. Message integrity is verified using the checksum by the application software, before consuming data.
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Implementation can be in large part overlapping with other safety mechanisms requiring information redundancy on data messages for communication peripherals. Optimizations are therefore possible

Table 30. LOCK_SM_0

SM CODE	LOCK_SM_0
Description	Lock mechanism for configuration options
Ownership	ST
Detailed implementation	The STM32F4 Series MCUs feature spread protection to prevent unintended configuration changes for some peripherals and system registers (for example PVD_LOCK, timers); the spread protection detects systematic faults in software application. The use of this method is encouraged to enhance the end application robustness to systematic faults
Error reporting	Not generated (when locked, register overwrites are just ignored)
Fault detection time	NA
Addressed fault model	None (Systematic only)
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	Not needed
Recommendations and known limitations	No DC associated because this test addresses systematic faults

3.6.5 EXTI controller

Table 31. NVIC_SM_0

SM CODE	NVIC_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This test is implemented by executing a periodical check of the configuration registers for a system peripheral against its expected value. Expected values are previously stored in RAM and adequately updated after each configuration change. The method mainly addresses transient faults affecting the configuration registers, by detecting bit flips in the registers contents. It addresses also permanent faults on registers because it is executed at least one time within PST after a peripheral update. Method must be implemented to any configuration register whose contents are able to interfere with NVIC or EXTI behavior in case of incorrect settings. Check includes NVIC vector table. According the state of the art automotive safety standard ISO26262, this method can achieve high levels of Diagnostic Coverage (refer to ISO26262:5, Table D.4) An alternative valid implementation requiring less space in SRAM can be realized on the basis of signature concept: – Peripheral registers to be checked are read in a row, computing a CRC checksum (use of hardware CRC is encouraged) – Obtained signature is compared with the golden value (computed in the same way after each register update, and stored in SRAM) – Coherence between signatures is checked by the application software – signature mismatch is considered as failure detection
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Values of configuration registers must be read after the boot before executing the first check
Periodicity	Periodic
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	This method addresses only failures affecting configuration registers, and not peripheral core logic or external interface. Attention must be paid to registers containing mixed combination of configuration and status bits. Mask must be used before saving register contents affecting signature, and related checks, to avoid false positive detections

Table 32. NVIC_SM_1

SM CODE	NVIC_SM_1
Description	Expected and unexpected interrupt check
Ownership	End user

Table 32. NVIC_SM_1 (continued)

Detailed implementation	According to IEC 61508:2 Table A.1 recommendations, a diagnostic measure for continuous, absence or cross-over of interrupt must be implemented. The method of expected and unexpected interrupt check is implemented at application software level. The guidelines for the implementation of the method are the following: – The list of the implemented interrupt for the MCU are well documented, reporting also the expected frequency of each request when possible (for example the interrupts related to ADC conversion completion, therefore coming on a deterministic way). – Individual counters are maintained for each interrupt request served, in order to detect in a given time frame the cases of a) no interrupt at all b) too many interrupt requests (“babbling idiot” interrupt source). The control of the time frame duration must be regulated according to the individual interrupt expected frequency. – Interrupt vectors related to unused interrupt source point to a default handler that reports, in case of triggering, a faulty condition (unexpected interrupt). – In case an interrupt service routine is shared between different sources, a plausibility check on the caller identity is implemented. – Interrupt requests related to non-safety-related peripherals are handled with the same method here described, despite their originator safety classification
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	In order to decrease the complexity of method implementation, it is suggested to use polling technique (when possible) instead of interrupt for end system implementation

3.6.6 Direct memory access controller (DMA)

Table 33. DMA_SM_0

SM CODE	DMA_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to DMA configuration register and channel addresses register as well. Detailed information on the implementation of this method can be found in *Section 3.6.5*.
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0

Table 33. DMA_SM_0 (continued)

Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 34. DMA_SM_1

SM CODE	DMA_SM_1
Description	Information redundancy on data packet transferred via DMA
Ownership	End user
Detailed implementation	This method is implemented adding to data packets transferred by DMA a redundancy check (like a CRC check, or similar one) with encoding capability. Full data packet redundancy would be overkilling. The checksum encoding capability must be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet Consistency of data packet must be checked by the application software before consuming data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	To give an example about checksum encoding capability, using just a bit-by-bit addition is unappropriated

Table 35. DMA_SM_2

SM CODE	DMA_SM_2
Description	Information redundancy including sender or receiver identifier on data packet transferred via DMA
Ownership	End user

Table 35. DMA_SM_2 (continued)

Detailed implementation	This method helps to identify inside the MCU the source and the originator of the message exchanged by DMA. Implementation is realized by adding an additional field to protected message, with a coding convention for message type identification fixed at MCU level. Guidelines for the identification fields are: – Identification field value must be different for each possible couple of sender or receiver on DMA transactions – Values chosen must be enumerated and non-trivial – Coherence between the identification field value and the message type is checked by application software before consuming data. This method, when implemented in combination with DMA_SM_4, makes available a kind of “virtual channel” between source and destinations entities
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	None

Table 36. DMA_SM_3

SM CODE	DMA_SM_3
Description	Periodical software test for DMA
Ownership	End user
Detailed implementation	This method requires the periodical testing of the DMA basic functionality, implemented through a deterministic transfer of a data packet from one source to another (for example from memory to memory) and the checking of the correct transfer of the message on the target. Data packets are composed by non-trivial patterns (avoid the use of 0x0000, 0xFFFF values) and organized in order to allow the detection during the check of the following failures: – Incomplete packed transfer – Errors in single transferred word – Wrong order in packed transmitted data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	Depends on implementation

Table 36. DMA_SM_3 (continued)

Periodicity	Periodic
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	None

Table 37. DMA_SM_4

SM CODE	DMA_SM_4
Description	DMA transaction awareness
Ownership	End user
Detailed implementation	DMA transactions are non-deterministic by nature, because typically driven by external events like communication messages reception. Anyway, well-designed safety systems should keep much control as possible of events – refer for instance to IEC61508:3 Table 2 item 13 requirements for software architecture. This method is based on system knowledge of frequency and type of expected DMA transaction. For instance, an externally connected sensor supposed to send periodically some messages to a STM32 peripheral. Monitoring DMA transaction by a dedicated state machine allows to detect missing or unexpected DMA activities
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Because DMA transaction termination is often linked to an interrupt generation, implementation of this method can be merged with the safety mechanism NVIC_SM_1: expected and unexpected interrupt check

3.6.7 Controller area network (bxCAN)

Table 38. CAN_SM_0

SM CODE	CAN_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to CAN configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*.
Error reporting	Refer to NVIC_SM_0

Table 38. CAN_SM_0 (continued)

Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 39. CAN_SM_1

SM CODE	CAN_SM_1
Description	Protocol error signals
Ownership	ST
Detailed implementation	CAN communication module embeds protocol error checks (like error counters) conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a marginal percentage of hardware random failures affecting the module itself. Error signals connected to these checkers are normally handled in a standard communication software, so the overhead is reduced
Error reporting	Several error condition are reported by flag bits in related CAN registers.
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	NA
Multiple faults protection	CAN_SM_2: Information redundancy techniques on messages, including end-to-end safing
Recommendations and known limitations	None

Table 40. CAN_SM_2

SM CODE	CAN_SM_2
Description	Information redundancy techniques on messages, including end-to-end safing
Ownership	End user

Table 40. CAN_SM_2 (continued)

Detailed implementation	This method aims to protect the communication between a peripheral and his external counterpart establishing a kind of “protected” channel. The aim is to specifically address communication failure modes as reported in IEC61508:2, 7.4.11.1. Implementation guidelines are the following: – Data packet must be protected (encapsulated) by an information redundancy check, like for instance a CRC checksum computed over the packet and added to payload. Checksum encoding capability must be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet. – Additional field added in payload reporting an unique identification of sender or receiver and an unique increasing sequence packet number – Timing monitoring of the message exchange (for example check the message arrival within the expected time window), detecting therefore missed message arrival conditions – Application software must verify before consuming data packet its consistency (CRC check), its legitimacy (sender or receiver) and the sequence correctness (sequence number check, no packets lost)
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Important note: it is assumed that the remote CAN counterpart has an equivalent capability of performing the checks described. A major overlap between the requirements of this method and the implementation of complex communication software protocols can exists. Due to large adoption of these protocols in industrial applications, optimizations can be possible

3.6.8 Universal synchronous/asynchronous receiver/transmitter (USART) 1/2/3/4/5/6/7/8

Table 41. UART_SM_0

SM CODE	UART_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to UART configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*.
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0

Table 41. UART_SM_0 (continued)

Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 42. UART_SM_1

SM CODE	UART_SM_1
Description	Protocol error signals
Ownership	ST
Detailed implementation	USART communication module embeds protocol error checks (like additional parity bit check, overrun, frame error) conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a marginal percentage of hardware random failures affecting the module itself. Error signals connected to these checkers are normally handled in a standard communication software, so the overhead is reduced.
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not required
Multiple faults protection	UART_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	USART communication module is fitted by several different configurations – the actual composition of communication error checks depends on selected configuration

Table 43. UART_SM_2

SM CODE	UART_SM_2
Description	Information redundancy techniques on messages
Ownership	End user
Detailed implementation	This method is implemented adding to data packets transferred by UART a redundancy check (like a CRC check, or similar one) with encoding capability. The checksum encoding capability must be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet. Consistency of data packet must be checked by the application software before consuming data

Table 43. UART_SM_2 (continued)

Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	It is assumed that the remote UART counterpart has an equivalent capability of performing the check described. Transmission full redundancy (message repetition) should not be used because its detection capability is limited to a subset of communication unit failure modes. To give an example on checksum encoding capability, using just a bit-by-bit addition is unappropriated

Table 44. UART_SM_3

SM CODE	UART_SM_3
Description	Information redundancy techniques on messages, including end-to-end safing
Ownership	End user
Detailed implementation	This method aims to protect the communication between a peripheral and his external counterpart. Refer to CAN_SM_2 description for detailed information
Error reporting	Refer to CAN_SM_2
Fault detection time	Refer to CAN_SM_2
Addressed fault model	Refer to CAN_SM_2
Dependency on MCU configuration	Refer to CAN_SM_2
Initialization	Refer to CAN_SM_2
Periodicity	Refer to CAN_SM_2
Test for the diagnostic	Refer to CAN_SM_2
Multiple faults protection	Refer to CAN_SM_2
Recommendations and known limitations	Important note: it is assumed that the remote UART counterpart has an equivalent capability of performing the checks described. Refer to CAN_SM_2 for further notice

3.6.9 Inter-integrated circuit (I2C) 1/2/3/4/5

Table 45. IIC_SM_0

SM CODE	IIC_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to I2C configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 46. IIC_SM_1

SM CODE	IIC_SM_1
Description	Protocol error signals
Ownership	ST
Detailed implementation	I2C communication module embeds protocol error checks (like overrun, underrun, packet error etc.) conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a marginal percentage of hardware random failures affecting the module itself.
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	IIC_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	Adoption of SMBus option grants the activation of more efficient protocol-level hardware checks like CRC-8 packet protection

Table 47. IIC_SM_2

SM CODE	IIC_SM_2
Description	Information redundancy techniques on messages
Ownership	End user
Detailed implementation	This method is implemented adding to data packets transferred by I2C a redundancy check (like a CRC check, or similar one) with encoding capability. The checksum encoding capability must be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet. Consistency of data packet must be checked by the application software before consuming data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	It is assumed that the remote I2C counterpart has an equivalent capability of performing the check described. Transmission full redundancy (message repetition) should not be used because its detection capability is limited to a subset of communication unit failure modes. To give an example on checksum encoding capability, using just a bit-by-bit addition is unappropriated. This method is superseded by IIC_SM_3 if hardware handled CRC insertion is possible

Table 48. IIC_SM_3

SM CODE	IIC_SM_3
Description	CRC packet-level
Ownership	ST
Detailed implementation	I2C communication module allows to activate for specific mode of operation (SMBus) the automatic insertion (and check) of CRC checksums to packet data.
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed

Table 48. IIC_SM_3 (continued)

Multiple faults protection	IIC_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	None

Table 49. IIC_SM_4

SM CODE	IIC_SM_4
Description	Information redundancy techniques on messages, including end-to-end safing
Ownership	End user
Detailed implementation	This method aims to protect the communication between a I2C peripheral and his external counterpart. Refer to CAN_SM_2 description for detailed information.
Error reporting	Refer to CAN_SM_2
Fault detection time	Refer to CAN_SM_2
Addressed fault model	Refer to CAN_SM_2
Dependency on MCU configuration	Refer to CAN_SM_2
Initialization	Refer to CAN_SM_2
Periodicity	Refer to CAN_SM_2
Test for the diagnostic	Refer to CAN_SM_2
Multiple faults protection	Refer to CAN_SM_2
Recommendations and known limitations	Important note: it is assumed that the remote I2C counterpart has an equivalent capability of performing the checks described. Refer to CAN_SM_2 for further notice

3.6.10 Serial peripheral interface (SPI) 1/2/3/4/5/6

Table 50. SPI_SM_0

SM CODE	SPI_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to SPI configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0

Table 50. SPI_SM_0 (continued)

Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 51. SPI_SM_1

SM CODE	SPI_SM_1
Description	Protocol error signals
Ownership	ST
Detailed implementation	SPI communication module embeds protocol error checks (like overrun, underrun, timeout etc.) conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a marginal percentage of hardware random failures affecting the module itself
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	NA
Multiple faults protection	SPI_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	None

Table 52. SPI_SM_2

SM CODE	SPI_SM_2
Description	Information redundancy techniques on messages
Ownership	End user
Detailed implementation	This method is implemented adding to data packets transferred by SPI a redundancy check (like a CRC check, or similar one) with encoding capability. The checksum encoding capability must be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet. Consistency of data packet must be checked by the application software before consuming data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None

Table 52. SPI_SM_2 (continued)

Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	It is assumed that the remote SPI counterpart has an equivalent capability of performing the check described. Transmission full redundancy (message repetition) should not be used because its detection capability is limited to a subset of communication unit failure modes. To give an example on checksum encoding capability, using just a bit-by-bit addition is unappropriated. This method is superseded by SSP_SM_3 if hardware handled CRC insertion is possible

Table 53. SPI_SM_3

SM CODE	SPI_SM_3
Description	CRC packet-level
Ownership	ST
Detailed implementation	SPI communication module allows to activate automatic insertion (and check) of CRC-8 or CRC-18 checksums to packet data
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	SPI_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	None

Table 54. SPI_SM_4

SM CODE	SPI_SM_4
Description	Information redundancy techniques on messages, including end-to-end safing
Ownership	End user
Detailed implementation	This method aims to protect the communication between SPI peripheral and his external counterpart. Refer to CAN_SM_2 description for detailed information.
Error reporting	Refer to CAN_SM_2
Fault detection time	Refer to CAN_SM_2

Table 54. SPI_SM_4 (continued)

Addressed fault model	Refer to CAN_SM_2
Dependency on MCU configuration	Refer to CAN_SM_2
Initialization	Refer to CAN_SM_2
Periodicity	Refer to CAN_SM_2
Test for the diagnostic	Refer to CAN_SM_2
Multiple faults protection	Refer to CAN_SM_2
Recommendations and known limitations	Important note: it is assumed that the remote SPI counterpart has an equivalent capability of performing the checks described. Refer to CAN_SM_2 for further notice

3.6.11 USB on-the-go full-speed/high-speed (OTG_FS/OTG_HS)

Table 55. USB_SM_0

SM CODE	USB_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to USB configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*.
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 56. USB_SM_1

SM CODE	USB_SM_1
Description	Protocol error signals
Ownership	ST
Detailed implementation	USB communication module embeds protocol error checks (like overrun, underrun, NRZI, bit stuffing etc.) conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a marginal percentage of hardware random failures affecting the module itself

Table 56. USB_SM_1 (continued)

Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	USB_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	None

Table 57. USB_SM_2

SM CODE	USB_SM_2
Description	Information redundancy techniques on messages
Ownership	ST or end user
Detailed implementation	The implementation of required information redundancy on messages, USB communication module is fitted by hardware capability. It basically allows to activate the automatic insertion (and check) of CRC checksums to packet data.
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Error reporting configuration, if interrupt events are planned
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	USB_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	None

Table 58. USB_SM_3

SM CODE	USB_SM_3
Description	Information redundancy techniques on messages, including end-to-end safing
Ownership	End user
Detailed implementation	This method aims to protect the communication between an USB peripheral and his external counterpart. Refer to CAN_SM_2 description for detailed information
Error reporting	Refer to CAN_SM_2

Table 58. USB_SM_3 (continued)

Fault detection time	Refer to CAN_SM_2
Addressed fault model	Refer to CAN_SM_2
Dependency on MCU configuration	Refer to CAN_SM_2
Initialization	Refer to CAN_SM_2
Periodicity	Refer to CAN_SM_2
Test for the diagnostic	Refer to CAN_SM_2
Multiple faults protection	Refer to CAN_SM_2
Recommendations and known limitations	This method apply in case USB bulk or isochronous transfers are used. For other transfers modes the USB hardware protocol already implements several features of this requirement. Refer to CAN_SM_2 for further notice

3.6.12 Analog-to-digital converters (ADC)

Table 59. ADC_SM_0

SM CODE	ADC_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to ADC configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 60. ADC_SM_1

SM CODE	ADC_SM_1
Description	Multiple acquisition by application software
Ownership	End user
Detailed implementation	This method implements a timing information redundancy by executing multiple acquisitions on the same input signal. Multiple acquisition data are then combined by a filter algorithm to determine the signal correct value

Table 60. ADC_SM_1 (continued)

Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	It is highly probable that this recommendation is satisfied by design by the end user application software. Usage of multiple acquisitions followed by average operations is a common technique in industrial applications where it is needed to survive with spurious EMI disturbs on sensor lines

Table 61. ADC_SM_2

SM CODE	ADC_SM_2
Description	Range check by application software
Ownership	End user
Detailed implementation	The guidelines for the implementation of the method are the following: – The expected range of the data to be acquired are investigated and adequately documented. Note that in a well-designed application it is improbable that during normal operation an input signal has a very near or over the upper and lower rail limit (saturation in signal acquisition). – If the application software is aware of the state of the system, this information is to be used in the range check implementation. For example, if the ADC value is the measurement of a current through a power load, reading an abnormal value such as a current flowing in opposite direction versus the load supply may indicate a fault in the acquisition module. – As the ADC module is shared between different possible external sources, the combination of plausibility checks on the different signals acquired can help to cover the whole input range in a very efficient way
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	The implementation (and the related diagnostic efficiency) of this safety mechanism are strongly application-dependent

Table 62. ADC_SM_3

SM CODE	ADC_SM_3
Description	Periodical software test for ADC
Ownership	End user
Detailed implementation	The method is implemented acquiring multiple signals and comparing the read value with the expected one, supposed to be know. Method can be implemented with different level of complexity: – Basic complexity: acquisition and check of upper or lower rails (VDD or VSS) and internal reference voltage – High complexity: in addition to basic complexity tests, acquisition of a DAC output connected to ADC input and checking all voltage excursion and linearity
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Periodic
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Combination of two different complexity method can be used to better optimize test frequency in high demand safety functions

Table 63. ADC_SM_4

SM CODE	ADC_SM_4
Description	1oo2 scheme for ADC inputs
Ownership	End user
Detailed implementation	This safety mechanism is implemented using two different SAR ADC channels to acquire the same input signal. The application software checks the coherence between the two readings. It is recommended to use two different ADC modules belonging to different ADC modules
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed

Table 63. ADC_SM_4 (continued)

Multiple faults protection	ADC_SM_0: Periodical read-back of ADC configuration registers
Recommendations and known limitations	This method can be used in conjunction with ADC_SM_0/ ADC_SM_2/ ADC_SM_3 to achieve highest level of ADC module diagnostic coverage

3.6.13 Digital-to-analog converter (DAC)

Table 64. DAC_SM_0

SM CODE	DAC_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to DAC configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 65. DAC_SM_1

SM CODE	DAC_SM_1
Description	DAC output loopback on ADC channel
Ownership	End user
Detailed implementation	Implementation is realized by routing the active DAC output to one ADC channel, and by checking the output current value with his expected one
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous or on demand
Test for the diagnostic	Not needed

Table 65. DAC_SM_1 (continued)

Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Efficiency versus transient failures is linked to final application characteristics. We define as Tm the minimum duration of DAC wrong signal permanence required to violate the related safety function(s). Efficiency is maximized when execution test frequency is higher than 1/Tm

3.6.14 Basic timers TIM 6/7

Table 66. GTIM_SM_0

SM CODE	GTIM_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to general purpose counter timer TIM6 or TIM7 configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 67. GTIM_SM_1

SM CODE	GTIM_SM_1
Description	1oo2 for counting timers
Ownership	End user
Detailed implementation	This method implements via software a 1oo2 scheme between two counting resources. The guidelines for the implementation of the method are the following: – Two timers are programmed with same time base or frequency. – In case of timer use as a time base: use in the application software one of the timer as time base source, and the other one just for check. Coherence check for the 1oo2 is done at application level, comparing two counters values each time the timer value is used to affect safety function. – In case of interrupt generation usage: use the first timer as main interrupt source for the service routines, and use the second timer as a “reference” to be checked at the initial of interrupt routine
Error reporting	Depends on implementation

Table 67. GTIM_SM_1 (continued)

Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Tolerance implementation in timer checks is recommended to avoid false positive outcomes of the diagnostic

3.6.15 Advanced, general timers TIM1/2/3/4/5/8/9/10/11/12/13/14

Note: As the timers are equipped with many different channels, each independent from the others, and possibly programmed to realize different features, the safety mechanism is selected individually for each channel.

Table 68. ATIM_SM_0

SM CODE	ATIM_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to advanced and general timers TIM1/2/3/4/5/8/9/10/11/12/13/14s. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 69. ATIM_SM_1

SM CODE	ATIM_SM_1
Description	1oo2 for counting timers
Ownership	End user

Table 69. ATIM_SM_1 (continued)

Detailed implementation	This method implements via software a 1oo2 scheme between two counting resources. The guidelines for the implementation of the method are the following: – Two timers are programmed with same time base or frequency. – In case of timer use as a time base: use in the application software one of the timer as time base source, and the other one just for check. Coherence check for the 1oo2 is done at application level, comparing two counters values each time the timer value is used to affect safety function. – In case of interrupt generation usage: use the first timer as main interrupt source for the service routines, and use the second timer as a “reference” to be checked at the initial of interrupt routine
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Tolerance implementation in timer checks is recommended to avoid false positive outcomes of the diagnostic. This method apply to timer channels merely used as elapsed time counters

Table 70. ATIM_SM_2

SM CODE	ATIM_SM_2
Description	1oo2 for input capture timers
Ownership	End user
Detailed implementation	This method is conceived to protect timers used for external signal acquisition and measurement, like “input capture” and “encoder reading”. Implementation requires to connect the external signals also to a redundant timer, and to perform a coherence check on the measured data at application level. Coherence check between timers is executed each time the reading is used by the application software
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed

Table 70. ATIM_SM_2 (continued)

Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	To reduce the potential effect of common cause failures, it is suggested to use for redundant check a channel belonging to a different timer module and mapped to non-adjacent pin on the device package

Table 71. ATIM_SM_3

SM CODE	ATIM_SM_3
Description	Loopback scheme for PWM outputs
Ownership	End user
Detailed implementation	This method is implemented by connecting the PWM to a separate timer channel to acquire the generated waveform characteristics. The guidelines are the following: – Both PWM frequency and duty cycle are measured and checked versus the expected value. – To reduce the potential effect of common cause failure, it is suggested to use for the loopback check a channel belonging to a different timer module and mapped to non-adjacent pins on the device package. This measure can be replaced under the end-user responsibility by different loopback schemes already in place in the final application and rated as equivalent. For example if the PWM is used to drive an external power load, the reading of the on-line current value can be used instead of the PWM duty cycle measurement.
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Efficiency versus transient failures is linked to final application characteristics. We define as Tm the minimum duration of PWM wrong signal permanence (wrong frequency, wrong duty, or both) required to violate the related safety function(s). Efficiency is maximized when execution test frequency is higher than 1/Tm

Table 72. ATIM_SM_4

SM CODE	ATIM_SM_4
Description	Lock bit protection for timers
Ownership	ST
Detailed implementation	This safety mechanism allows the end user to lock down specified configuration options, avoiding unintended modifications by application software. It addresses therefore software development systematic faults

Table 72. ATIM_SM_4 (continued)

Error reporting	NA
Fault detection time	NA
Addressed fault model	None (Fault avoidance)
Dependency on MCU configuration	None
Initialization	Lock protection must be enabled using LOCK bits in the TIMx_BDTR register
Periodicity	Continuous
Test for the diagnostic	NA
Multiple faults protection	NA
Recommendations and known limitations	This method does not addresses timer configuration changes due to soft-errors

3.6.16 General-purpose input/output (GPIO) - port A/B/C/D/E/F/G/H/I /J/K

Table 73. GPIO_SM_0

SM CODE	GPIO_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to GPIO configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	GPIO availability can differ according to part number
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 74. GPIO_SM_1

SM CODE	GPIO_SM_1
Description	1oo2 for input GPIO lines
Ownership	End user

Table 74. GPIO_SM_1 (continued)

Detailed implementation	This method addresses GPIO lines used as inputs. Implementation is done by connecting the external safety-related signal to two independent GPIO lines. Comparison between the two GPIO values is executed by application software each time the signal is used to affect application software behavior
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	To reduce the potential impact of common cause failure, it is recommended to use GPIO lines: – Belonging to different i/o ports (for instance PORT A and B) – With different bit port number (for instance PORTA.1 and PORTB.5) – Mapped to non-adjacent pins on the device package

Table 75. GPIO_SM_2

SM CODE	GPIO_SM_2
Description	Loopback scheme for output GPIO lines
Ownership	End user
Detailed implementation	This method addresses GPIO lines used as outputs. Implementation is done by a loopback scheme, connecting the output to a different GPIO line programmed as input and by using the input line to check the expected value on output port. Comparison is executed by application software periodically and each time output is updated
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed

Table 75. GPIO_SM_2 (continued)

Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	To reduce the potential impact of common cause failure, it is recommended to use GPIO lines: – belonging to different i/o ports (for instance PORT A and B) – with different bit number (for instance PORTA.1 and PORTB.5) – mapped to non-adjacent pins on the device package Efficiency versus transient failures is linked to final application characteristics. We define as Tm the minimum duration of GPIO output wrong signal permanence required to violate the related safety function(s). Efficiency is maximized when execution test frequency is higher than 1/Tm

Table 76. GPIO_SM_3

SM CODE	GPIO_SM_3
Description	GPIO port configuration lock register
Ownership	ST
Detailed implementation	This safety mechanism prevents configuration changes for GPIO registers; it addresses therefore systematic faults in software application. The use of this method is encouraged to enhance the end-application robustness for systematic faults
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	None (Systematic only)
Dependency on MCU configuration	None
Initialization	The correct write sequence must be applied to bit 16 (LCKK) of GPIOx_LCKR after the final GPIO configuration has been written by the application software.
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	Not needed
Recommendations and known limitations	This method does not address transient faults (soft errors) that can possibly cause bit-flips on GPIO registers at running time

3.6.17 Real-time clock module (RTC)

Table 77. RTC_SM_0

SM CODE	RTC_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to RTC configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0

Table 77. RTC_SM_0 (continued)

Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 78. RTC_SM_1

SM CODE	RTC_SM_1
Description	Application check of running RTC
Ownership	End user
Detailed implementation	The application software implements some plausibility check on RTC calendar or timing data, mainly after a power-up and further date reading by RTC. The guidelines for the implementation of the method are the following: – RTC backup registers are used to store coded information in order to detect the absence of VBAT during power-off period. – RTC backup registers are used to periodically store compressed information on current date or time – The application software executes minimal consistence checks for date reading after power-on (detecting “past” date or time retrieve). – Application software periodically checks that RTC is actually running, by reading RTC timestamp progress and comparing with an elapsed time measurement based on STM32 internal clock or timers
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Periodical
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	This method provides a limited diagnostic coverage for RTC failure modes. In case of end user application where RTC timestamps accuracy can affect in severe way the safety function (e.g. medical data storage devices), it is strongly recommended to adopt more efficient system-level measures

Table 79. RTC_SM_2

SM CODE	RTC_SM_2
Description	Information redundancy on backup registers
Ownership	End user
Detailed implementation	Data stored in RTC backup registers must be protected by a checksum with encoding capability (for instance, CRC). Checksum must be checked by application software before consuming stored data. This method guarantees data versus erases due to backup battery failures
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	Implementation of this safety method shows a partial overlap with an already foreseen method for Cortex®-M4 (CPU_SM_1); optimizations in implementing both methods are therefore possible

Table 80. RTC_SM_3

SM CODE	RTC_SM_3
Description	Application-level measures to detect failures in timestamps/event capture
Ownership	End user
Detailed implementation	This method must detect failures affecting the RTC capability to correct execute the timestamps/event capture functions. Due to the nature strictly application-dependent of this solution, no detailed guidelines for its implementation are given here
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Periodic/On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: Periodical core self-test software
Recommendations and known limitations	This method must be used only if the timestamps/event capture function is used in the safety function implementation. It is worth to note that the use of timestamp/event capture in safety related applications where the MCU is in sleep or stop mode is prevented by the assumed requirement ASR7 (refer to *Section 3.3.1*)

3.6.18 Power control

Table 81. VSUP_SM_0

SM CODE	VSUP_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 82. VSUP_SM_1

SM CODE	VSUP_SM_1
Description	Supply voltage internal monitoring (PVD)
Ownership	ST
Detailed implementation	The device features an embedded programmable voltage detector (PVD) that monitors the VDD power supply and compares it to the VPVD threshold. An interrupt can be generated when VDD drops below the VPVD threshold or when VDD is higher than the VPVD threshold
Error reporting	Interrupt Event generation
Fault detection time	Depends on threshold programming, refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Protection enable by PVDE bit and threshold programming in Power control register (PWR_CR)
Periodicity	Continuous
Test for the diagnostic	VSUP_SM_0: periodical read-back of configuration registers
Multiple faults protection	CPU_SM_5: external watchdog
Recommendations and known limitations	Internal monitoring PVD has limited capability to address failures affecting STM32F4 Series internal voltage regulator. Refer to device FMEA for details.

Table 83. VSUP_SM_2

SM CODE	VSUP_SM_2
Description	Independent watchdog
Ownership	ST
Detailed implementation	The independent watchdog is fed directly by VDD; therefore, major failures in the power supplies for digital logic (core or peripherals) does not affect its behavior but may lead to a violation of the IDWG timeout
Error reporting	Reset signal generation
Fault detection time	Depends on implementation (watchdog timeout interval)
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	IWDG activation. It is recommended to use the “Hardware watchdog” in Option byte settings (IWDG is automatically enabled after reset)
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_1: control flow monitoring in application software
Recommendations and known limitations	An external watchdog (refer to CPU_SM_5) can guarantee the same level of protection

Table 84. VSUP_SM_3

SM CODE	VSUP_SM_3
Description	Internal temperature sensor check
Ownership	End user
Detailed implementation	The internal temperature sensor must be periodically tested in order to detect abnormal increase of the die temperature – hardware faults in supply voltage system may cause excessive power consumption and consequent temperature rise
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	None
Periodicity	Periodic
Test for the diagnostic	Not needed
Multiple faults protection	VSUP_SM_3: Supply voltage internal monitoring (PVD)
Recommendations and known limitations	This method also mitigates the eventuality of common-cause affecting the MCU and due to too high temperature. Refer to the device datasheet to set the threshold temperature

3.6.19 Reset and clock control (RCC) subsystem

Table 85. CLK_SM_0

SM CODE	CLK_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to configuration registers for clock and reset system (refer to RCC register map). Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 86. CLK_SM_1

SM CODE	CLK_SM_1
Description	Clock security system (CSS)
Ownership	ST
Detailed implementation	The clock security system (CSS) detects the loss of HSE clock activity and executes the corresponding recovery action, such as: – Switch-off HSE – Commutation on the HIS – Generation of related NMI
Error reporting	NMI
Fault detection time	Depends on implementation (clock frequency value)
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	CSS protection must be enabled on Clock interrupt register (RCC_CIR) after boot stabilization
Periodicity	Continuous
Test for the diagnostic	CLK_SM_0: periodical read-back of configuration registers

Table 86. CLK_SM_1 (continued)

Multiple faults protection	CPU_SM_5: external watchdog
Recommendations and known limitations	It is recommended to carefully read Reference Manual instruction on NMI generation, in order to correct managing the faulty situation by application software features

Table 87. CLK_SM_2

SM CODE	CLK_SM_2
Description	Independent watchdog
Ownership	ST
Detailed implementation	The independent watchdog IWDG is able to detect failures in internal main MCU clock (lower frequency)
Error reporting	Reset signal generation
Fault detection time	Depends on implementation (watchdog timeout interval)
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	IWDG activation. It is recommended to use the “Hardware watchdog” in Option byte settings (IWDG is automatically enabled after reset)
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_1: control flow monitoring in application software
Recommendations and known limitations	In case of usage of IWDG window option, end user must consider possible tolerance in application software execution, to avoid false error reports (affecting system availability)

Table 88. CLK_SM_3

SM CODE	CLK_SM_3
Description	Internal clock cross-measure
Ownership	End user
Detailed implementation	This method is implemented using TIM14 capabilities to be fed by the 32 KHz RTC clock or an external clock source (if available). TIM14 counter progresses are compared with another counter (fed by internal clock). Abnormal values of oscillator frequency can be therefore detected.
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Periodic
Test for the diagnostic	Not needed

Table 88. CLK_SM_3 (continued)

Multiple faults protection	CPU_SM_1: control flow monitoring in application software CPU_SM_5: external watchdog
Recommendations and known limitations	Efficiency versus transient faults is negligible. It provides only medium efficiency in permanent clock-related failure mode coverage

3.6.20 Independent watchdog (IWDG), system window watchdog (WWDG)

Table 89. WDG_SM_0

SM CODE	WDG_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to WDG or WDG configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 90. WDG_SM_1

SM CODE	WDG_SM_1
Description	Software test for watchdog at startup
Ownership	End user
Detailed implementation	This safety mechanism ensures the right functionality of the internal watchdogs in use. At startup, the software test programs the watchdog with the required expiration timeout, stores a specific non-trivial code in SRAM and waits for the reset signal. After the watchdog reset, the software understands that the watchdog has correctly triggered, and does not execute the procedure again
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	Depends on implementation

Table 90. WDG_SM_1 (continued)

Periodicity	Startup (see below)
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	In a typical end user application, this test can be executed only at startup and during maintenance or offline periods. It could be associated to IEC61508 concept of “proof test” and so it cannot be accounted for a diagnostic coverage contribution during operating time.

3.6.21 Debug

Table 91. DBG_SM_0

SM CODE	DBG_SM_0
Description	Independent watchdog
Ownership	ST
Detailed implementation	The debug unintentional activation due to hardware random fault results in the massive disturbance of CPU operations, leading to intervention of the independent watchdog or alternately, the other system watchdog WWGDG or an external one
Error reporting	Reset signal generation
Fault detection time	Depends on implementation (watchdog timeout interval)
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_1: control flow monitoring in application software
Recommendations and known limitations	None

3.6.22 Cyclic redundancy-check module (CRC)

Table 92. CRC_SM_0

SM CODE	CRC_SM_0
Description	CRC self-coverage
Ownership	ST
Detailed implementation	The CRC algorithm implemented in this module (CRC-32 Ethernet polynomial: 0x4C11DB7) offers excellent features in terms of error detection in the message. Therefore permanent and transient faults affecting CRC computations are easily detected by any operations using the module to recompute an expected signature
Error reporting	Depends on implementation
Fault detection time	Depends on implementation

Table 92. CRC_SM_0 (continued)

Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: periodical core self-test software
Recommendations and known limitations	None

3.6.23 System configuration controller (SYSCFG)

Table 93. SYSCFG_SM_0

SM CODE	SYSCFG_SM_0
Description	Periodical read-back of configuration registers
Ownership	End user
Detailed implementation	This method must be applied to System Configuration controller configuration registers. This method is strongly recommended to protect registers related to hardware diagnostics activation and error reporting chain related features. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	This method is mainly overlapped by several other “configuration register read-back” required for other MCU peripherals. It is reported here for the sake of completeness

Table 94. DIAG_SM_0

SM CODE	DIAG_SM_0
Description	Periodical read-back of hardware diagnostics configuration registers
Ownership	End user

Table 94. DIAG_SM_0 (continued)

Detailed implementation	In STM32F4 Series several hardware-based efficient safety mechanisms (e.g. ECC on Flash) are available. This method shall be applied to any configuration register related to diagnostic measure operations, including error reporting. End user shall therefore individuate configuration registers related to: – Hardware diagnostic enable – Interrupt/NMI enable (if used for diagnostic error management)
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

3.6.24 SD/SDIO/MMC card host interface (SDMMC)

Table 95. SDIO_SM_0

SM CODE	SDIO_SM_0
Description	Periodical read-back of SDIO/SMMC configuration registers
Ownership	End user
Detailed implementation	This method must be applied to SDIO/SMMC configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 96. SDIO_SM_1

SM CODE	SDIO_SM_1
Description	Protocol error signals including hardware CRC
Ownership	ST
Detailed implementation	SDIO/SMMC communication module embeds protocol error checks (like overrun, underrun, timeout etc.) and CRC-packet checks as well, conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a percentage of hardware random failures affecting the module itself
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (for example baud rate), please refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	SDIO_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	-

Table 97. SDIO_SM_2

SM CODE	SDIO_SM_2
Description	Information redundancy techniques on messages
Ownership	End user
Detailed implementation	This method is implemented adding to data packets transferred by SDIO/SMMC a redundancy check (like a CRC check, or similar one) with encoding capability. The checksum encoding capability shall be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet. Consistency of data packet shall be checked by the application software before consuming data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed

Table 97. SDIO_SM_2

Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	To give an example on checksum encoding capability, using just a bit-by-bit addition is unappropriated. This safety mechanism can overlap with information redundancy techniques implemented at system level to address failure of physical device connected to SDIO/SMMMC port

Note: The safety mechanisms mentioned above are addressing the SDIO/SMMC interface included in STM32 MCUs. No claims are done in this Safety Manual about the mitigation of hardware random faults affecting the external memory connected to SDIO/SMMC port.

3.6.25 DSI Host (DSIHOST)

Table 98. DSI_SM_0

SM CODE	DSI_SM_0
Description	Periodical read-back of DSI Host configuration registers.
Ownership	End User
Detailed implementation	This method must be applied to DSI Host configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5: EXTI controller*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 99. DSI_SM_1

SM CODE	DSI_SM_1
Description	Protocol error signals and information redundancy including hardware checksums
Ownership	ST
Detailed implementation	DSI communication/command protocol is based on a packet handling concept, including (where applicable) ECC and checksum capabilities. This mechanism, mainly implemented to manage on field communication disturbance, is able to achieve a relevant diagnostic coverage on several DSI module failure modes
Error reporting	Error conditions are reported by flag bits in related registers

Table 99. DSI_SM_1 (continued)

Fault detection time	Depends on peripheral configuration and the type of violation detected, please refer to functional documentation
Addressed fault model	Permanent & Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	DSI_SM_0: Periodical read-back of DSI Host configuration registers.
Recommendations and known limitations	-

Note: The above-described safety mechanisms addresses the DSI interface included in STM32 MCUs, including PHY. Because actual capability of correct physical signal generation to drive the connected monitor is not addressed by these safety mechanisms, in case such feature is considered safety relevant the End user is warned to evaluate the adoption of adequate system-level measures.

3.6.26 Ethernet (ETH): media access control (MAC) with DMA controller.

Table 100. FSMC_SM_0

SM CODE	ETH_SM_0
Description	Periodical read-back of Ethernet configuration registers.
Ownership	End User
Detailed implementation	This method must be applied to Ethernet configuration registers (including those relate to unused module features) Detailed information on the implementation of this method can be found in *Section 3.6.5: EXTI controller*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 101. FSMC_SM_1

SM CODE	ETH_SM_1
Description	Protocol error signals including hardware CRC
Ownership	ST
Detailed implementation	Ethernet communication module embeds protocol error checks (like overrun, underrun, timeout, packet composition violation etc.) and CRC-packet checks as well, conceived to detect network-related abnormal conditions. These mechanisms are able anyway to detect a percentage of hardware random failures affecting the module itself.
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), please refer to functional documentation
Addressed fault model	Permanent & Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	SDIO_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	-

Table 102. FSMC_SM_2

SM CODE	ETH_SM_2
Description	Information redundancy techniques on messages, including End to End safing End User
Ownership	This method aim to protect the communication between a peripheral and his external counterpart. It is used in Ethernet local safety concept to address failures not detected by ETH_SM_1 and to increase its associated diagnostic coverage. Refer to CAN_SM_2 description for detailed information.
Detailed implementation	Depends on implementation
Error reporting	Depends on implementation
Fault detection time	Permanent & Transient
Addressed fault model	None
Dependency on MCU configuration	
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed

Table 102. FSMC_SM_2 (continued)

Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	The implementation on the application software of complex Ethernet-based communication stacks (like TCP/IP) is able to satisfy the requirements of this method.

Note: The use of DMA feature inside Ethernet module will bring to adopt the same set of safety mechanism defined for the system DMA (refer to Section 3.6.6: Direct memory access controller (DMA)).

3.6.27 Flexible static memory controller (FSMC)

Table 103. FSMC_SM_0

SM CODE	FSMC_SM_0
Description	Control flow monitoring in application software
Ownership	End user
Detailed implementation	If FSMC is used to connect an external memory containing software code to be executed by the CPU, permanent and transient faults affecting the FSMC memory controller are able to interfere with the access operation by the CPU, leading to wrong data or instruction fetches. A strong control flow mechanism linked to a system watchdog is able to detect such failures, in case they interfere with the expected flow of the application software. The implementation of this method is identical to the one reported for CPU_SM_1, refer there for details
Error reporting	Depends on implementation
Fault detection time	Depends on implementation. Higher value is fixed by watchdog timeout interval
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	FSMC interface is available only on selected part numbers
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	This mechanism must be used just if FSMC external memory is used to store executable programs

Table 104. FSMC_SM_1

SM CODE	FSMC_SM_1
Description	Information redundancy on external memory connected to FSMC
Ownership	End user

Table 104. FSMC_SM_1 (continued)

Detailed implementation	If FSMC interface is used to connect an external memory where safety-relevant data are stored, information redundancy techniques for stored data are able to address faults affecting the FSMC interface. The possible techniques are: To use redundant copies of safety relevant data and perform coherence check before consuming. To organize data in arrays and compute the checksum field to be checked before use
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	FSMC interface is available only on selected part numbers
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	This mechanism must be used just if FSMC external memory is used to store safety-related data. This safety mechanism can overlap with information redundancy techniques implemented at system level to address failure of physical device connected to FSMC port

Table 105. FSMC_SM_2

SM CODE	FSMC_SM_2
Description	Periodical read-back of FSMC configuration registers
Ownership	End user
Detailed implementation	This method must be applied to FSMC configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	FSMC interface is available only on selected part numbers
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 106. FSMC_SM_3

SM CODE	FSMC_SM_3
Description	ECC engine on NAND interface in FSMC module
Ownership	ST
Detailed implementation	The FMC NAND Card controller includes two error correction code computation hardware blocks, one per memory bank. They reduce the host CPU workload when processing the ECC by software. ECC mechanism protects data integrity on the external memory connected to NAND port
Error reporting	Refer to functional documentation
Fault detection time	ECC bits are checked during a memory reading
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	FSMC interface is available only on selected part numbers
Initialization	None
Periodicity	Continuous
Test for the diagnostic	Not needed
Multiple faults protection	FSMC_SM_2: Periodical read-back of FSMC configuration registers
Recommendations and known limitations	This method has negligible efficiency in detecting hardware random failures affecting the FSMC interface. It can be part of End user safety concept because addressing memories outside STM32F4 MCU

3.6.28 Quad-SPI interface (QUADSPI)

Table 107. QSPI_SM_0

SM CODE	QSPI_SM_0
Description	Periodical read-back of QUADSPI configuration registers
Ownership	End user
Detailed implementation	This method must be applied to QUADSPI configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 108. QSPI_SM_1

SM CODE	QSPI_SM_1
Description	Protocol error signals including hardware CRC
Ownership	ST
Detailed implementation	QUADSPI communication module embeds protocol error checks (like overrun, underrun, timeout etc.), conceived to detect communication-related abnormal conditions. These mechanisms are able anyway to detect a percentage of hardware random failures affecting the module itself.
Error reporting	Error flag raise and optional Interrupt Event generation
Fault detection time	Depends on peripheral configuration (e.g. baud rate), please refer to functional documentation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	QSPI_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	-

Table 109. QSPI_SM_2

SM CODE	QSPI_SM_2
Description	Information redundancy techniques on messages
Ownership	End user
Detailed implementation	This method is implemented adding to data packets (not commands) transferred by QSPI interface a redundancy check (like a CRC check, or similar one) with encoding capability. The checksum encoding capability shall be robust enough to guarantee at least 90% probability of detection for a single bit flip in the data packet. Consistency of data packet shall be checked by the application software before consuming data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed

Table 109. QSPI_SM_2 (continued)

Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	To give an example on checksum encoding capability, using just a bit-by-bit addition is unappropriated. This safety mechanism can overlap with information redundancy techniques implemented at system level to address failure of physical device connected to QSPI port

3.6.29 Serial audio interface (SAI)

Table 110. SAI_SM_0

SM CODE	SAI_SM_0
Description	Periodical read-back of SAI configuration registers
Ownership	End user
Detailed implementation	This method must be applied to SAI configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 111. SAI_SM_1

SM CODE	SAI_SM_1
Description	SAI output loopback scheme
Ownership	End user
Detailed implementation	This method uses a loopback scheme to detect permanent and transient faults on the output channel used for serial audio frame generation. It is implemented by connecting the second serial audio interface as input for primary output generation. The application software is able therefore to identify wrong or missing audio frame generation
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None

Table 111. SAI_SM_1 (continued)

Initialization	Depends on implementation
Periodicity	Continuous/ On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	Efficiency versus transient failures is linked to final application characteristics. We define as Tm the minimum duration of serial audio wrong signal permanence required to violate the related safety function(s). Efficiency is maximized when execution test frequency is higher than 1/Tm. Method to be used when SAI interface safety-related use is “audio stream generation”

Table 112. SAI_SM_2

SM CODE	SAI_SM_2
Description	1oo2 scheme for SAI module
Ownership	End user
Detailed implementation	This safety mechanism is implemented using the two SAI interfaces to decode/receive the same input stream audio. The application software checks the coherence between the received data
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	None
Initialization	Depends on implementation
Periodicity	On demand
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	The MCU performance overload and the implementation complexity associated to this method can be relevant. Method to be used when SAI interface safety-related use is “audio stream receive”

3.6.30 True random number generator (RNG)

Table 113. RNG_SM_0

SM CODE	RNG_SM_0
Description	Periodical read-back of RNG configuration register RNG_CR
Ownership	End user
Detailed implementation	This method must be applied to RNG configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0

Table 113. RNG_SM_0 (continued)

Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	RNG module available only on specific part numbers
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 114. RNG_SM_1

SM CODE	RNG_SM_1
Description	RNG module entropy on-line tests
Ownership	ST and End user
Detailed implementation	RNG module include an internal diagnostic for the analog source entropy that can be used to detect failures on the module itself. Furthermore, the required test on generated random number difference between the previous one (as required by FIPS PUB 140-2) can be exploited as well. Implementation: – Check for RNG error conditions – Check the difference between generated random number and the previous one
Error reporting	CEIS, SEIS error bits in RNG status register (RNG_SR) Application software error for FIPS PUB 140-2 test fail
Fault detection time	Depends on implementation
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	RNG module available only on specific part numbers
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	-

3.6.31 Cryptographic processor (CRYP)

Table 115. CRYP_SM_0

SM CODE	CRYP_SM_0
Description	Periodical read-back of CRYP configuration registers
Ownership	End user
Detailed implementation	This method must be applied to CRYP configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	CRYP module available only on specific part numbers
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 116. CRYP_SM_1

SM CODE	CRYP_SM_1
Description	Encryption/decryption collateral detection
Ownership	ST
Detailed implementation	Encryption and decryption operations performed by CRYP module are composed by several data manipulations and checks, with different level of complexity according to the selected chaining algorithm. A major part of the hardware random failures affecting CRYP module leads to algorithm violations/errors. Leading to decoding errors on the receiver side
Error reporting	Several error condition can happens, check functional documentation
Fault detection time	Depends on peripheral configuration
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	CRYP module available only on specific part numbers
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	CRYP_SM_2: Information redundancy techniques on messages
Recommendations and known limitations	-

Table 117. CRYP_SM_2

SM CODE	CRYP_SM_2
Description	Information redundancy techniques on messages, including end-to-end safing
Ownership	End user
Detailed implementation	This method aim to protect the communication between a peripheral and his external counterpart. It is used in CRYP local safety concept to address failures not detected by the encryption/decryption features. Refer to CAN_SM_2 description for detailed information
Error reporting	Refer to CAN_SM_2
Fault detection time	Refer to CAN_SM_2
Addressed fault model	Refer to CAN_SM_2
Dependency on MCU configuration	CRYP module available only on specific part numbers
Initialization	Refer to CAN_SM_2
Periodicity	Refer to CAN_SM_2
Test for the diagnostic	Refer to CAN_SM_2
Multiple faults protection	Refer to CAN_SM_2
Recommendations and known limitations	Important note: it is assumed that the remote counterpart has an equivalent capability of performing the checks described. Refer to CAN_SM_2 for further notice

Note: Hardware random failures consequences on potential security features violations are ***not*** *analyzed in this manual.*

3.6.32 HASH processor (HASH)

Table 118. HASH_SM_0

SM CODE	HASH_SM_0
Description	Periodical read-back of HASH configuration registers
Ownership	End user
Detailed implementation	This method must be applied to HASH configuration registers. Detailed information on the implementation of this method can be found in *Section 3.6.5*
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	HASH module available only on specific part numbers
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0

Table 118. HASH_SM_0 (continued)

Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 119. HASH_SM_1

SM CODE	HASH_SM_1
Description	HASH processing collateral detection
Ownership	ST
Detailed implementation	Message digest computation performed by HASH module is composed by several data manipulations and checks. A major part of the hardware random failures affecting HASH module will lead to algorithm violations/errors, and so to decoding errors on the receiver side
Error reporting	Several error condition can happens, check functional documentation
Fault detection time	Depends on peripheral configuration
Addressed fault model	Permanent and Transient
Dependency on MCU configuration	HASH module available only on specific part numbers
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	HASH_SM_0: periodical read-back of HASH configuration registers CPU_SM_0: periodical core self-test software
Recommendations and known limitations	-

*Note: Hardware random failures consequences on potential security features violations are **not** analyzed in this manual.*

3.6.33 Digital camera interface (DCMI)

Table 120. DCMI_SM_0

SM CODE	DCMI_SM_0
Description	Periodical read-back of DCMI configuration registers
Ownership	End user
Detailed implementation	This method must be applied to DCMI configuration registers. Detailed information on the implementation of this method can be found in Section 3.6.5.
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0

Table 120. DCMI_SM_0 (continued)

Dependency on MCU configuration	DCMI interface is available only on selected part numbers
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 121. DCMI_SM_1

SM CODE	DCMI_SM_1
Description	DCMI video input data synchronization
Ownership	ST
Detailed implementation	According to the nature of video data stream received, DCMI module implements synchronization controls, from the simplest one (hardware synchronization) to the most complex (e.g. embedded data synchronization mode). DCMI internal failures leading to the incapability of correcting synchronizing the data stream can be therefore detected
Error reporting	No explicit error signal/message generation is provided (*)
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	DCMI interface is available only on selected part numbers
Initialization	Depends on implementation
Periodicity	Continuous
Test for the diagnostic	N/A
Multiple faults protection	DCMI_SM_0: Periodical read-back of DCMI configuration registers
Recommendations and known limitations	(*) For its nature, the detection of an actual hardware failure by this safety mechanism can be confused with functional-related scenarios (e.g. camera device disconnected or powered-off). It is responsibility of the application software to discriminate, as far as it is technically possible, among different events

3.6.34 LCD-TFT Display Controller (LTDC)

Table 122. LCD_SM_0

SM CODE	LCD_SM_0
Description	Periodical read-back of LCD configuration registers and buffer memory.
Ownership	End user
Detailed implementation	This method must be applied to LCD configuration registers and to the buffer memory as well. Detailed information on the implementation of this method can be found in *Section 3.6.5*

Table 122. LCD_SM_0 (continued)

Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0
Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

Table 123. LCD_SM_1

SM CODE	LCD_SM_1
Description	LCD signals acquisition by ADC channel
Ownership	End user
Detailed implementation	Correct generation of LCD driving signals is checked by ADC reading versus expected values
Error reporting	Depends on implementation
Fault detection time	Depends on implementation
Addressed fault model	Permanent
Dependency on MCU configuration	None
Initialization	None
Periodicity	Periodic
Test for the diagnostic	Not needed
Multiple faults protection	CPU_SM_0: Periodical core self test software
Recommendations and known limitations	This method is conceived to mainly detect permanent failures affecting analog parts and therefore the execution on periodic way is acceptable. Diagnostic coverage achievable depends on the quantity of LCD signals checked

Note: The above-described safety mechanism addresses the LCD interface included in STM32 MCUs. Because actual capability of correct image generation on LCD is not addressed by this safety mechanism, in case such feature is considered safety relevant the End user is warned to evaluate the adoption of adequate system-level measures.

3.6.35 Disable and periodic cross-check of unintentional activation of unused peripherals

This section reports the safety mechanism that addresses peripherals not used by the safety application, or not used at all.

Table 124. FFI_SM_0

SM CODE	FFI_SM_0
Description	Unused peripherals disable
Ownership	End user
Detailed implementation	This method contributes to the reduction of the probability of cross-interferences caused by peripherals not used by the software application, in case a hardware failure causes an unintentional activation. After the system boot, the application software must disable all unused peripherals with this procedure: – Enable reset flag on AHB and APB peripheral reset register – Disable clock distribution on AHB and APB peripheral clock enable register
Error reporting	NA
Fault detection time	NA
Addressed fault model	NA
Dependency on MCU configuration	None
Initialization	NA
Periodicity	Startup
Test for the diagnostic	Not needed
Multiple faults protection	FFI_SM_1: Periodical read-back of interference avoidance registers
Recommendations and known limitations	None

Table 125. FFI_SM_1

SM CODE	FFI_SM_1
Description	Periodical read-back of interference avoidance registers
Ownership	End user
Detailed implementation	This method contributes to the reduction of the probability of cross-interferences between peripherals that can potentially conflict on the same input/output pins, including for instance unused peripherals. This diagnostic measure must be applied to following registers: – Clock enable and disable registers – Alternate functions programming registers Detailed information on the implementation of this method can be found in *Section 3.6.5*.
Error reporting	Refer to NVIC_SM_0
Fault detection time	Refer to NVIC_SM_0

Table 125. FFI_SM_1 (continued)

Addressed fault model	Refer to NVIC_SM_0
Dependency on MCU configuration	Refer to NVIC_SM_0
Initialization	Refer to NVIC_SM_0
Periodicity	Refer to NVIC_SM_0
Test for the diagnostic	Refer to NVIC_SM_0
Multiple faults protection	Refer to NVIC_SM_0
Recommendations and known limitations	Refer to NVIC_SM_0

3.7 Conditions of use

Table 126 provides a summary of the safety concept recommendations reported in *Section 3.6: Description of hardware and software diagnostics*. The conditions of use to be applied to STM32F4 Series MCUs are reported in form of safety mechanism requirements. Exception is represented by some conditions of use introduced by FMEA analysis in order to correctly address specific failure modes. These conditions of use are reported at the end of *Table 126*.

Rank column reports how related safety mechanism has been considered during the analysis, with following meaning:

- M = this safety mechanism is always operating during normal operations – no end user activity can deactivate it.
- ++ = Highly recommended being a common practice and considered in this Safety Manual for the computation of the safety metrics to achieve SIL2 on a single MCU.
- + = Recommended as additional safety measure, but not considered in this Safety Manual for the computation of safety metrics. STM32F4 Series users can skip the implementation in case it is in contradiction with functional requirements or overlapped by another mechanism marked as “++”.
- o = optional, not needed or related to specific MCU configuration

The “X” marker in the “Perm” and “Trans” columns in *Table 126*, indicates that the related safety mechanism is effective for such fault model.

Table 126. List of safety mechanisms

STM32F4 Series function	Diagnostic	Description	Rank	Perm	Trans
Arm® Cortex®-M4 CPU	CPU_SM_0	Periodical software test addressing permanent faults in Arm® Cortex®-M4 CPU core	++	X	-
	CPU_SM_1	Control flow monitoring in application software	++	X	X
	CPU_SM_2	Double computation in application software	++	-	X
	CPU_SM_3	Arm® Cortex®-M4 HardFault exceptions	M	X	X
	CPU_SM_4	Stack hardening for application software	+	X	X
	CPU_SM_5	External watchdog	+(1)	X	X
	CPU_SM_6	Independent watchdog	++(1)	X	X
	CPU_SM_7	MPU – Memory protection unit	++(2)	X	X
	MPU_SM_0	Periodical read-back of MPU configuration registers	++(2)	X	X
Embedded Flash memory	FLASH_SM_0	Periodical software test for Flash memory	++	X	-
	FLASH_SM_1	Control flow monitoring in application software	++	X	X
	FLASH_SM_2	Arm® Cortex®-M4 HardFault exceptions	M	X	X
	FLASH_SM_3	Option byte write protection	M	-	-
	FLASH_SM_4	Static data encapsulation	++	X	X
	FLASH_SM_5	Option byte redundancy with load verification	M	X	X
	FLASH_SM_6	Flash unused area filling code	+	-	-
	FLASH_SM_8	Read/Write/Proprietary code protection	+	-	-
Embedded SRAM	RAM_SM_0	Periodical software test for SRAM memory	++	X	-
	RAM_SM_2	Stack hardening for application software	+	X	X
	RAM_SM_3	Information redundancy for system variables in application software	++	X	X
	RAM_SM_4	Control flow monitoring in application software	o(3)	X	X
	RAM_SM_5	Periodical integrity test for application software in RAM	o(3)	X	X
	RAM_SM_6	Read protection (RDP), Write protection (WRP)	+	-	-

Table 126. List of safety mechanisms (continued)

STM32F4 Series function	Diagnostic	Description	Rank	Perm	Trans
System architecture	BUS_SM_0	Periodical software test for interconnections	++	X	-
	BUS_SM_1	Information redundancy in intra-chip data exchanges	++	X	X
EXTI controller	NVIC_SM_0	Periodical read-back of configuration registers	++	X	X
	NVIC_SM_1	Expected and unexpected interrupt check by application software	++	X	X
DMA	DMA_SM_0	Periodical read-back of configuration registers	++	X	X
	DMA_SM_1	Information redundancy on data packet transferred via DMA	++	X	X
	DMA_SM_2	Information redundancy including sender or receiver identifier on data packet transferred via DMA	++	X	X
	DMA_SM_3	Periodical software test for DMA	++	X	-
	DMA_SM_4	DMA transaction awareness	++	X	X
bxCAN	CAN_SM_0	Periodical read-back of configuration registers	++	X	X
	CAN_SM_1	Protocol error signals	++	X	X
	CAN_SM_2	Information redundancy techniques on messages, including end-to-end safing	++	X	X
USART	UART_SM_0	Periodical read-back of configuration registers	++	X	X
	UART_SM_1	Protocol error signals	++	X	X
	UART_SM_2	Information redundancy techniques on messages	++	X	X
	UART_SM_3	Information redundancy techniques on messages, including end-to-end safing	++	X	X
I2C	IIC_SM_0	Periodical read-back of configuration registers	++	X	X
	IIC_SM_1	Protocol error signals	++	X	X
	IIC_SM_2	Information redundancy techniques on messages	++	X	X
	IIC_SM_3	CRC packet-level	+	X	X
	IIC_SM_4	Information redundancy techniques on messages, including end-to-end safing	+	X	X

Table 126. List of safety mechanisms (continued)

STM32F4 Series function	Diagnostic	Description	Rank	Perm	Trans
SPI	SPI_SM_0	Periodical read-back of configuration registers	++	X	X
	SPI_SM_1	Protocol error signals	++	X	X
	SPI_SM_2	Information redundancy techniques on messages	++	X	X
	SPI_SM_3	CRC packet-level	+	X	X
	SPI_SM_4	Information redundancy techniques on messages, including end-to-end Safing	+	X	X
USB OTG	USB_SM_0	Periodical read-back of configuration registers	++	X	X
	USB_SM_1	Protocol error signals	++	X	X
	USB_SM_2	Information redundancy techniques on messages	++	X	X
	USB_SM_3	Information redundancy techniques on messages, including end-to-end safing	+	X	X
ADC	ADC_SM_0	Periodical read-back of configuration registers	++	X	X
	ADC_SM_1	Multiple acquisition by application software	++	-	X
	ADC_SM_2	Range check by application software	++	X	X
	ADC_SM_3	Periodical software test for ADC	++	X	-
	ADC_SM_4	1oo2 scheme for ADC inputs	+	X	X
DAC	DAC_SM_0	Periodical read-back of configuration registers	++	X	X
	DAC_SM_1	DAC output loopback on ADC channel	++	X	X
Basic timers TIM6/7	GTIM_SM_0	Periodical read-back of configuration registers	++	X	X
	GTIM_SM_1	1oo2 for counting timers	++	X	X
Advanced, general and low-power timers TIM1/2/3/4/5/8/9/10/11/12/13/14	ATIM_SM_0	Periodical read-back of configuration registers	++	X	X
	ATIM_SM_1	1oo2 for counting timers	++	X	X
	ATIM_SM_2	1oo2 for input capture timers	++	X	X
	ATIM_SM_3	Loopback scheme for PWM outputs	++	X	X
	ATIM_SM_4	Lock bit protection for timers	+	-	-
CRC	CRC_SM_0	CRC self-coverage	++	X	X

Table 126. List of safety mechanisms (continued)

STM32F4 Series function	Diagnostic	Description	Rank	Perm	Trans
GPIO	GPIO_SM_0	Periodical read-back of configuration registers	++	X	X
	GPIO_SM_1	1oo2 for input GPIO lines	++	X	X
	GPIO_SM_2	Loopback scheme for output GPIO lines	++	X	X
	GPIO_SM_3	GPIO port configuration lock register	+	-	-
RTC	RTC_SM_0	Periodical read-back of configuration registers	++	X	X
	RTC_SM_1	Application check of running RTC	++	X	X
	RTC_SM_2	Information redundancy on backup registers	+	X	X
	RTC_SM_3	Application-level measures to detect failures in timestamps or event capture	o	X	X
Power control	VSUP_SM_0	Periodical read-back of configuration registers	++	X	X
	VSUP_SM_1	Supply voltage monitoring	++	X	-
	VSUP_SM_2	Independent Watchdog	++	X	-
	VSUP_SM_3	Internal temperature sensor check	o	-	-
Clock and Reset	CLK_SM_0	Periodical read-back of configuration registers	++	X	X
	CLK_SM_1	CSS Clock Security System	++	X	-
	CLK_SM_2	Independent Watchdog	++	X	-
	CLK_SM_3	Internal clock cross-measure	+	X	-
IWDG/WWDG	WDG_SM_0	Periodical read-back of configuration registers	++	X	X
	WDG_SM_1	Software test for watchdog at startup	o	X	-
Debug	DBG_SM_0	Independent watchdog	++	X	X
System or peripheral control	LOCK_SM_0	Lock mechanism for configuration options	+	-	-
	SYSCFG_SM_0	Periodical read-back of configuration registers	++	X	X
	DIAG_SM_0	Periodical read-back of hardware diagnostics configuration registers	++	X	X
SDMMC	SDIO_SM_0	Periodical read-back of SDIO/SMMC configuration registers.	++	X	X
	SDIO_SM_1	Protocol error signals including hardware CRC	++	X	X
	SDIO_SM_2	Information redundancy techniques on messages	++	X	X

Table 126. List of safety mechanisms (continued)

STM32F4 Series function	Diagnostic	Description	Rank	Perm	Trans
DSI HOST (DSIHOST)	SDI_SM_0	Periodical read-back of DSI Host configuration registers	++	X	X
	SDI_SM_1	Protocol error signals and information redundancy including hardware checksums	++	X	X
Ethernet (ETH): media access control (MAC) with DMA controller	ETH_SM_0	Periodical read-back of Ethernet configuration registers.	++	X	X
	ETH_SM_1	Protocol error signals including hardware CRC	++	X	X
	ETH_SM_2	Information redundancy techniques on messages, including End to End safing	++	X	X
Flexible static memory controller (FSMC)	FSMC_SM_0	Control flow monitoring in application software	++(4)	X	X
	FSMC_SM_1	Information redundancy on external memory connected to FSMC	++(4)	X	X
	FSMC_SM_2	Periodical read-back of FSMC configuration registers.	++	X	X
	FSMC_SM_3	ECC engine on NAND interface in FSMC module	o	X	X
QUADSPI	QSPI_SM_0	Periodical read-back of QUADSPI configuration registers.	++	X	X
	QSPI_SM_1	Protocol error signals including hardware CRC	++	X	X
	QSPI_SM_2	Information redundancy techniques on messages	++	X	X
SAI	SAI_SM_0	Periodical read-back of SAI configuration registers.	++	X	X
	SAI_SM_1	SAI output loopback scheme	++	X	X
	SAI_SM_2	1oo2 scheme for SAI module	++	X	X
RNG	RNG_SM_0	Periodical read-back of RNG configuration register RNG_CR.	++	X	X
	RNG_SM_1	RNG module entropy on-line tests	++	X	-
CRYP	CRYP_SM_0	Periodical read-back of CRYP configuration registers	++	X	X
	CRYP_SM_1	Encryption/decryption collateral detection	++	X	X
	CRYP_SM_2	Information redundancy techniques on messages, including end-to-end safing	++	X	X

Table 126. List of safety mechanisms (continued)

STM32F4 Series function	Diagnostic	Description	Rank	Perm	Trans
HASH	HASH_SM_0	Periodical read-back of HASH configuration registers	++	X	X
	HASH_SM_1	HASH processing collateral detection	++	X	X
DCMI	DCMI_SM_0	Periodical read-back of DCMI configuration registers	++	X	X
	DCMI_SM_1	DCMI video input data synchronization	++	X	X
LCD-TFT	LCD_SM_0	Periodical read-back of LCD configuration registers and buffer memory.	++	X	X
	LCD_SM_1	LCD acquisition by ADC channel	++	X	-
Part separation (no interference)	FFI_SM_0	Unused peripherals disable	++	-	-
	FFI_SM_1	Periodical read-back of interference avoidance registers	++	-	-
Arm® Cortex®-M4 CPU	CoU_1	The reset condition of Arm® Cortex®-M4 CPU must be compatible as valid safe state at system level	++	-	-
Debug	CoU_2	STM32F4 Series debug features must not be used in safety function(s) implementation	++	-	-
Arm® Cortex®-M4 / Supply system	CoU_3	Low power mode state must not be used in safety function(s) implementation	++	-	-
STM32F4 Series peripherals	CoU_4	End user must implement the required combination of safety mechanism/CoUs for each STM32 peripherals used in safety function(s) implementation	++	X	X
Flash subsystem	CoU_5	During Flash bank mass erase and reprogramming there must not be safety functions(s) executed by STM32F4 MCU.	++	-	-
Flash subsystem	CoU_6	On-field application software live update by dual-Flash system must include the execution of code/data integrity check by methods like FLASH_SM_0	++	X	X
CPU subsystem	CoU_7	In case of multiple safety functions implementations, methods to guarantee their mutual independence must include MPU use.	++	-	-

1. To achieve SIL2 on a single MCU, method CPU_SM_5 is rated as “+”. Anyway, in case of specific definition for system-level safe state, it can be necessary to rate CPU_SM_5 as ++; in that case CPU_SM_6 can be rated as “+”. Refer to the “Recommendations” row of *Table 11: CPU_SM_6* for more details.
2. Can be considered ranked as “+” if no multiple safety functions are implemented.
3. Must be considered ranked as “++” if the application software is executed on RAM.
4. Can be considered ranked as “o” depending on the intended use of external memory connected to FSMC.

The above-described safety mechanism or conditions of use are conceived with different levels of abstraction depending on their nature: the more a safety mechanism is implemented as application-independent, the wider is its possible use on a large range of end-user applications.

The safety analysis highlights two major partitions inside the MCU:

- System-critical MCU modules. Every end-user application is affected from a safety point of view by a failure on these modules. Because these modules are used by every end user application, related methods or safety mechanism are mainly conceived to be application-independent. For STM32F4 Series microcontroller system critical modules are: CPU, Reset, Power, Clocks, Busmatrixs and Interconnect, Flash and RAM memories (including their interfaces)
- Peripheral modules. Such modules could be not used by the end-user application, or they could be used for non-safety related tasks. Related safety methods are therefore implemented mainly at application level, as application software solutions or architectural solutions.

4 Safety results

This section reports the results of the safety analysis of the STM32F4 Series MCUs, according to IEC 61508 and to ST methodology flow, related to the hardware random and dependent failures.

4.1 Random hardware failure safety results

The analysis for random hardware failures of STM32F4 Series devices reported in this Safety Manual is executed according to ST methodology flow for safety analysis of semiconductor devices according IEC61508. The accuracy of results obtained are guaranteed by three factors:

- ST methodology flow strict adherence to IEC61508 requirements and prescriptions
- The use during the analysis of detailed and reliable information on microcontroller design
- The use of state-of-the-art fault injections methods and tools for safety metrics verification

The STM32F4 Series safety analysis has been therefore able to explore the overall and exhaustive list of MCU failure modes, and to individuate for each of them an adequate mitigation measure (safety mechanism). The overall list of STM32F4 Series failure modes is maintained in related FMEA document. STM32F4 Series FMEA document can be provided on demand, refer to your local ST sales contact.

In summary, with the adoptions of the safety mechanisms and conditions of use reported in *Section 3.7: Conditions of use*, it is possible to achieve the integrity levels summarized in *Table 127*.

Table 127. Overall achievable safety integrity levels

MCUs used	Safety architecture	Target	Safety analysis result
1	1oo1/1oo1D	SIL2 LD	Achievable
		SIL2 HD/CM	Achievable with potential performance impact(1)
2	1oo2	SIL3 LD	Achievable
		SIL3 HD/CM	Achievable with potential performance impact

1. Note that the potential performance impact related to some above-reported target achievements is mainly related to the need of execution of periodical software-based diagnostics (refer to safety mechanism description for details). The impact is therefore strictly related to how much “aggressive” the system level PST is (see *Section 3.3.1: Assumed safety requirements).*

The resulting relative safety metrics (DC and SFF) and absolute safety metrics (PFH, PFD) are not reported in this section but in the FMEDA snapshot, due to:

- The large number of STM32F4 Series part numbers,
- The possibility to declare non-safety-relevant unused peripherals, and
- The possibility to enable or not the different available safety mechanisms.

The FMEDA snapshot is a static document reporting the safety metrics computed at different detail levels (at microcontroller level and for microcontroller basic functions) for a given combination of safety mechanisms and for a given part number. If FMEDA computation sheet is needed, early contact the local STMicroelectronics sales representative, in order to receive information on expected delivery dates for specific MCU target part number.

Note: *Safety metrics computations are restricted to STM32F4 Series boundary, therefore not including the WDTe, PEv and VMONe (they are described in Section 3.2)*

4.1.1 Safety analysis results customization

The safety analysis executed for STM32F4 Series devices and contained in this Safety Manual considers all microcontroller modules to be safety related, and so able to interfere with the safety function, with no exclusion. This is in line with the conservative approach to be followed during the analysis of a general-purpose microcontroller, in order to be agnostic versus the final application. This means that no microcontroller module has been declared as “safe” as per IEC61508-4, 3.6.8, and therefore all microcontroller modules are included in SFF computations.

In actual end-user applications, not all the STM32F4 Series parts or modules are used for the implementation of the safety function. That can happens under two possible alternative conditions:

- The part is not used at all (disabled).
- The part is used for the implementation of a function that is non-safety related (for example a GPIO line driving a “power-on” signaling led on an electronic board).

Requiring the implementation of the respective safety mechanism for those unused parts could result in an overkill. The safety analysis results can be therefore customized.

The end user can define the selected STM32F4 Series parts as “non-safety related” under the following conditions (end user responsibility):

- Collect rationales and evidences that the parts play no role in safety function implementation.
- Collect rationales and evidences that the parts do not interfere with the safety function during normal operation due to final system design decisions.
- Fulfill the below-reported general condition for the mitigation of the intra-MCU interferences (*Table 128*).

The end user is therefore allowed for “non-safety related” parts to do the following:

- Discard the part contribution from metrics computations in FMEDA;
- Do not implement the related safety mechanisms listed in *Table 126: List of safety mechanisms*.

With regard to SFF computation, this procedure is equivalent to declare “no part/no effect” as per IEC61508-4, 3.6.13 or 14 definition any failure of discarded modules.

4.1.2 General requirements for freedom from interferences (FFI)

A dedicated analysis has highlighted a list of general requirements to be followed in order to mitigate potential interferences between STM32F4 Series internal modules in case of internal failures (freedom from interferences, FFI). These precautions are integral part of the STM32F4 Series safety concept and they can play a relevant role when multiple microcontroller modules are declared as “non-safety related” by the end user as per *Section 4.1.1*.

The requirement for the end user is to implement the safety mechanism listed in *Table 128* (implementation details can be found in Description of hardware and software diagnostics) despite any evaluation about their contribution for the safety metrics computations.

Table 128. List of general requirements for FFI

Diagnostic	Description
FFI_SM_0	Unused peripheral disable
FFI_SM_1	Periodical read-back of interference avoidance registers
BUS_SM_0	Periodical software test for interconnections
NVIC_SM_0	Periodical read-back of configuration registers
NVIC_SM_1	Expected and unexpected interrupt check by application software
DMA_SM_0	Periodical read-back of configuration registers
DMA_SM_2	Information redundancy including sender or receiver identifier on data packet transferred via DMA(1)
DMA_SM_4	DMA transactions awareness(1)
GPIO_SM_0	Periodical read-back of configuration registers

1. To be implemented only if DMA is actually used

4.1.3 Notes on multiple faults scenario

In principle, IEC61508 requires to analyze multiple faults scenarios, therefore restrictions to one fault at time conditions could be not acceptable. Accordingly, the safety analysis for STM32F4 Series took into consideration also multiple faults scenarios. Furthermore, following the spirit of ISO26262 (the reference and state-of-art standard norm for integrated circuit safety analysis) the analysis investigated faults able to “disable” each safety mechanism, in order to individuate mitigation measure for such condition. *Section 3.6* tables report on the “Multiple faults protection” field the associated safety mechanisms needed to correctly manage a multi-fault scenario, including mitigation measures against safety mechanism disable.

It is strongly recommended to include into the safety concept the implementation of such mitigation measures. This is more relevant for long-term operating systems, where error accumulation issues must be considered.

4.2 Dependent failures analysis

The analysis of dependent failures is important for microcontrollers. The main sub-classes of dependent failures are the Common Cause Failures (CCF). Their analysis is ruled by the IEC 61508:2 annex E that lists the design requirements to be verified to allow the use of on-chip redundancy for ICs with one common semiconductor substrate. However, annexes E.1 and E.2 apply for HFT=1 while the Annex E.3 must be applied to every on-chip redundancy, intended also in terms of diagnostic implemented on the same silicon.

As there are no on-chip redundancy on STM32F4 Series devices, the CCF quantification through the BetaIC computation method is not required. Note that in the case of 1oo2 safety architecture implementation, the end user is required to evaluate the parameter βD, which is the measure of the common-cause between the two channels used in PFH computation.

The STM32F4 Series device architecture and structures can be potential sources of dependent failures. These are analyzed in the following sections. The referred safety mechanisms are described in detail in *Section 3.6: Description of hardware and software diagnostics*.

4.2.1 Power supply

Power supply is a potential source of dependent failures, because any alteration of the power the supply can affect many parts, leading to not-independent failures. The following safety mechanisms address and mitigate those dependent failures:

- VSUP_SM_1: detection of abnormal value of supply voltage;
- VSUP_SM_2: the independent watchdog has a different supply source from the digital core of the MCU, and this diversity helps to mitigate dependent failures related to the main supply alterations.

The adoption of such safety mechanisms is therefore highly recommended despite their minor contribution to the safety metrics to reach the required safety integrity level. Refer to *Section 3.6.18: Power control* for the detailed safety mechanism descriptions.

4.2.2 Clock

System clocks are a potential source of dependent failures, because alterations in the clock characteristics (frequency, jitter) can affect many parts, leading to not-independent failures. The following safety mechanisms address and mitigate those dependent failures:

- CLK_SM_1: the clock security system is able to detect hard alterations (stop) of system clock and activate the adequate recovery actions.
- CLK_SM_2: the independent watchdog has a dedicated clock source. The frequency alteration of the system clock leads to the watchdog window violations by the triggering routine on the application software, leading to the MCU reset by watchdog.

The adoption of such safety mechanism is therefore highly recommended despite their minor contribution to the safety metrics to reach the required safety integrity level. Refer to *Section 3.6.19: Reset and clock control (RCC) subsystem* for detailed safety mechanisms description.

4.2.3 DMA

DMA is a widely shared resource involved in data transfers operated mainly by all peripherals. Failures of DMA can interfere with the behavior of the system peripherals or application software, leading to non independent failures. The safety mechanisms addressing such dependent failures are the following:

- DMA_SM_0,
- DMA_SM_1,
- DMA_SM_2.

The adoption of such safety mechanisms is therefore highly recommended. It is worth to note that if DMA is not used for data transfer, then only DMA_SM_0 must be implemented. Refer to *Section 3.6.6: Direct memory access controller (DMA)* for detailed safety mechanisms description.

4.2.4 Internal temperature

The abnormal increase of the internal temperature is a potential source of dependent failures, because it can affect many MCU parts and therefore lead to not-independent failures. The safety mechanism to be used to mitigate this potential effect is the following:

- VSUP_SM_3: the internal temperature read and check allow the user to quickly detect potential risky conditions before they lead to a series of internal failures. Refer to *Section 3.6.18: Power control* for the detailed safety mechanism descriptions.

5 List of evidences

The Safety Case Database stores all the informations related to the safety analysis performed to derive the results and conclusions reported in this Safety Manual.

In detail, the Safety Case Database is composed of the following:

- Safety Case with the full list of all safety analysis related documents
- ST internal FMEDA tool database for the computation of the safety metrics, including the estimated and measured values
- Safety Report, the document that describes in detail the safety analysis executed on STM32F4 Series devices and the clause-by-clause compliance to IEC 61508
- ST internal fault injection campaign database including tools configuration and settings, fault injection logs and results

Due to presence of ST confidential information, above-described contents are not publicly available and are available for possible competent bodies audit and inspections. This is in line with the clarification of Note 2 on IEC61508:2, 7.4.9.7 requirement.

Appendix A Change impact analysis for other safety standards

The safety analysis reported in this Safety Manual is executed according to IEC 61508 safety norm. This appendix reports the outcomes of a change impact analysis with respect to different safety standards. The following topics are considered for each addressed new safety standard:

- Differences in the suggested hardware architecture (architectural categories), and how to map to safety architectures of IEC 61508.
- Differences in the safety integrity level definitions and metrics computation methods, and how to recompute and judge the safety performances of STM32F4 Series devices according to the new standard.
- Work products required by the new safety norms, and how to remap or rework (if needed) the existing work products resulting as output of the IEC 61508 compliance activity.

The safety standards examined within this change impact analysis are the followings:

- ISO 13849-1:2006, ISO 13849-2:2010 – Safety of machinery and Safety-related parts of control systems
- IEC 62061:2005/AMD1:2012 – Safety of machinery and Functional safety of safety-related electrical, electronic and programmable electronic control systems
- IEC 61800-5-2:2007, ed.1.0 – Adjustable speed electrical power drive systems – Part 5-2: Safety requirements – Functional
- ISO 26262:2010 – Road vehicles - Electrical or electronic (EE) systems.

A.1 ISO 13849-1 / ISO 13849-2

The ISO 13849-1 is a type B1 standard. It provides a guideline for the development of safety-related parts of machinery control systems (SRP or CS) including programmable electronics, hardware and software.

A.1.1 Architectural categories

The section §6.2 of ISO 13849 identifies five categories for the basic parameters, DC, MTTFd and CCF, reflecting the expected resistance to faults of SRP or CS under design and needed for achieving the required PLr. For each category, the standard suggests a typical architecture that meets the related requirements.

Considering ISO 13849 architectural categories defined in §6.2 and focusing on microcontrollers, *Table 129* presents a summary for end users willing to develop Logic Solver units suitable for safety critical channels and performing a defined safety function.

The assumptions are listed hereafter:

1. The safety function is realized by combining in series the elements (SRP or CS) input system, signal processing unit, output system.
2. The SRP or CSs elements may be assigned to one or different categories and different PLs.
3. The safety function is completely in the scope of the end user application.
4. The STM32F4 Series MCUs with the adoption of safety mechanism described in this Safety Manual as single compliant item is by itself suitable for CM application up to PLd (equivalent to SIL2).

The ISO 13849 architectural categories for Logic Solver are shown in *Table 129*.

Table 129. ISO 13849 architectural categories

Cat.	Ref. §	Summary	Designated architecture of Logic	Block diagram
B	6.2.3	The main category; occurrence of one fault can lead to the loss of the safety function. No need of DC and CCF (usually single channel), MTTFd is low or medium. Highest achievable is PL = b	Single channel architecture, one MCU in 1oo1 Refer to *Section 3* Compliant item's MTTFd = high	*Figure 6*
1	6.2.4	Enforcing category B requirements by adopting solutions based on "well tried components" for safety critical application and "well tried" safety principles. A microprocessor is not classified as a "well tried" component. No need of DC and CCF (usually single channel), MTTFd is high. Highest achievable is PL = c.	Single channel architecture, one MCU in 1oo1 Refer to *Section 3* Compliant item's MTTFd = high	*Figure 6*
2	6.2.5	With respect to category 1, it is expected to include in the architecture a test equipment performing checks on the safety function and reporting its loss. Overall DC is low, CCF must be evaluated, MTTFd can range from low to high allowing up to PL = d.	Single channel architecture, one MCU in 1oo1d Refer to *Section 3* Compliant item's MTTFd = high TE is in charge of end user, PL = d	*Figure 7*
3	6.2.6	With respect to category 1, it is expected to have fault detection mechanisms and any single fault occurrence does not lead the loss of the safety function.Overall DC is low, CCF must be evaluated for the channels, MTTFd can range from low to high allowing up to PL = d	Double channel architecture, two identical MCUs in 1oo2 Refer to *Section 3* Continuous testing or monitoring Compliant item's MTTFd = high	*Figure 8*
4	6.2.7	With respect to category 1, it is expected to have fault detection mechanisms and any single fault occurrence does not lead the loss of the safety function. Overall DC is high, CCF must be evaluated for the channels, MTTFd is high allowing PL = e	Double channel architecture, two identical MCUs in 1oo2 Refer to *Section 3* Continuous testing or monitoring Compliant item's MTTFd = high PLe achievable	*Figure 8*

Figure 6. Block diagram for ISO 13849 Cat. B and Cat. 1

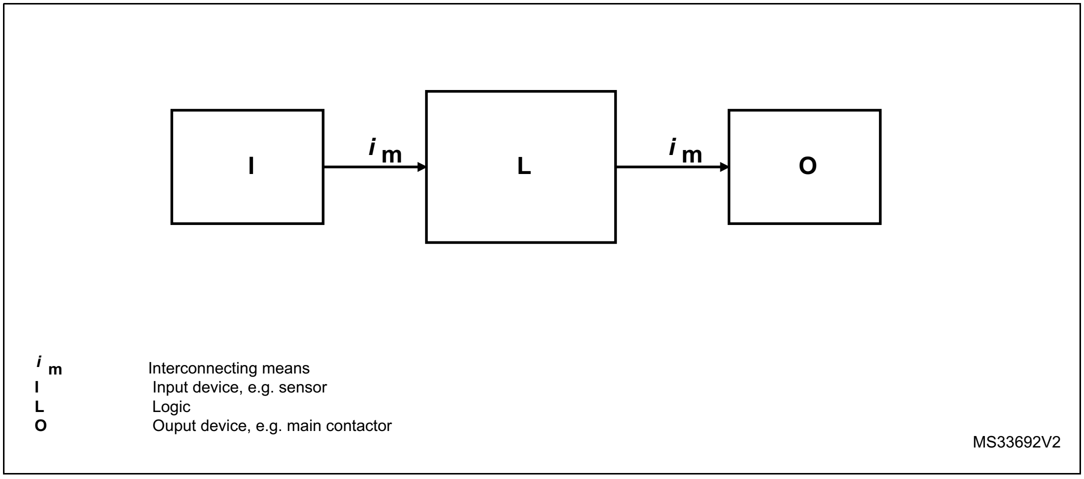


Figure 7. Block diagram for ISO 13849 Cat. 2

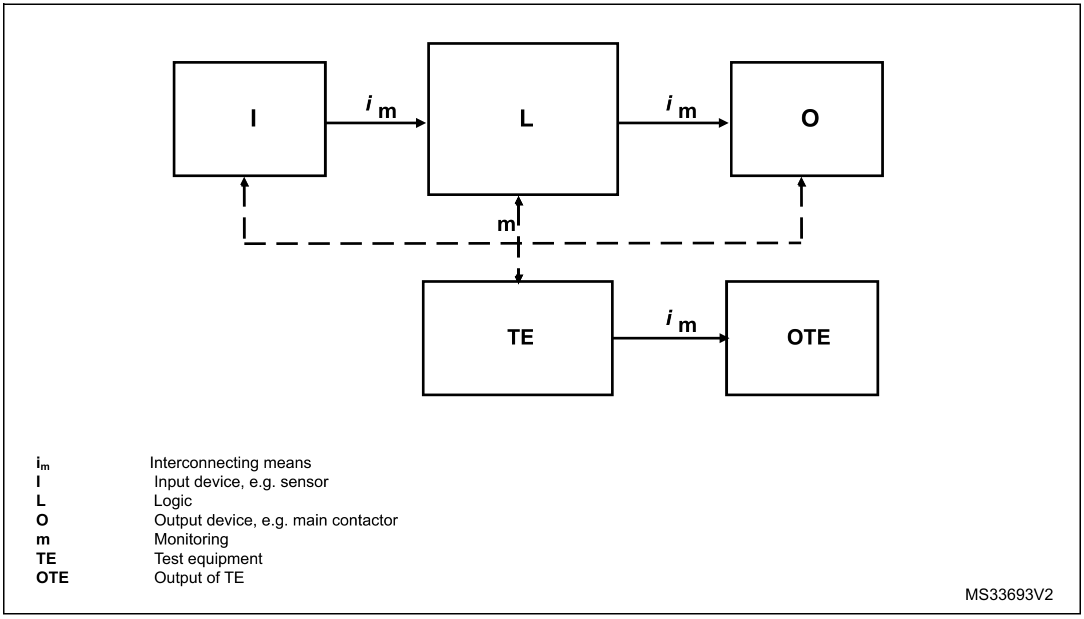
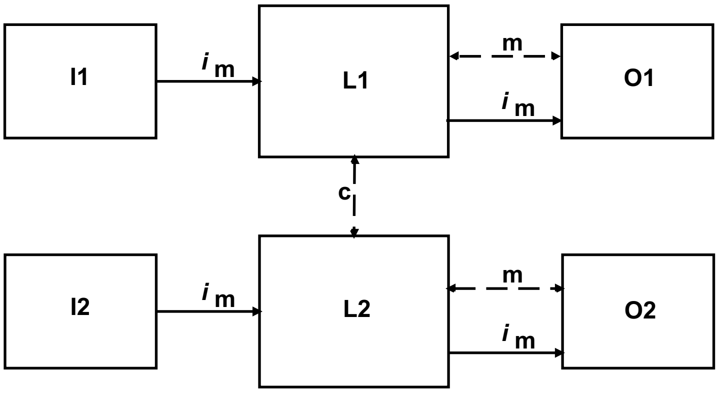


Figure 8. Block diagram for ISO 13849 Cat. 3 and Cat. 4



Im	Interconnecting means
I1, I2	Input device, e.g. sensor
L1, L2	Logic
O1, O2	Output device, e.g. main contactor
m	Monitoring
cm	Cross monitoring

MS33694V1

A.1.2 Safety metrics computation

Appendix C of ISO 13849 presents tables of standardized MTTFd for the various electric or electronics components. However, table C.3 in ISO 13849 points to ICs manufacturer's data while attempting to classify MTTFd for programmable ICs. As a consequence, safety analysis results of this Safety Manual can be re-mapped in ISO 13849 domain, because even computed for IEC 61508 they are definitely more and more accurate in the definition of dangerous failures identification.

When for a certain component PFH << 1 we can assume that MTTFd = 1 / PFH [years].

From the reliability theory, MTTF (the inverse of λ and PFH) is a metric applicable only to not reparable systems. Nowadays it is a common practice to use MTBF also for not reparable systems where MTBF has to be understood as the average time for the first (and only) failure of the equipment; in this case MTBF is equal to MTTF.

In ISO 13849-1 the DC for each single component has the same meaning of the IEC 61508 metric; results of this Safety Manual can therefore be reused. However, this standard defines the concept of DC_{avg} applicable to the whole SRP or CS in the form of the equation defined in Annex E, formula E.1, where the contribution of each part of the control system is weighted with respect to MTTF of the various subsystems of the channel. The standard denies any possibility of fault exclusion while calculating DC_{avg} (ISO13849-2 Tab.D.21 no exclusion allowed) and this is the same assumption done in STM32F4 Series analysis in this Safety Manual.

It is necessary to calculate the DC_{avg} only for subsystem made of a 2 MCUs architecture by applying the formula:

$$DC_{avg} = \frac{\dfrac{DC_{MCU1}}{MTTF_{MCU1}} + \dfrac{DC_{MCU2}}{MTTF_{MCU2}}}{\dfrac{1}{MTTF_{MCU1}} + \dfrac{1}{MTTF_{MCU2}}}$$

For two identical MCUs having the same DC and MTTF, DC_{avg} = DC.

Note: *An evaluation of the possible common failure modes is required for any architectural solution implemented with two channels. ISO 13849 defines a simplified approach with respect to IEC 61508 approach.*

Table 7 of the ISO 13849 standard provides a simplified procedure for PL evaluation of SRP or CS based on category, DC_{avg} and MTTFd. It is worth to note that each architectural solution analyzed in this Safety Manual results in PFH-values producing high values of MTTF.

A.1.3 Work products

Table 130 lists the work products required by the ISO 13849, and how to map these into available work products from IEC 61508 compliance activity:

Table 130. ISO 13849 work product grid

ISO 13849-1		**STM32F4 Series IEC 61508 document**
Information to be provided	**ISO 13849-1 Part-Clause**	
Safety functions provided by the SRP or CS	10 Technical documentation	End user responsibility
Characteristics of each safety function		
Exact points at which the safety-related part(s) start and end		
Environmental conditions		
Performance level (PL)		
Category or categories selected		
Parameters relevant to the reliability (MTTFd, DC, CCF and mission time)	10 Technical documentation	STM32F4 Series Safety Manual and FMEA
Measures against systematic failure		
Technology or technologies used;		
All safety-relevant faults considered		
Justification for fault exclusions (see ISO 13849-2)	10 Technical documentation	End user responsibility
Design rationale (e.g. faults considered, faults excluded)	10 Technical documentation	STM32F4 Series Safety Manual
Measures against reasonably foreseeable misuse		
Dated reference to this part of ISO 13849 (that is “ISO 13849-1:2006”);	11 Information for use	
Category (B, 1, 2, 3, or 4)		
Performance level (a, b, c, d, or e)		
Use of de-energization (see ISO 13849-2)	G.2 Measures for the control of systematic failures	
Measures for controlling the effects of voltage breakdown, voltage variations, overvoltage, under voltage		

Table 130. ISO 13849 work product grid (continued)

ISO 13849-1		STM32F4 Series IEC 61508 document
Information to be provided	**ISO 13849-1 Part-Clause**	
Measures for controlling or avoiding the effects of the physical environment (for example, temperature, humidity, water, vibration, dust, corrosive substances, electromagnetic interference and its effects)	G.2 Measures for the control of systematic failures	End user responsibility
Program sequence monitoring must be used with SRP or CS containing software to detect defective program sequences		
Measures for controlling the effects of errors and other effects arising from any data communication process (see IEC 61508-2:2000, 7.4.8)		
Failure detection by automatic tests	G.2 Measures for the control of systematic failures	STM32F4 Series Safety Manual
Computer-aided design tools capable of simulation or analysis	G.3 Measures for avoidance of systematic failures	End user responsibility
Simulation	-	
Safety-related specification for machine control	App. J, tab.J.1 (SW)	End user responsibility
Definition of the control architecture		
Software descriptions	App. J, tab.J.1 (SW)	Software User Guide (End user responsibility because in charge of implementing software-based diagnostics)
Function block modeling	App. J, tab.J.1 (SW)	SW requirements specification (End user responsibility because in charge of implementing software-based diagnostics)
Encoding comments in the code	App. J, tab.J.1 (SW)	Code inspection results (End user responsibility because in charge of implementing software-based diagnostics)
Encoding re-reading sheets		

Table 130. ISO 13849 work product grid (continued)

ISO 13849-1		STM32F4 Series IEC 61508 document
Information to be provided	**ISO 13849-1 Part-Clause**	
Correspondence matrix	App. J, tab.J.1 (SW)	Software module test specification Software system integration test specification Programmable electronic hardware and software integration tests specification (End user responsibility because in charge of implementing software-based diagnostics)
Test sheets	App. J, tab.J.1 (SW)	Software module test report Software system integration test report Programmable electronic hardware and software integration tests report SW verification report (End user responsibility because in charge of implementing software-based diagnostics)

A.2 IEC 62061:2005/AMD1:2012

This standard is applicable in the specification, design and verification or validation of Safety-Related Electrical Control Systems (SRECS) of machines. SRECS is the electrical or electronics control system of the machine which failure could lead to reduction or loss of safety. SRECS implements a Safety-Related Control Function (SRCF) to prevent any increase of the risk.

With respect of the safety lifecycle, the scope of this standard is limited from safety requirements allocation to safety validation.

IEC 62061 is the special standard for the machine domain within the framework of the more generic IEC 61508:2010. Since it is just an application standard, IEC 62061 is not strict with respect to the technical solutions. Moreover it is focused on electrical, electronic and programmable electronic parts of safety-related control systems.

Note that §3.2.26 and §3.2.27 in IEC 62061 apply only to SRECS in HD or CM, suitable for the machines domain. LD equipment are still ruled by IEC 61508 requirements.

The close relationship with IEC 61508:2010 is synthesized by the main assumption that the design of complex electronic components as subsystems or elements of subsystems has to be compliant with requirements of IEC 61508:2010 part 2, Route 1H, ref. to §7.4.4.2. Coming from the IEC 62061 definition §3.2.8, natively a microprocessor has to be considered as a complex component.

For this reason, the results reported in this Safety Manual for the STM32F4 Series item (refer to *Section 4: Safety results*), in the scope of IEC 61508 are still applicable also in the machines context ruled by IEC 62061.

End-users can effectively adopt the STM32F4 Series compliant item to design SRECS suitable for the achievement of SIL2 or SIL3 (by adopting two STM32F4 Series MCUs) machines control loops.

The standard defines as “subsystem” (refer to §3.2.5) the level of parts for a system architecture where a dangerous failure could lead to the loss of the safety function.

Concerning the integrity levels achievable for subsystems, the standard suggests a classification based on HFT and SFF as shown in *Table 131*.

Table 131. SIL classification versus HFT

SFF	HFT		
	0	1	2
< 60%	Not allowed	SIL1	SIL2
60% - < 90%	SIL1	SIL2	SIL3
90% - < 99%	SIL2	SIL3	SIL3
≥ 99%	SIL3	SIL3	SIL3

SIL 3 is the highest requirement for SRCF in this context. SIL 4 is out of scope since the final outcome of the development is a control system for one machine only.

For the designer, the SIL values listed in the table has to be seen as the SILCL for the subsystem where SILCL is the maximum SIL claimable for a SRECS subsystem, as defined in IEC 62061, §3.2.24.

A.2.1 Architectural categories

The standard in §6.7.8.2 defines a set of basic system architectures to be used for the design of SRECS implementing their SRCFs. A key point is the definition of “subsystem”, refer to §3.2.5, as the level of parts for a system architecture where a dangerous failure could lead to the loss of the safety function.

Focusing on the microcontrollers, IEC 62061 proposed architectures are here quickly summarized for supporting end users in the development of their Logic Solver units usable as subsystems for the implementation of a SRCF.

The assumptions for the correct understanding of the architectures are listed hereafter:

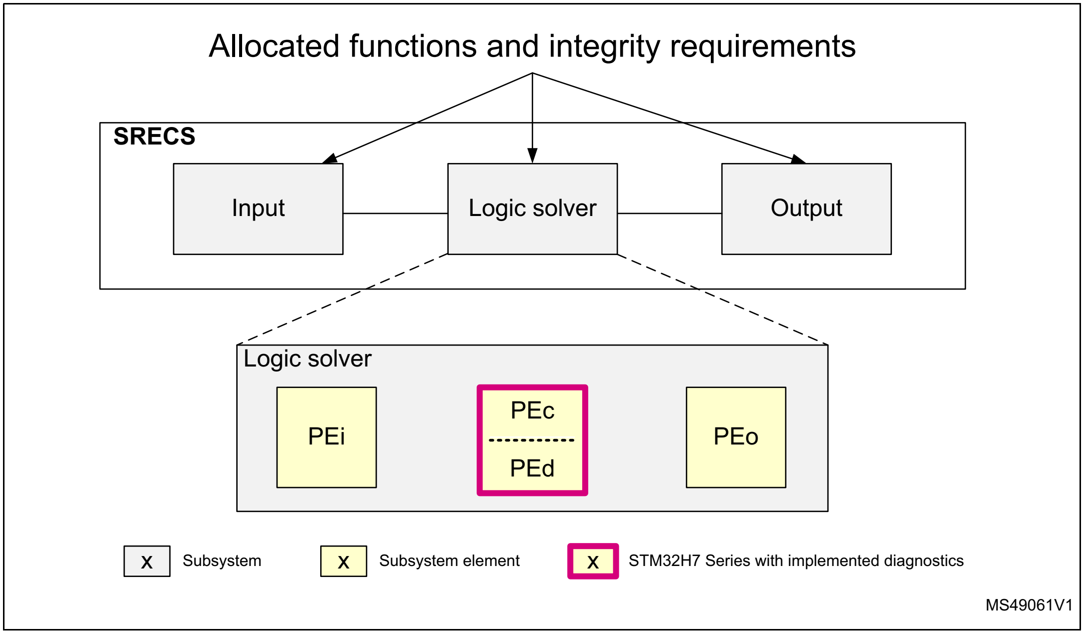
1. The SRCF is completely in the scope of the end user.
2. The STM32F4 Series device with the adoption of safety mechanism described in this Safety Manual as single compliant item is by itself suitable for applications up to SILCL 2.
3. Two identical STM32F4 Series devices with the adoption of safety mechanism described in this Manual must be used for achieving HFT ≠ 0, when required by basic architectures.
4. For a microcontroller, the parameter T1, mentioned in the standard as the minimum between service life or proof test, is intended as the lifetime (mission time) assumed equal to 10 years, as per *Section 3.3.1: Assumed safety requirements* of this Manual.

Table 132. IEC 62061 architectural categories

Cat.	Ref. §	Summary	Basic architecture of Logic
A	6.7.8.2.2	Equivalent of 1oo1, with HFT = 0, no diagnostic function(s). Overall PFH_{DssA} is the probability of dangerous failure of MCU	Single channel architecture, one MCU in 1oo1, n=1 $PFH_{DSSA} = \lambda_{De1}\left[\frac{1}{Hours}\right]$ – SILCL = 1 if SFF < 90% – SILCL = 2 if 90 ≤ SFF < 99% – SILCL = 3 if SFF ≥ 99%
B	6.7.8.2.3	Equivalent to 1oo2 with HFT = 1, a single failure does not lead to the loss of SRCF. No diagnostic function(s).	Dual channel architecture with two identical MCUs – SILCL = 1 if SFF < 60% – SILCL = 2 if 60% ≤ SFF < 90% – SILCL = 3 if SFF ≥ 90% In this case: $\lambda_{De1} = \lambda_{De2} = \lambda_{De}$ $\lambda_{DSSB} = (1-\beta)^2 \times \lambda_{De}^2 \times T_1 + \beta \times \lambda_{De}$ For β factor see *Section 4.2*
C	6.7.8.2.4	It is the equivalent of 1oo1d with a diagnostic function that initiates a reaction function as a dangerous failure happens on SRCF. NOTE: diagnostic function provides the Logic Solver with a diagnosis of an external subsystem, e.g. the actuator	Single channel architecture, one MCU in 1oo1, n=1 Diagnostic function is in charge of end user – SILCL = 1 if SFF < 90% – SILCL = 2 if 90 < SFF < 99% – SILCL = 3 if SFF ≥ 99% $\lambda_{DSSC} = \lambda_{De1}(1 - DC_1)$ DC (Diagnostic Coverage) as resulting from FMEDA $PFH_{DSSC} = \lambda_{DSSC}\left[\frac{1}{Hours}\right]$
D	6.7.8.2.5	Any single failure does not lead to a loss of the SRCF; it is equivalent to 1oo2d with HFT = 1, with diagnostic function(s). NOTE: diagnostic function provides the Logic Solver with a diagnosis of an external subsystem, e.g. the actuator	Dual channel architecture with two identical MCUs Diagnostic function is in charge of end user – SILCL = 1 if SFF < 60% – SILCL = 2 if 60% ≤ SFF < 90% – SILCL = 3 if SFF ≥ 90% For β factor see *Section 4.2* DC (Diagnostic Coverage) as resulting from FMEDA In this case: – $\lambda_{De1} = \lambda_{De2} = \lambda_{De}$ – T2 has to be defined at Logic Solver level by end user

Based on IEC 62061 §6, *Figure 9* shows how to proceed with the development of SRECS implementing the generic control architecture depicted in figure B.1 of the standard where the microprocessor here presented is an STM32F4 Series device with the adoption of the safety mechanisms as defined in *Section 3.7: Conditions of use*.

Figure 9. SRECS high-level diagram



A.2.2 Safety metrics computation

The failure rate (λ) in T is the smaller proof test interval or the life time of the subsystem.

As seen in ISO13849, the approximation §6.7.8.2.1 NOTE2 is still considered valid, hence λ = 1 / MTTF, where it is assumed that 1 >> λ x T.

So, as $PFH_D = \lambda_D \times 1h$, it will be $PFH_D = 1 / MTTF$.

Safety analysis executed for STM32F4 Series according IEC 61508 is more and more accurate for the definition of dangerous failure identifications that can be re-mapped in IEC 62061 domain. Thus, the values of λ and PFH reported in the FMEDA (refer to *Section 4: Safety results*) are still valid, and can be used in the formulas of the previous paragraph.

There is no need for re-computation for the SFF of a microcontroller. The end user uses the same value resulting from this Safety Manual.

As previously discussed in *Section 4.2: Dependent failures analysis*, in evaluating CCF for those basic architectures with an HFT = 1, the end user uses the same result, if available, as achieved by the IEC 61508 approach (refer to IEC 61508:2010-6 Annex D). Alternatively, the end user can apply the simplified approach from the standard (refer to Annex F) to calculate the β factor value to be used in formulas for PFHD.

A.2.3 Work products

Table 133 lists the work products required by the IEC 62061 standard and their mapping with the work products from IEC 61508 compliance activity:

Table 133. IEC 62061 work product grid

IEC 62061 1.1 Tab.8		STM32F4 Series IEC 61508 document
Information to be provided	**IEC 62061-1.1 Clause**	
Functional safety plan	4.2.1	End user responsibility
Specification of requirements for SRCFs	5.2	
Functional safety requirements specification for SRCFs	5.2.3	
Safety integrity requirements specification for SRCFs	5.2.4	
SRECS design	6.2.5	STM32F4 Series Safety Manual
Structured design process	6.6.1.2	End user responsibility
SRECS design documentation	6.6.1.8	
Structure of function blocks	6.6.2.1.1	
SRECS architecture	6.6.2.1.5	STM32F4 Series Safety Manual
Subsystem safety requirements specification	6.6.2.1.7	End user responsibility
Subsystem realization	6.7.2.2	
Subsystem architecture (elements & their interrelationships)	6.7.4.3.1.2	STM32F4 Series Safety Manual
Fault exclusions claimed when estimating fault tolerance or SFF	6.7.6.1c / 6.7.7.3	End user responsibility
Software safety requirements specification	6.10.1	
Software based parameterization	6.11.2.4	
Software configuration management items	6.11.3.2.2	
Suitability of software development tools	6.11.3.4.1	
Documentation of the application program	6.11.3.4.5	
Results of application software module testing	6.11.3.7.4	
Results of application software integration testing	6.11.3.8.2	
Documentation of SRECS integration testing	6.12.1.3	
Documentation of SRECS installation	6.13.2.2	
Documentation for installation, use and maintenance	7.2	
Documentation of SRECS validation testing	8.2.4	
Documentation for SRECS configuration management	9.3.1	

A.3 IEC 61800-5-2:2007

The scope of this standard is the functional safety of adjustable speed electric drive systems. Part 5.2 of the IEC 61800 defines the requirements for the design, development, integration and validation of the safety related parts for power drive speed applications, PDS(SR), within the framework of IEC 61508 first edition. More precisely, this part of IEC 61800 just limits its application to those PSD(RS) operating in HD or CM, referring to §3.10 NOTE1, implementing safety functions with a target integrity up to SIL 3.

Form the architectural point of view, this limitation is reflected in two tables, §6.2.2.3 Tab. 3 and Tab. 4, for the two different types of classified devices. The CPU or the whole microcontroller, since these are complex electronics parts, is classified as Type B. Also the concept of HFT is derived from IEC 61508 as it is.

A.3.1 Architectural categories

From the architectural point of view, IEC 61800 application is reflected in two tables, §6.2.2.3 Tab. 3 and Tab. 4, for the two different types of classified devices. The CPU or the whole microcontroller, considered as complex electronics parts, are classified as Type B. Also the concept of HFT is derived from IEC 61508 as it is. Architectural remapping on IEC61508 is therefore straightforward.

A.3.2 Safety metrics computation

The PFH of a safety function performed by PDS(SR) is evaluated by the application of IEC 61508-2. The strong link with the norm IEC 61508 is reflected also by the adoption in IEC 61800-5-2 of the same relevant metrics PFH, ref. to §6.2.1, and SFF, ref. to §6.2.3.So, results of this Safety Manual (and related FMEA or FMEDA) can be re-mapped in IEC61800 domain.

A.3.3 Work products

Table 134 lists the work products required by the IEC 61800-5-2 standard and their mapping with the work products from IEC 61508 compliance activity.

Table 134. IEC 61800 work product grid

IEC 618000 5.2		STM32F4 Series IEC 61508 document
Information to be provided	**IEC 61800-5.2 Part-Clause**	
Safety requirements specification (SRS) for PDS(SR) including safety function requirements and safety integrity requirements	5.4	End user responsibility
Verification of PDS(SR) safety requirements specification	8.2	
Hardware design on an architectural level	6	
Software design on an architectural level	IEC 61508-3	
Estimation of the probability of failure of safety functions due to random hardware failures on a level of functional block diagrams	IEC 61508-2	STM32F4 Series Safety Manual and FMEDA

Table 134. IEC 61800 work product grid (continued)

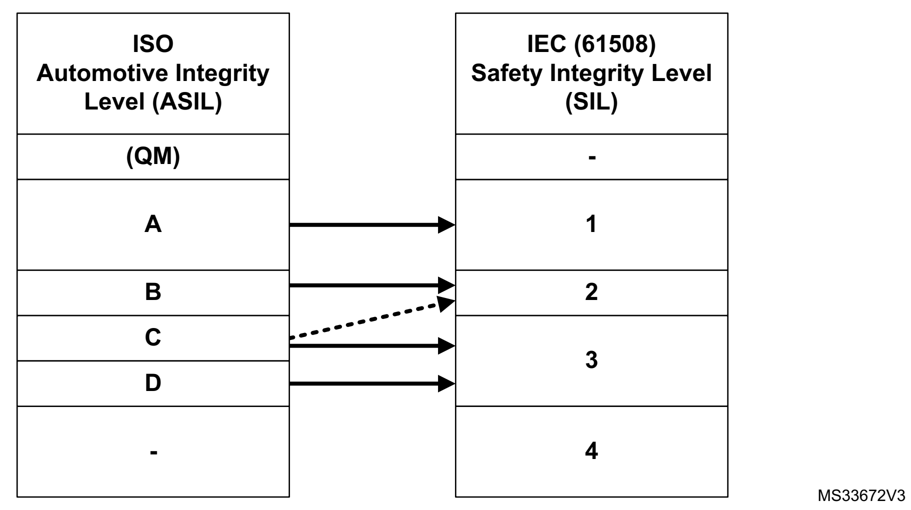
IEC 618000 5.2		STM32F4 Series IEC 61508 document
Information to be provided	**IEC 61800-5.2 Part-Clause**	
Reviews of system design	8.2	End user responsibility
Detailed planning of the validation of safety related PDS(SR).	8.3	
Hardware design	6	
Software design		
Reliability Prediction	6	STM32F4 Series Safety Manual and FMEDA
Reviews of the system design	8.2	End user responsibility
Functional tests on module level		
Integration and test of the safety related PDS(SR).	6.5	
Review of HW or SW integration test results and documentation	8.2	
Develop user documentation describing PDS(SR) installation, commissioning, operation and maintenance.	7	
Complete software and appropriate documentation	8.3	
Documentation of the results of the validation tests		
Validation tests and procedures according to the validation plan		
Documentation of the results of the validation tests		
Subsystem testing plan	6.2.4.1.4	
Integration testing plan		
Validation testing plan		
Configuration testing plan		
Detailed results of each test	9.2.g)	
Any discrepancy between expected and actual results	9.2.h)	
Conclusion of the test: either it has been passed or the reasons for failure	9.2.i)	

A.4 ISO 26262:2010

This international standard is the reference for the functional safety for the automotive domain. It derives from IEC 61508 standard, and includes relevant modifications.

ISO 26262 redefines the safety integrity levels in term of Automotive SIL (ASIL) with a scale from A, the lowest level, to D, the highest level. A correlation matrix between SIL and ASIL values has been empirically identified by TÜV SÜD and is illustrated in *Figure 10*.

Figure 10. Correlation matrix between SIL and ASIL



A.4.1 Architectural categories

Not Applicable - since ISO 26262 does not define any category.

A.4.2 Safety metrics computation

Hardware metrics in ISO 26262 standard have been defined with a slightly different perspective from IEC61508:

- Single Point Fault Metric (SPFm): defined with the same formula of SFF in IEC61508, can differ according to different definition of safe faults (see below)
- Diagnostic Coverage (DC) is defined in the same way of IEC61508;
- Latent Faults Metric (LFm): dedicated ISO26262 safety metrics to evaluate the robustness of the design against faults affecting diagnostic parts. We have no equivalent in IEC61508.

It is worth noting that these failures that are classified in IEC 61508 standard as no-parts/no-effect, in ISO26262 are classified as “safe failures”. As a result, IEC61508 computations for SFF are “conservative” and so using as SPF values taken from STM32F4 Series FMEDA is possible.

For such kind of Commercial Off-the-Shelf (COTS) microcontroller as STM32F4 Series, the natural target in ISO scenario is ASIL B (90% SPF target for permanent and transient, and 60% for latent). As these are the same targets as for 1oo1 SIL2 case, it can be assumed that the same set of conditions of use or safety mechanisms apply. Metrics computations are detailed into the FMEDA for microcontrollers of the STM32F4 Series; note that the resulting PMHF values comply with the expectations for an ASIL B MCU.

We can conclude that the ASIL B target is achievable with some constraints for the final application. Note that safety diagnostic measures based on periodical execution of software are executed at least once each FTTI.

For the STM32F4 Series devices, the fulfillment of ASIL B latent faults metrics (60%) is achievable with the adoption of the same safety mechanism combination that guarantees the microcontroller to be suitable for SIL2 applications.

Note: Due to differences between IEC61508 and ISO26262 interpretation on local targets for microcontroller modules or functions, safety performances achieved by STM32F4 Series in a SIL2 scenario could be not compatible with an ISO26262 application based on ISO26262-5, 9.4.3 section (the so-called 'cut-set' approach). If your ISO26262 safety analysis uses such approach, check carefully STM32F4 Series FMEDA failure rates at function level.

A.4.3 Work products

Table 135 lists the work products required by the ISO 26262 standard and their mapping with the work products from IEC 61508 compliance activity:

Table 135. IEC 26262 work product grid

IEC 26262		STM32F4 Series IEC 61508 document
Information to be provided	**IEC 26262 Part-Clause**	
Technical safety requirements specification	4-6.5.1	STM32F4 SeriesSafety Manual
Technical safety concept	4-7.5.1	
Safety analysis reports resulting from requirement	4-7.5.6	
Hardware safety requirements verification report	5-6.5.3	
Hardware safety analysis report	5-7.5.2	
Analysis of the effectiveness of the architecture of the item to cope with the random hardware failures	5-8.5.1	
Review report of evaluation of the effectiveness of the architecture of the item to cope with the random hardware failures	5-8.5.2	
Analysis of safety goal violations due to random hardware failures	5-9.5.1	
Review report of evaluation of safety goal violations due to random hardware failures	5-9.5.3	STM32F4 Series FMEDA
Software safety requirements specification	6-6.5.1	End user Responsibility
Software architectural design specification	6-7.5.1	
Software verification report (refined)	6-11.5.3	
Results of safety analyses	9-8.5.1	STM32F4 Series Safety Manual, FMEA and FMEDA

Note: STM32F4 Series FMEA should be reworked in order to map IEC61508 reference failure modes into ISO26262 ones.

Revision history

Table 136. Document revision history

Date	Revision	Changes
01-Dec-2014	1	Initial release.
30-Jan-2015	2	Updated the applicability of the user manual to the microcontrollers of the STM32F4 series, and to STM32-SafeSIL part number. Updated: – *Figure 1: STMicroelectronics product development process* – *Figure 16: Block diagram for IEC 62061 Cat. B* – *Figure 18: Block diagram for IEC 62061 Cat. D*
24-Nov-2017	3	Removed: – *former fRMethodology* – *Dual MCU architecture* – *Latent Fault detection* – *examples of safety architecture* – *fRSTL_STM32F0_SIL2(3)* Added: – *Figure 3: 1oo1 reference architecture* – *Figure 4: 1oo2 reference architecture* – *Section 3.6.23: System configuration controller (SYSCFG)* – *Section 3.7: Conditions of use* – *Table 5* to *Table 125* for description of hardware and software diagnostics Updated: – *Section 3.3.1: Assumed safety requirements* – *Section A.4.1: Architectural categories* – *Table 127: Overall achievable safety integrity levels* – *Table 135: IEC 60730 required safety mechanism for Class B/C compliance*
19-Apr-2018	4	Updated: – *Introduction* – *Table 2: Mapping between this document content and IEC 61508-2 Annex D requirements* – *Section 4.2: Dependent failures analysis* – *Section A.2: IEC 62061:2005/AMD1:2012* – *Table 131: SIL classification versus HFT* Deleted: – *Section A.4: IEC 60730-1:2010*