INTERNATIONAL STANDARD

ISO 26262-5

First edition
2011-11-15

Road vehicles — Functional safety —

Part 5: **Product development at the hardware level**

Véhicules routiers — Sécurité fonctionnelle —

Partie 5: Développement du produit au niveau du matériel

Reference number
ISO 26262-5:2011(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 26262-5 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 26262 consists of the following parts, under the general title *Road vehicles — Functional safety*:

- *Part 1: Vocabulary*
- *Part 2: Management of functional safety*
- *Part 3: Concept phase*
- *Part 4: Product development at the system level*
- *Part 5: Product development at the hardware level*
- *Part 6: Product development at the software level*
- *Part 7: Production and operation*
- *Part 8: Supporting processes*
- *Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses*
- *Part 10: Guideline on ISO 26262*

Introduction

ISO 26262 is the adaptation of IEC 61508 to comply with needs specific to the application sector of electrical and/or electronic (E/E) systems within road vehicles.

This adaptation applies to all activities during the safety lifecycle of safety-related systems comprised of electrical, electronic and software components.

Safety is one of the key issues of future automobile development. New functionalities not only in areas such as driver assistance, propulsion, in vehicle dynamics control and active and passive safety systems increasingly touch the domain of system safety engineering. Development and integration of these functionalities will strengthen the need for safe system development processes and the need to provide evidence that all reasonable system safety objectives are satisfied.

With the trend of increasing technological complexity, software content and mechatronic implementation, there are increasing risks from systematic failures and random hardware failures. ISO 26262 includes guidance to avoid these risks by providing appropriate requirements and processes.

System safety is achieved through a number of safety measures, which are implemented in a variety of technologies (e.g. mechanical, hydraulic, pneumatic, electrical, electronic, programmable electronic) and applied at the various levels of the development process. Although ISO 26262 is concerned with functional safety of E/E systems, it provides a framework within which safety-related systems based on other technologies can be considered. ISO 26262:

a) provides an automotive safety lifecycle (management, development, production, operation, service, decommissioning) and supports tailoring the necessary activities during these lifecycle phases;

b) provides an automotive-specific risk-based approach to determine integrity levels [Automotive Safety Integrity Levels (ASIL)];

c) uses ASILs to specify applicable requirements of ISO 26262 so as to avoid unreasonable residual risk;

d) provides requirements for validation and confirmation measures to ensure a sufficient and acceptable level of safety being achieved;

e) provides requirements for relations with suppliers.

Functional safety is influenced by the development process (including such activities as requirements specification, design, implementation, integration, verification, validation and configuration), the production and service processes and by the management processes.

Safety issues are intertwined with common function-oriented and quality-oriented development activities and work products. ISO 26262 addresses the safety-related aspects of development activities and work products.

Figure 1 shows the overall structure of this edition of ISO 26262. ISO 26262 is based upon a V-model as a reference process model for the different phases of product development. Within the figure:

- the shaded “V”s represent the interconnection between ISO 26262-3, ISO 26262-4, ISO 26262-5, ISO 26262-6 and ISO 26262-7;

- the specific clauses are indicated in the following manner: “m-n”, where “m” represents the number of the particular part and “n” indicates the number of the clause within that part.

EXAMPLE “2-6” represents Clause 6 of ISO 26262-2.

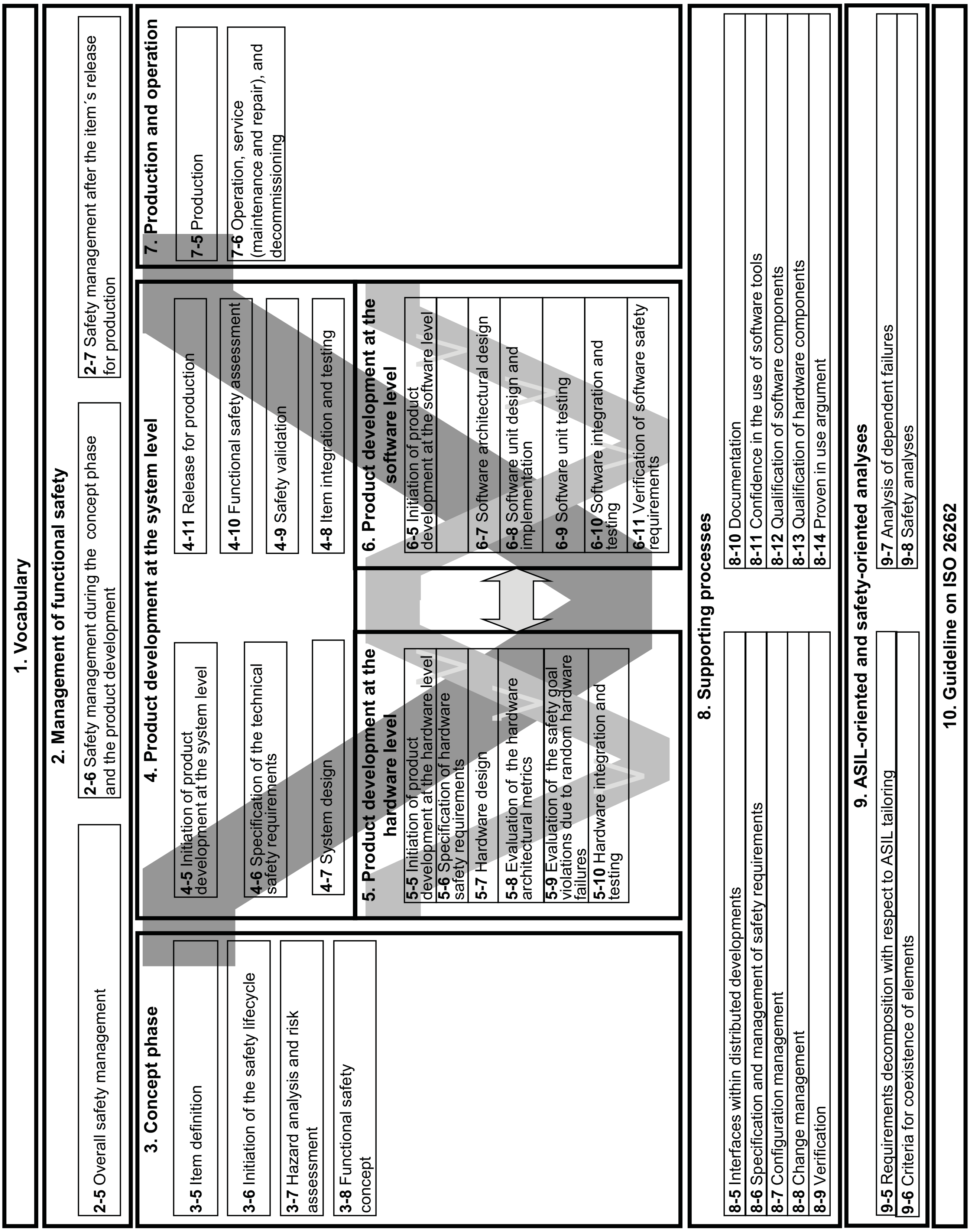

1. Vocabulary
2. Management of functional safety
2-5 Overall safety management
2-6 Safety management during the concept phase and the product development
2-7 Safety management after the item´s release for production
3. Concept phase
3-5 Item definition
3-6 Initiation of the safety lifecycle
3-7 Hazard analysis and risk assessment
3-8 Functional safety concept
4. Product development at the system level
4-5 Initiation of product development at the system level
4-6 Specification of the technical safety requirements
4-7 System design
4-8 Item integration and testing
4-9 Safety validation
4-10 Functional safety assessment
4-11 Release for production
5. Product development at the hardware level
5-5 Initiation of product development at the hardware level
5-6 Specification of hardware safety requirements
5-7 Hardware design
5-8 Evaluation of the hardware architectural metrics
5-9 Evaluation of the safety goal violations due to random hardware failures
5-10 Hardware integration and testing
6. Product development at the software level
6-5 Initiation of product development at the software level
6-6 Specification of software safety requirements
6-7 Software architectural design
6-8 Software unit design and implementation
6-9 Software unit testing
6-10 Software integration and testing
6-11 Verification of software safety requirements
7. Production and operation
7-5 Production
7-6 Operation, service (maintenance and repair), and decommissioning
8. Supporting processes
8-5 Interfaces within distributed developments
8-6 Specification and management of safety requirements
8-7 Configuration management
8-8 Change management
8-9 Verification
8-10 Documentation
8-11 Confidence in the use of software tools
8-12 Qualification of software components
8-13 Qualification of hardware components
8-14 Proven in use argument
9. ASIL-oriented and safety-oriented analyses
9-5 Requirements decomposition with respect to ASIL tailoring
9-6 Criteria for coexistence of elements
9-7 Analysis of dependent failures
9-8 Safety analyses
10. Guideline on ISO 26262

Figure 1 — Overview of ISO 26262

Road vehicles — Functional safety —

Part 5: Product development at the hardware level

1 Scope

ISO 26262 is intended to be applied to safety-related systems that include one or more electrical and/or electronic (E/E) systems and that are installed in series production passenger cars with a maximum gross vehicle mass up to 3 500 kg. ISO 26262 does not address unique E/E systems in special purpose vehicles such as vehicles designed for drivers with disabilities.

Systems and their components released for production, or systems and their components already under development prior to the publication date of ISO 26262, are exempted from the scope. For further development or alterations based on systems and their components released for production prior to the publication of ISO 26262, only the modifications will be developed in accordance with ISO 26262.

ISO 26262 addresses possible hazards caused by malfunctioning behaviour of E/E safety-related systems, including interaction of these systems. It does not address hazards related to electric shock, fire, smoke, heat, radiation, toxicity, flammability, reactivity, corrosion, release of energy and similar hazards, unless directly caused by malfunctioning behaviour of E/E safety-related systems.

ISO 26262 does not address the nominal performance of E/E systems, even if dedicated functional performance standards exist for these systems (e.g. active and passive safety systems, brake systems, Adaptive Cruise Control).

This part of ISO 26262 specifies the requirements for product development at the hardware level for automotive applications, including the following:

- requirements for the initiation of product development at the hardware level,
- specification of the hardware safety requirements,
- hardware design,
- hardware architectural metrics, and
- evaluation of violation of the safety goal due to random hardware failures and hardware integration and testing.

The requirements of this part of ISO 26262 for hardware elements are applicable both to non-programmable and programmable elements, such as ASIC, FPGA and PLD. Furthermore, for programmable electronic elements, requirements in ISO 26262-6, ISO 26262-8:2011, Clause 11, and ISO 26262-8:2011, Clause 12, are applicable.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 26262-1:2011, *Road vehicles — Functional safety — Part 1: Vocabulary*

ISO 26262-2:2011, *Road vehicles — Functional safety — Part 2: Management of functional safety*

ISO 26262-4:2011, *Road vehicles — Functional safety — Part 4: Product development at the system level*

ISO 26262-6:2011, *Road vehicles — Functional safety — Part 6: Product development at the software level*

ISO 26262-7:2011, *Road vehicles — Functional safety — Part 7: Production and operation*

ISO 26262-8:2011, *Road vehicles — Functional safety — Part 8: Supporting processes*

ISO 26262-9:2011, *Road vehicles — Functional safety — Part 9: Automotive Safety Integrity Level (ASIL)-oriented and safety-oriented analyses*

3 Terms, definitions and abbreviated terms

For the purposes of this document, the terms, definitions and abbreviated terms given in ISO 26262-1:2011 apply.

4 Requirements for compliance

4.1 General requirements

When claiming compliance with ISO 26262, each requirement shall be complied with, unless one of the following applies:

a) tailoring of the safety activities in accordance with ISO 26262-2 has been planned and shows that the requirement does not apply, or

b) a rationale is available that the non-compliance is acceptable and the rationale has been assessed in accordance with ISO 26262-2.

Information marked as a “NOTE” or “EXAMPLE” is only for guidance in understanding, or for clarification of the associated requirement, and shall not be interpreted as a requirement itself or as complete or exhaustive.

The results of safety activities are given as work products. “Prerequisites” are information which shall be available as work products of a previous phase. Given that certain requirements of a clause are ASIL-dependent or may be tailored, certain work products may not be needed as prerequisites.

“Further supporting information” is information that can be considered, but which in some cases is not required by ISO 26262 as a work product of a previous phase and which may be made available by external sources that are different from the persons or organizations responsible for the functional safety activities.

4.2 Interpretations of tables

Tables are normative or informative depending on their context. The different methods listed in a table contribute to the level of confidence in achieving compliance with the corresponding requirement. Each method in a table is either

a) a consecutive entry (marked by a sequence number in the leftmost column, e.g. 1, 2, 3), or

b) an alternative entry (marked by a number followed by a letter in the leftmost column, e.g. 2a, 2b, 2c).

For consecutive entries, all methods shall be applied as recommended in accordance with the ASIL. If methods other than those listed are to be applied, a rationale shall be given that these fulfil the corresponding requirement.

For alternative entries, an appropriate combination of methods shall be applied in accordance with the ASIL indicated, independent of whether they are listed in the table or not. If methods are listed with different degrees of recommendation for an ASIL, the methods with the higher recommendation should be preferred. A rationale shall be given that the selected combination of methods complies with the corresponding requirement.

NOTE A rationale based on the methods listed in the table is sufficient. However, this does not imply a bias for or against methods not listed in the table.

For each method, the degree of recommendation to use the corresponding method depends on the ASIL and is categorized as follows:

- "++" indicates that the method is highly recommended for the identified ASIL;
- "+" indicates that the method is recommended for the identified ASIL;
- "o" indicates that the method has no recommendation for or against its usage for the identified ASIL.

4.3 ASIL-dependent requirements and recommendations

The requirements or recommendations of each subclause shall be complied with for ASIL A, B, C and D, if not stated otherwise. These requirements and recommendations refer to the ASIL of the safety goal. If ASIL decomposition has been performed at an earlier stage of development, in accordance with ISO 26262-9:2011, Clause 5, the ASIL resulting from the decomposition shall be complied with.

If an ASIL is given in parentheses in ISO 26262, the corresponding subclause shall be considered as a recommendation rather than a requirement for this ASIL. This has no link with the parenthesis notation related to ASIL decomposition.

5 Initiation of product development at the hardware level

5.1 Objectives

The objective of the initiation of the product development for the hardware is to determine and plan the functional safety activities during the individual subphases of hardware development. This also includes the necessary supporting processes described in ISO 26262-8.

This planning of hardware-specific safety activities is included in the safety plan (see ISO 26262-2:2011, 6.4.3, and ISO 26262-4:2011, 5.4).

5.2 General

The necessary activities and processes needed to develop hardware that meets the safety requirements are planned. Figure 2 illustrates the hardware level product development process steps in order to comply with the requirements of this part of ISO 26262, and the integration of these steps within the ISO 26262 framework.

The necessary activities and processes for the product development at the hardware level include:

— the hardware implementation of the technical safety concept;

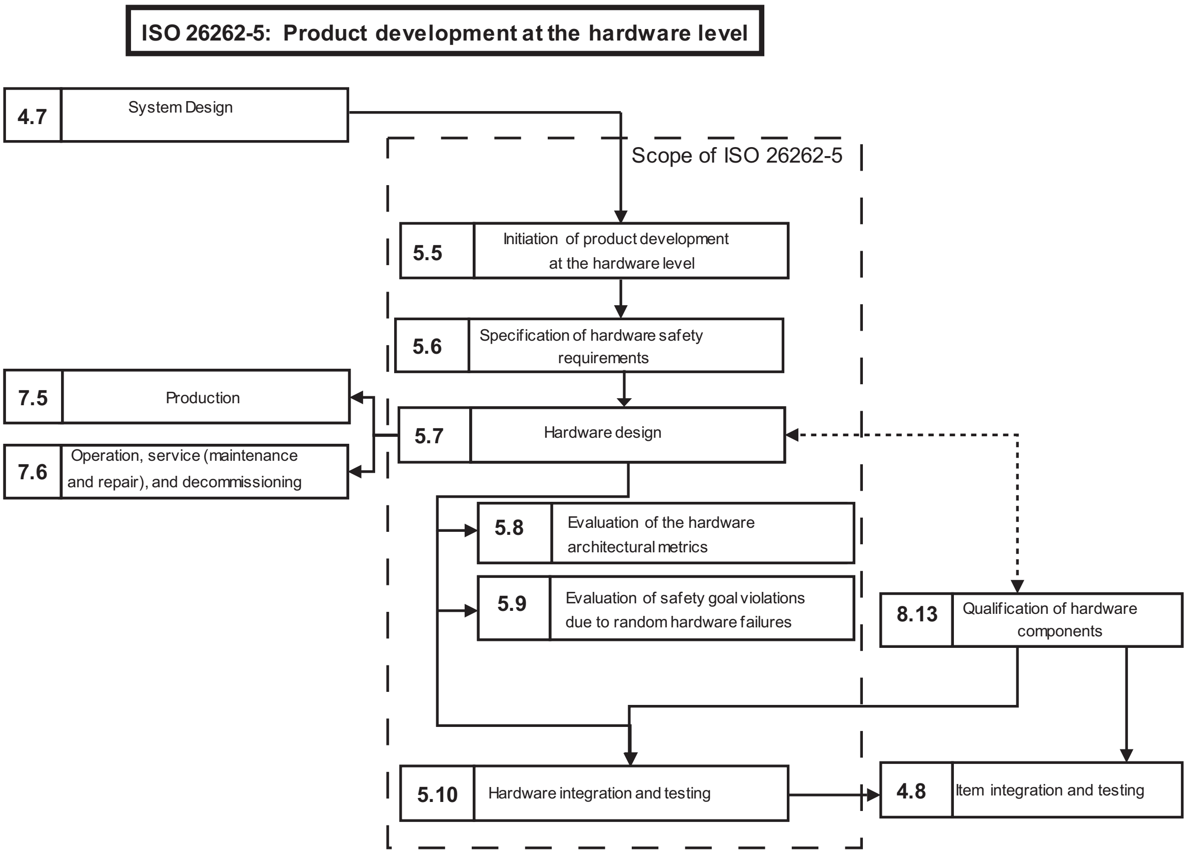
— the analysis of potential hardware faults and their effects; and

— the coordination with software development.

By contrast to the software development subphases, this part of ISO 26262 contains two clauses describing quantitative evaluations of the overall hardware architecture of the item.

Clause 8 describes two metrics to evaluate the effectiveness of the hardware architecture of the item and the implemented safety mechanisms to cope with random hardware failures.

As a complement to Clause 8, Clause 9 describes two alternatives to evaluate whether the residual risk of safety goal violations is sufficiently low, either by using a global probabilistic approach or by using a cut-set analysis to study the impact of each identified fault of a hardware element upon the violation of the safety goals.


ISO 26262-5: Product development at the hardware level
4.7
System Design
Scope of ISO 26262-5
5.5
Initiation of product development at the hardware level
5.6
Specification of hardware safety requirements
7.5
Production
7.6
Operation, service (maintenance and repair), and decommissioning
5.7
Hardware design
5.8
Evaluation of the hardware architectural metrics
5.9
Evaluation of safety goal violations due to random hardware failures
8.13
Qualification of hardware components
5.10
Hardware integration and testing
4.8
Item integration and testing

NOTE Within the figure, the specific clauses of each part of ISO 26262 are indicated in the following manner: “m-n”, where “m” represents the number of the part and “n” indicates the number of the clause, e.g. “4.7” represents Clause 7 of ISO 26262-4.

Figure 2 — Reference phase model for the product development at the hardware level

5.3 Inputs to this clause

5.3.1 Prerequisites

The following information shall be available:

- project plan (refined) in accordance with ISO 26262-4:2011, 5.5.1;
- safety plan (refined) in accordance with ISO 26262-4:2011, 5.5.2; and
- item integration and testing plan (refined) in accordance with ISO 26262-4:2011, 5.5.3.

5.3.2 Further supporting information

The following information can be considered:

- qualification report (of hardware components or parts), if applicable (see ISO 26262-8:2011, 13.5.3).

5.4 Requirements and recommendations

5.4.1 The safety plan in accordance with ISO 26262-2 shall be detailed, including determination of appropriate methods and measures, with respect to the activities for the product development at the hardware level, consistent with the planning of activities in ISO 26262-6.

5.4.2 The hardware development process for the hardware of the item, including methods and tools, shall be consistent across all subphases of the hardware development, and consistent with system and software subphases, so that the requirement flow retains its accuracy and consistency during the hardware development.

5.4.3 The tailoring of the safety lifecycle activities for product development at the hardware level shall be performed in accordance with ISO 26262-2:2011, 6.4.5, and based on the reference phase model given in Figure 2.

5.4.4 The reuse of hardware components, or the use of qualified hardware components or parts, shall be identified and the resulting tailoring of the safety activities shall be described.

5.5 Work products

5.5.1 **Safety plan** (refined) resulting from requirements 5.4.1 to 5.4.4.

6 Specification of hardware safety requirements

6.1 Objectives

The first objective of this clause is to specify the hardware safety requirements. They are derived from the technical safety concept and system design specification.

The second objective is to verify that the hardware safety requirements are consistent with the technical safety concept and the system design specification.

A further objective of this phase is to detail the hardware-software interface (HSI) specification initiated in ISO 26262-4:2011, Clause 7.

6.2 General

The technical safety requirements are allocated to hardware and software. The requirements that are allocated to both are further partitioned to yield hardware only safety requirements. The hardware safety requirements are further detailed, considering design constraints and the impact of these design constraints on the hardware.

6.3 Inputs to this clause

6.3.1 Prerequisites

The following information shall be available:

- safety plan (refined) in accordance with 5.5;
- technical safety concept in accordance with ISO 26262-4:2011, 7.5.1;
- system design specification in accordance with ISO 26262-4:2011, 7.5.2; and
- hardware-software interface specification in accordance with ISO 26262-4:2011, 7.5.3.

6.3.2 Further supporting information

The following information can be considered:

- software safety requirements specification (see ISO 26262-6:2011, 6.5.1).

6.4 Requirements and recommendations

6.4.1 A hardware safety requirements specification for the hardware elements of the item shall be derived from the technical safety requirements allocated to hardware.

6.4.2 The hardware safety requirements specification shall include each hardware requirement that relates to safety, including the following:

NOTE 1 The hardware safety requirements described in bullets a), b), c), or d) include the attributes needed to ensure the effectiveness of the above safety mechanisms.

a) the hardware safety requirements and relevant attributes of safety mechanisms to control internal failures of the hardware of the element, this includes internal safety mechanisms to cover transient faults when shown to be relevant due, for instance, to the technology used;

EXAMPLE 1 Attributes can include the timing and detection abilities of a watchdog.

b) the hardware safety requirements and relevant attributes of safety mechanisms to ensure the element is tolerant to failures external to the element;

EXAMPLE 2 The functional behaviour required for an ECU in the event of an external failure, such as an open-circuit on an input of the ECU.

c) the hardware safety requirements and relevant attributes of safety mechanisms to comply with the safety requirements of other elements;

EXAMPLE 3 Diagnosis of sensors or actuators.

d) the hardware safety requirements and relevant attributes of safety mechanisms to detect and signal internal or external failures; and

NOTE 2 The hardware safety requirements described in bullet d) include safety mechanisms to prevent faults from being latent.

EXAMPLE 4 The specified fault reaction time for the hardware part of a safety mechanism, so as to be consistent with the fault tolerant time interval.

e) the hardware safety requirements not specifying safety mechanisms.

EXAMPLE 5 Examples are:

— requirements on the hardware elements to meet the target values for random hardware failures as described in 6.4.3 and 6.4.4;

— requirements for the avoidance of a specific behaviour (for instance, “a particular sensor shall not produce an unstable output”);

— requirements allocated to hardware elements implementing the intended functionality; and

— requirements specifying design measures on harnesses or connectors.

6.4.3 This requirement applies to ASIL (B), C, and D of the safety goal. The target values specified to comply with ISO 26262-4:2011, Clause 7, for the metrics of Clause 8 of this part of ISO 26262 shall be considered when deriving values for the hardware elements of the item.

NOTE This activity can include a split of target values in the case of a distributed development as given in ISO 26262-8:2011, Clause 5.

6.4.4 This requirement applies to ASIL (B), C, and D of the safety goal. The target values specified to comply with ISO 26262-4:2011, Clause 7, for the procedures of Clause 9 of this part of ISO 26262 shall be considered when deriving values for the hardware elements of the item.

NOTE This activity can include a split of target values in the case of a distributed development as given in ISO 26262-8:2011, Clause 5.

6.4.5 The hardware safety requirements shall be specified in accordance with ISO 26262-8:2011, Clause 6.

6.4.6 The criteria for design verification of the hardware of the item or element shall be specified, including environmental conditions (temperature, vibration, EMI, etc.), specific operational environment (supply voltage, mission profile, etc.) and component specific requirements:

a) for verification by qualification for hardware components or part of intermediate complexity, the criteria shall meet the needs of ISO 26262-8:2011, Clause 13, and

b) for verification by testing, the criteria shall meet the needs of Clause 10.

6.4.7 The hardware safety requirements shall comply with the fault tolerant time interval for safety mechanisms as specified in ISO 26262-4:2011, 6.4.2.3.

6.4.8 The hardware safety requirements shall comply with the multiple-point fault detection interval as specified in ISO 26262-4:2011, 6.4.4.2.

NOTE 1 In the case of ASIL C and D safety goals, and if the corresponding safety concept does not prescribe specific values, the multiple-point fault detection intervals can be specified to be equal or lower than the item's “power-up to power-down” cycle.

NOTE 2 Appropriate multiple-point fault detection intervals can also be justified by the quantitative analysis of the occurrence of random hardware failures (see Clause 9).

6.4.9 The hardware safety requirements shall be verified in accordance with ISO 26262-8:2011, Clauses 6 and 9, in order to provide evidence of their:

a) consistency with the technical safety concept, the system design specification and the hardware specifications;

b) completeness with respect to the technical safety requirements allocated to the hardware element;

c) consistency with the relevant software safety requirements; and

d) correctness and accuracy.

6.4.10 The HSI specification initiated in ISO 26262-4:2011, Clause 7, shall be refined sufficiently to allow for the correct control and usage of the hardware by the software, and shall describe each safety-related dependency between hardware and software.

6.4.11 The persons responsible for hardware and software development shall be jointly responsible for the verification of the adequacy of the refined HSI specification.

6.5 Work products

6.5.1 Hardware safety requirements specification (including test and qualification criteria) resulting from requirements 6.4.1 to 6.4.8.

6.5.2 Hardware-software interface specification (refined) resulting from requirements 6.4.10 and 6.4.11.

NOTE This work product refers to the same work product as given in ISO 26262-6:2011 6.5.2.

6.5.3 Hardware safety requirements verification report resulting from requirement 6.4.9.

7 Hardware design

7.1 Objectives

The first objective of this clause is to design the hardware in accordance with the system design specification and the hardware safety requirements.

The second objective of this clause is to verify the hardware design against the system design specification and the hardware safety requirements.

7.2 General

Hardware design includes hardware architectural design and hardware detailed design. Hardware architectural design represents all hardware components and their interactions with one another. Hardware detailed design is at the level of electrical schematics representing the interconnections between hardware parts composing the hardware components.

In order to develop a single hardware design both hardware safety requirements as well as all non-safety requirements have to be complied with. Hence, in this subphase, safety and non-safety requirements are handled within one development process.

7.3 Inputs to this clause

7.3.1 Prerequisites

The following information shall be available:

- hardware safety requirements specification in accordance with 6.5.1;
- hardware-software interface specification (refined) in accordance with 6.5.2;
- system design specification in accordance with ISO 26262-4:2011, 7.5.2; and
- safety plan (refined) in accordance with 5.5.

7.3.2 Further supporting information

The following information can be considered:

- software safety requirements specification (see ISO 26262-6:2011, 6.5.1).

7.4 Requirements and recommendations

7.4.1 Hardware architectural design

7.4.1.1 The hardware architecture shall implement the hardware safety requirements defined in Clause 6.

7.4.1.2 Each hardware component shall inherit the highest ASIL from the hardware safety requirements it implements.

NOTE Each characteristic of the hardware component will inherit the highest ASIL from the hardware safety requirements that it implements.

7.4.1.3 If ASIL decomposition is applied to the hardware safety requirements during hardware architectural design, it shall be applied in accordance with ISO 26262-9:2011, Clause 5.

7.4.1.4 If a hardware element is made of sub-elements that have different ASILs assigned, or sub-elements that have no ASIL assigned and safety-related sub-elements, then each of these shall be treated in accordance with the highest ASIL, unless the criteria for coexistence in accordance with ISO 26262-9 are met.

7.4.1.5 The traceability between the hardware safety requirements and their implementation shall be maintained down to the lowest level of hardware components.

NOTE The traceability is not required down to hardware detailed design and no ASILs are assigned to hardware parts.

7.4.1.6 In order to avoid failures resulting from high complexity the hardware architectural design shall exhibit the following properties by use of the principles listed in Table 1:

a) modularity;

b) adequate level of granularity; and

c) simplicity.

Table 1 — Properties of modular hardware design

Properties		ASIL			
		A	B	C	D
1	Hierarchical design	+	+	+	+
2	Precisely defined interfaces of safety-related hardware components	++	++	++	++
3	Avoidance of unnecessary complexity of interfaces	+	+	+	+
4	Avoidance of unnecessary complexity of hardware components	+	+	+	+
5	Maintainability (service)	+	+	++	++
6	Testability[a]	+	+	++	++

[a] Testability includes testability during development and operation.

7.4.1.7 Non-functional causes for failure of a safety-related hardware component shall be considered during hardware architectural design, including the following influences, if applicable: temperature, vibrations, water, dust, EMI, cross-talk originating either from other hardware components of the hardware architecture or from its environment.

7.4.2 Hardware detailed design

7.4.2.1 In order to avoid common design faults, relevant lessons learned shall be applied in accordance with ISO 26262-2:2011, 5.4.2.7.

7.4.2.2 Non-functional causes for failure of a safety-related hardware part shall be considered during hardware detailed design, including the following influences, if applicable: temperature, vibrations, water, dust, EMI, noise factor, cross-talk originating either from other hardware parts of the hardware component or from its environment.

7.4.2.3 The operating conditions of the hardware parts used in the hardware detailed design shall comply with the specification of their environmental and operational limits.

7.4.2.4 Robust design principles should be considered.

NOTE Robust design principles can be shown by use of checklists based on QM methods.

EXAMPLE Conservative specification of components.

7.4.3 Safety analyses

7.4.3.1 Safety analyses on hardware design to identify the causes of failures and the effects of faults shall be applied in accordance with Table 2 and ISO 26262-9:2011, Clause 8.

NOTE 1 The initial purpose of the safety analyses is to support the specification of the hardware design. Subsequently, the safety analyses can be used for verification of the hardware design (see 7.4.4).

NOTE 2 In its aims of supporting the specification of the hardware design, qualitative analysis can be appropriate and sufficient.

Table 2 — Hardware design safety analysis

Methods		ASIL			
		A	B	C	D
1	Deductive analysis[a]	o	+	++	++
2	Inductive analysis[b]	++	++	++	++
NOTE The level of detail of the analysis is commensurate with the level of detail of the design. Both methods can, in certain cases, be carried out at different levels of detail.					

[a] A typical deductive analysis method is FTA.

[b] A typical inductive analysis method is FMEA.

7.4.3.2 This requirement applies to ASIL (B), C, and D of the safety goal. For each safety-related hardware component or part, the safety analyses shall identify the following for the safety goal under consideration:

a) safe faults;

b) single-point faults or residual faults; and

c) multiple-point faults (either perceived, detected or latent).

NOTE 1 In most of the cases, the analysis can be limited to dual-point faults. But sometimes multiple-point faults of a higher order than two can be shown relevant in the technical safety concept (e.g. when implementing redundant safety mechanisms).

NOTE 2 The intention of the identification of dual-point faults is not to require a systematic analysis of every possible combination of two hardware faults but, at a minimum, to consider combinations that derive from the technical safety concept (for instance the combination of two faults where one fault affects a safety-related element and another fault affects the corresponding safety mechanism intended to achieve or maintain a safe state).

7.4.3.3 This requirement applies to ASIL (B), C, and D of the safety goal. Evidence of the effectiveness of safety mechanisms to avoid single-point faults shall be made available.

For that purpose:

a) evidence of the ability of the safety mechanisms to maintain a safe state, or to switch safely into a safe state, shall be made available (in particular, appropriate failure mitigation ability within the fault tolerant time interval); and

b) diagnostic coverage with respect to residual faults shall be evaluated.

NOTE 1 A fault that can occur at anytime (e.g. not only at power-up) cannot be considered as being effectively covered if its diagnostic test interval, plus the fault reaction time of the associated safety mechanism, is longer than the relevant fault tolerant time interval.

NOTE 2 If a fault is such that it is possible to demonstrate that it occurs at power-up only and its probability of occurrence is negligible during the duration of the vehicle trip, then for those faults to execute a test at start-up after power-on is acceptable.

NOTE 3 An analysis such as FMEA or FTA can be used to structure the rationale.

NOTE 4 Depending on the knowledge of the failure modes of the hardware elements and their consequences at higher levels, the evaluation can be either a global diagnostic coverage of the hardware element, or a more detailed failure mode coverage evaluation.

NOTE 5 Annex D can be used as a starting point for diagnostic coverage (DC) with the claimed DC supported by a proper rationale.

7.4.3.4 This requirement applies to ASIL (B), C, and D of the safety goal. Evidence of the effectiveness of safety mechanisms to avoid latent faults shall be made available.

For that purpose:

a) evidence of the failure detection, and the ability to notify to the driver, within the acceptable multiple-point fault detection interval for latent faults, shall be made available in order to determine which faults remain latent and which faults are not latent; and

b) diagnostic coverage with respect to latent faults shall be evaluated.

NOTE 1 A fault cannot be considered covered if its diagnostic test interval, plus the fault reaction time of the associated safety mechanism, is longer than the relevant multiple-point fault detection interval for latent faults.

NOTE 2 An analysis such as FMEA or FTA can be used to structure the rationale.

NOTE 3 Annex D can be used as a starting point for DC with the claimed DC supported by a proper rationale.

NOTE 4 Depending on the knowledge of the failure modes of the hardware elements and their consequences at higher levels, the evaluation can be either a global diagnostic coverage of the hardware element, or a more detailed failure mode coverage evaluation.

7.4.3.5 If applicable, evidence that the hardware design is compliant with its requirements on independence shall be provided based on an analysis of dependent failures in accordance with ISO 26262-9:2011, Clause 7.

7.4.3.6 If new hazards introduced by the hardware design are not already covered by an existing safety goal, they shall be introduced and evaluated in the hazard analysis and risk assessment in accordance with the change management process in ISO 26262-8.

NOTE Newly identified hazards, not already covered by an existing safety goal, are usually non-functional hazards. Non-functional hazards are outside the scope of ISO 26262, but they can be annotated in the hazard analysis and risk assessment with the following statement “No ASIL is assigned to this hazard as it is not within the scope of ISO 26262”. However, an ASIL can be assigned for reference purposes.

7.4.4 Verification of hardware design

7.4.4.1 The hardware design shall be verified in accordance with ISO 26262-8, Clause 9, for compliance and completeness with respect to the hardware safety requirements. To achieve this, the methods listed in Table 3 shall be considered.

Table 3 — Hardware design verification

Methods		ASIL			
		A	B	C	D
1a	Hardware design walk-through[a]	++	++	o	o
1b	Hardware design inspection[a]	+	+	++	++
2	Safety analyses	In accordance with 7.4.3			
3a	Simulation[b]	o	+	+	+
3b	Development by hardware prototyping[b]	o	+	+	+

NOTE The scope of this verification review is technical correctness of the hardware design.

[a] Methods 1a and 1b serve as a check of the complete and correct implementation of the hardware safety requirements in the hardware design.

[b] Methods 3a and 3b serve as a check of particular points of the hardware design (e.g. as a fault injection technique) for which analytical methods 1 and 2 are not considered to be sufficient.

7.4.4.2 If it is discovered, during hardware design, that the implementation of any hardware safety requirement is not feasible, a request for change shall be issued in accordance with the change management process in ISO 26262-8.

7.4.5 Production, operation, service and decommissioning

7.4.5.1 Safety-related special characteristics shall be specified if safety analyses have shown them to be relevant. Attributes of safety-related special characteristics shall include:

a) the verification measures for production and operation; and

b) the acceptance criteria for these measures.

EXAMPLE A safety analysis of hardware design that relies on new sensor technologies (e.g., camera or radar sensors) can reveal the relevance of special installation procedures for these sensors. In such a case, additional verification measures for these components can be necessary during the production phase.

7.4.5.2 Instructions for assembly, disassembly and decommissioning of safety-related hardware elements shall be specified, if these operations can impact the technical safety concept.

7.4.5.3 The traceability of safety-related hardware elements shall be ensured, in accordance with ISO 26262-7:2011, 5.4.1.2.

NOTE This can include adequate labelling or other identification of hardware elements to indicate that they are safety-related.

7.4.5.4 Instructions for the maintenance of safety-related hardware elements shall be specified, if the maintenance can impact the technical safety concept.

7.5 Work products

7.5.1 Hardware design specification resulting from requirements in 7.4.1 and 7.4.2.

7.5.2 Hardware safety analysis report resulting from requirements in 7.4.3.

7.5.3 Hardware design verification report resulting from requirements in 7.4.4.

7.5.4 Specification of requirements related to production, operation, service and decommissioning resulting from requirements in 7.4.5.

8 Evaluation of the hardware architectural metrics

8.1 Objectives

The objective of this clause is to evaluate the hardware architecture of the item against the requirements for fault handling as represented by the hardware architectural metrics.

8.2 General

This clause describes two hardware architectural metrics for the evaluation of the effectiveness of the architecture of the item to cope with random hardware failures.

These metrics and associated target values apply to the whole hardware of the item and are complementary to the evaluation of safety goal violations due to random hardware failures described in Clause 9.

The random hardware failures addressed by these metrics are limited to some of the item's safety-related electrical and electronic hardware parts, namely those that can significantly contribute to the violation or the achievement of the safety goal, and to the single-point, residual and latent faults of those parts. For electromechanical hardware parts, only the electrical failure modes and failure rates are considered.

NOTE Hardware elements whose faults are multiple-point faults with a higher order than two can be omitted from the calculations unless they can be shown to be relevant in the technical safety concept.

The hardware architectural metrics can be applied iteratively during the hardware architectural design and the hardware detailed design.

The hardware architectural metrics are dependent upon the whole hardware of the item. Compliance with the target figures prescribed for the hardware architectural metrics is achieved for each safety goal in which the item is involved.

These hardware architectural metrics are defined to achieve the following objectives:

- be objectively assessable: metrics are verifiable and precise enough to differentiate between different architectures;
- support evaluation of the final design (the precise calculations are done with the detailed hardware design);
- make available ASIL dependent pass/fail criteria for the hardware architecture;
- reveal whether or not the coverage by the safety mechanisms, to prevent risk from single-point or residual faults in the hardware architecture, is sufficient (single-point fault metric);
- reveal whether or not the coverage by the safety mechanisms, to prevent risk from latent faults in the hardware architecture, is sufficient (latent-fault metric);
- address single-point faults, residual faults and latent faults;
- ensure robustness concerning uncertainty of hardware failure rates;
- be limited to safety-related elements; and
- support usage on different element levels, e.g. target values can be assigned to suppliers' hardware elements.

EXAMPLE To facilitate distributed developments, target values can be assigned to microcontrollers or ECUs.

8.3 Inputs of this clause

8.3.1 Prerequisites

The following information shall be available:

- hardware safety requirements specification in accordance with 6.5.1;
- hardware design specification in accordance with 7.5.1; and
- hardware safety analysis report in accordance with 7.5.2.

8.3.2 Further supporting information

The following information can be considered:

— technical safety concept (see ISO 26262-4:2011, 7.5.1); and

— system design specification (see ISO 26262-4:2011, 7.5.2).

8.4 Requirements and recommendations

8.4.1 This requirement applies to ASIL (B), C, and D of the safety goal. The concepts of diagnostic coverage, single-point fault metric and latent-fault metric, in accordance with Annex C, shall apply to requirements 8.4.2 to 8.4.9.

8.4.2 This requirement applies to ASIL (B), C, and D of the safety goal. The diagnostic coverage of safety-related hardware elements by safety mechanisms shall be estimated with respect to residual faults and with respect to relevant latent faults.

NOTE 1 For this purpose, Tables D.1 to D.14 can be used as a starting point with the claimed DC supported by a proper rationale.

NOTE 2 Depending on the knowledge of the failure modes of the hardware elements and their consequences at a higher level, the evaluation can be either a global diagnostic coverage of the hardware element, or a more detailed failure mode coverage evaluation.

8.4.3 This requirement applies to ASIL (B), C, and D of the safety goal. The estimated failure rates for hardware parts used in the analyses shall be determined:

a) using hardware part failure rates data from a recognised industry source, or

EXAMPLE Commonly recognised industry sources to determine the hardware part failure rates and the failure mode distributions include IEC/TR 62380, IEC 61709, MIL HDBK 217 F notice 2, RIAC HDBK 217 Plus, UTE C80-811, NPRD 95, EN 50129:2003, Annex C, IEC 62061:2005, Annex D, RIAC FMD97 and MIL HDBK 338.

NOTE 1 The failure rate values given in these databases are generally considered to be pessimistic.

b) using statistics based on field returns or tests. In this case, the estimated failure rate should have an adequate confidence level, or

c) using expert judgement founded on an engineering approach based on quantitative and qualitative arguments. Expert judgement shall be exercised in accordance with structured criteria as a basis for this judgement. These criteria shall be set before the estimation of failure rates is made.

NOTE 2 The criteria for expert judgment can include field experience, testing, reliability analysis, and novelty of design.

8.4.4 This requirement applies to ASIL (B), C, and D of the safety goal. If sufficient evidence of the calculated failure rate of a single-point fault or latent fault cannot be made available, alternative means shall be proposed (e.g. add safety mechanisms to detect and control this fault).

NOTE Sufficient evidence means, for instance, that evidence is given that the failure rate has been determined using one of the methods listed in 8.4.3.

8.4.5 This requirement applies to ASIL (B), C, and D of the safety goal. For each safety goal, a quantitative target value for the "single-point fault metric" as required in ISO 26262-4:2011, 7.4.4.2, shall be based on one of the following sources of reference target values:

a) derived from the hardware architectural metrics calculation applied on similar well-trusted design principles, or

NOTE 1 Two similar designs have similar functionalities and similar safety goals with the same assigned ASIL.

b) derived from Table 4.

Table 4 — Possible source for the derivation of the target “single-point fault metric” value

	ASIL B	ASIL C	ASIL D
Single-point fault metric	≥90 %	≥97 %	≥99 %

NOTE 2 This quantitative target is intended to provide:

- design guidance; and
- evidence that the design complies with the safety goals.

8.4.6 This requirement applies to ASIL (B), (C), and D of the safety goal. For each safety goal, a quantitative target value for “latent-fault metric” as required in ISO 26262-4:2011, 7.4.4.2 shall be based on one of the following sources of reference target values:

a) derived from the hardware architectural metrics calculation applied on similar well-trusted design principles; or

NOTE 1 Two similar designs have similar functionalities and similar safety goals with the same assigned ASIL.

b) derived from Table 5.

Table 5 — Possible source for the derivation of the target “latent-fault metric” value

	ASIL B	ASIL C	ASIL D
Latent-fault metric	≥60 %	≥80 %	≥90 %

NOTE 2 This quantitative target is intended to provide:

- design guidance; and
- evidence that the design complies with the safety goals.

8.4.7 This requirement applies to ASIL (B), C, and D of the safety goal. For each safety goal, the whole hardware of the item shall comply with one of the following alternatives:

a) to meet the target “single-point fault metric” value, as described in 8.4.5, or

b) to meet the appropriate targets prescribed at the hardware element level which are sufficient to comply with the single-point fault metric's target value assigned to the whole hardware of the item, given in requirement 8.4.5, with the rationale for compliance with these targets at the hardware element level.

NOTE 1 If an item contains different kinds of hardware elements with significantly different failure rate levels, the risk exists that compliance with the hardware architectural metrics only focus on the kind of hardware elements with the highest magnitude of failure rates. (One example where this can occur is for the single-point fault metric for which compliance can be achieved by considering the failure rates for failures of wires / fuses / connectors, while disregarding the failure rates of hardware parts with significantly lower failure rates.) The prescription of appropriate metric target values for each kind of hardware helps to avoid this side effect.

NOTE 2 The transient faults are considered when shown to be relevant due, for instance, to the technology used. They can be addressed either by specifying and verifying a dedicated target “single-point fault metric” value to them (as explained in NOTE 1) or by a qualitative rationale based on the verification of the effectiveness of the internal safety mechanisms implemented to cover these transient faults.

NOTE 3 If the target is not met, the rationale for how the safety goal is achieved will be assessed as given in 4.1.

NOTE 4 Some or all of the applicable safety goals can be considered together for the determination of the single-point fault metric; but in this case the metric's target to be considered is that of the safety goal with the highest ASIL.

8.4.8 This requirement applies to ASIL (B), (C), and D of the safety goal. For each safety goal, the whole hardware of the item shall comply with one of the following alternatives:

a) to meet the target “latent-fault metric” value, as described in 8.4.6, or

b) to meet the appropriate targets prescribed at the hardware element level which are sufficient to comply with the latent-fault metric's target value assigned to the whole hardware of the item as described in requirement 8.4.6 and to provide the rationale for compliance with these targets at the hardware element level, or

c) to meet the target values for the diagnostic coverage, with respect to latent faults, identical to the target value given in reference 8.4.6 for the latent-fault metric (treated as a diagnostic coverage), for each hardware element with faults that can lead to the unavailability of a safety mechanism (to prevent a fault from violating the safety goal). This alternative applies when each safety mechanism, whose unavailability can contribute to the violation of the safety goal, is based on fault detection,

NOTE 1 Alternative c) is limited to the cases where each relevant safety mechanism is based on fault detection. It is supposed that in this case the potentially latent faults of the intended functionality are alerted through the detection of these safety mechanisms. In other cases this alternative cannot be applied and alternatives a) and b) are the only possibilities.

NOTE 2 In the case of c), a metric is not calculated, only the coverage of the hardware elements by safety mechanisms with respect to latent faults is evaluated.

NOTE 3 If an item contains different kinds of hardware elements with significantly different failure rate levels, the risk exists that compliance with the hardware architectural metrics only focuses on the kind of hardware elements with the highest magnitude of failure rates. (One example where this can occur is for the single-point fault metric for which compliance could be achieved by considering the failure rates for failures of wires / fuses / connectors, while disregarding the failure rates of hardware parts with significantly lower failure rates.) The prescription of appropriate metric target values for each kind of hardware helps to avoid this side effect.

NOTE 4 If the target is not met, the rationale for how the safety goal is achieved will be assessed as given in 4.1.

NOTE 5 Some or all of the applicable safety goals can be considered together for the determination of the latent-fault metric; but in this case the metric's target to be considered is that of the safety goal with the highest ASIL.

8.4.9 This requirement applies to ASIL (B), C, and D of the safety goal. A verification review of the result of the applied methods in 8.4.7 and 8.4.8 shall be performed in order to provide evidence of its technical correctness and completeness in accordance with ISO 26262-8:2011, Clause 9.

NOTE Careful verification of the single-point fault metric ensures that only failure rates of safety-related hardware elements are taken into consideration, so that the metric is not inappropriately skewed by unnecessary safety-related hardware elements without the potential for having single-point faults or residual faults (e.g. by adding unnecessary hardware elements to a safety mechanism).

8.5 Work products

8.5.1 Analysis of the effectiveness of the architecture of the item to cope with the random hardware failures resulting from requirements 8.4.1 to 8.4.8.

8.5.2 Review report of evaluation of the effectiveness of the architecture of the item to cope with the random hardware failures resulting from requirement 8.4.9.

9 Evaluation of safety goal violations due to random hardware failures

9.1 Objectives

The objective of the requirements in this clause is to make available criteria that can be used in a rationale that the residual risk of a safety goal violation, due to random hardware failures of the item, is sufficiently low.

NOTE "Sufficiently low" means "comparable to residual risks on items already in use".

9.2 General

Two alternative methods (see 9.4) are proposed to evaluate whether the residual risk of safety goal violations is sufficiently low.

Both methods evaluate the residual risk of violating a safety goal due to single-point faults, residual faults, and plausible dual-point faults. Multiple-point faults can also be considered if shown to be relevant to the safety concept. In this analysis, coverage of safety mechanisms will be considered for residual and dual-point faults, and exposure duration will be considered as well for dual-point faults.

The first method consists of using a probabilistic metric called "Probabilistic Metric for random Hardware Failures" (PMHF), to evaluate the violation of the considered safety goal using, for example, quantified FTA and to compare the result of this quantification with a target value.

The second method consists of the individual evaluation of each residual and single-point fault, and of each dual-point failure leading to the violation of the considered safety goal. This analysis method can also be considered to be a cut-set analysis.

NOTE In the context of reliability analysis, a cut-set in a fault tree is a set of basic events whose occurrence leads to the occurrence of the top event.

The chosen method can be applied iteratively during the hardware architectural design and the hardware detailed design.

The scope of this clause is limited to the random hardware failures of the item. The parts considered in the analyses are the electrical and electronic hardware parts. For electromechanical hardware parts, only the electrical failure modes and failures rate are considered.

9.3 Inputs to this clause

9.3.1 Prerequisites

The following information shall be available:

- hardware safety requirements specification in accordance with 6.5.1;
- hardware design specification in accordance with 7.5.1; and
- hardware safety analysis report in accordance with 7.5.2.

9.3.2 Further supporting information

The following information can be considered:

- technical safety concept (see ISO 26262-4:2011, 7.5.1); and
- system design specification (see ISO 26262-4:2011, 7.5.2).

9.4 Requirements and recommendations

9.4.1 General

This requirement applies to ASIL (B), C and D of the safety goal. The item shall comply with either 9.4.2 or 9.4.3.

9.4.2 Evaluation of Probabilistic Metric for random Hardware Failures (PMHF)

9.4.2.1 This requirement applies to ASIL (B), C, and D of the safety goal. Quantitative target values for the maximum probability of the violation of each safety goal due to random hardware failures as required in ISO 26262-4:2011, 7.4.4.3, shall be defined using one of the sources a), b) or c) of reference target values, as outlined below:

a) derived from Table 6, or

b) derived from field data from similar well-trusted design principles, or

c) derived from quantitative analysis techniques applied to similar well-trusted design principles using failure rates in accordance with 8.4.3.

NOTE 1 These quantitative target values derived from sources a), b), or c) do not have any absolute significance and are only useful to compare a new design with existing ones. They are intended to make available design guidance as described in 9.1 and to make available evidence that the design complies with the safety goals.

NOTE 2 Two similar designs have similar functionalities and similar safety goals with the same assigned ASIL.

Table 6 — Possible source for the derivation of the random hardware failure target values

ASIL	Random hardware failure target values
D	$<10^{-8}$ h^{-1}
C	$<10^{-7}$ h^{-1}
B	$<10^{-7}$ h^{-1}
NOTE The quantitative target values described in this table can be tailored as specified in 4.1 to fit specific uses of the item (e.g. if the item is able to violate the safety goal for durations longer than the typical use of a passenger car).	

9.4.2.2 This requirement applies to ASIL (B), C, and D of the safety goal. Quantitative target values of requirement 9.4.2.1 shall be expressed in terms of average probability per hour over the operational lifetime of the item.

9.4.2.3 This requirement applies to ASIL (B), C, and D of the safety goal. A quantitative analysis of the hardware architecture with respect to the single-point, residual and dual-point faults shall provide evidence that target values of requirement 9.4.2.1 have been achieved. This quantitative analysis shall consider:

a) the architecture of the item;

b) the estimated failure rate for the failure modes of each hardware part that would cause a single-point fault or a residual fault;

c) the estimated failure rate for the failure modes of each hardware part that would cause a dual-point fault;

d) the diagnostic coverage of safety-related hardware elements by safety mechanisms; and

e) the exposure duration in the case of dual-point faults.

NOTE 1 Failure modes of hardware elements that can cause a failure of a safety-related hardware element and its safety mechanism simultaneously are considered in the quantitative analysis. They can be single-point faults, residual faults or multiple-point faults.

NOTE 2 Exposure duration starts as soon as the fault can occur and includes:

a) the multiple-point fault detection interval associated with each safety mechanism, or the lifetime of the vehicle if the fault is not indicated to the driver (latent fault);

b) the maximum duration of a trip (in the case that the driver is requested to stop in a safe manner); and

c) the average time interval until the vehicle is at the workshop for repair (in the case that the driver is alerted to have the vehicle repaired).

Therefore, exposure duration depends on the type of monitoring involved (e.g. continuous monitoring, periodic self-tests, driver monitoring, no monitoring) and the kind of reaction when the fault has been detected. It can be as short as a few milliseconds in the case of a continuous monitoring triggering a transition to a safe state. It can be as long as the car lifetime when there is no monitoring.

Example of assumptions on the average time to vehicle repair, depending on the fault type:

- 200 vehicle trips for reduction of comfort features;
- 50 vehicle trips for reduction of driving support features;
- 20 vehicle trips for amber warning lights or impacts on driving behaviour;
- one vehicle trip for red warning lights.

The time taken for repair is usually not considered (except to evaluate hazards that can expose maintenance personnel).

The mean duration of a vehicle trip can be considered as being equal to 1 h.

NOTE 3 In most cases, multiple-point failures of a higher order than two have a negligible contribution with respect to the quantitative target values. However, in some particular cases (very high failure rate or poor diagnostic coverage), it can be necessary to provide two redundant safety mechanisms to reach the target. When the technical safety concept is based on redundant safety mechanisms, multiple-point failures of a higher order than two are considered in the analysis.

NOTE 4 For safety mechanisms using integrated diagnostics, Tables D.1 to D.14 can be used as a starting point to evaluate the diagnostic coverage of these safety mechanisms, with the claimed DC supported by a proper rationale.

NOTE 5 Situations when the item is in power-down mode are not included in the calculation of the average probability per hour, thereby preventing the artificial reduction of the average probability per hour. Thus, for an item that is only operational 1 h each day, the remaining 23 h are not considered in the calculation of this operational target value.

NOTE 6 If the target is not met, the rationale for how the safety goal is achieved will be assessed as given in 4.1.

NOTE 7 Depending on the knowledge of the failure modes of the hardware elements and their consequences at a higher level, the evaluation can be either a global diagnostic coverage of the hardware element, or a more detailed failure mode coverage evaluation.

9.4.2.4 This requirement applies to ASIL C and D of the safety goal. A single-point fault occurring in a hardware part shall only be considered acceptable if dedicated measures are taken.

NOTE Dedicated measures can include:

a) design features such as hardware part over design (e.g. electrical or thermal stress rating) or physical separation (e.g. spacing of contacts on a printed circuit board);

b) a special sample test of incoming material to reduce the risk of occurrence of this failure mode;

c) a burn-in test;

d) a dedicated control set as part of the control plan; and

e) assignment of safety-related special characteristics.

9.4.2.5 This requirement applies to ASIL C and D of the safety goal. A hardware part shall be dealt with by dedicated measures (the note in 9.4.2.4 lists examples of dedicated measures) if its diagnostic coverage (with respect to residual faults) is lower than 90 %.

NOTE The proportion of safe faults of the hardware part can be considered when determining the coverage of the safety mechanisms. In this case the calculation of the coverage is done analogously to the calculation of the single-point fault metric, but at the hardware part level instead of at the item level.

9.4.2.6 This requirement applies to ASIL (B), C, and D of the safety goal. The failure rates for hardware parts used in the analyses shall be estimated in accordance with 8.4.3.

9.4.2.7 This requirement applies to ASIL (B), C, and D of the safety goal. In order to avoid bias in the quantification, if failure rates from multiple sources are combined, they shall be scaled using a scaling factor to be consistent. Scaling is possible if a rationale for the scaling factor between two failure rate sources is available.

NOTE Guidance is given in Annex F on the application of scaling factors.

9.4.3 Evaluation of each cause of safety goal violation

9.4.3.1 A method for evaluation of each cause of a safety goal violation due to random hardware failures is illustrated by flowcharts in Figures 3 and 4. Each single-point fault is evaluated using criteria on the occurrence of the fault. Each residual fault is evaluated using criteria combining the occurrence of the fault and the efficiency of the safety mechanism.

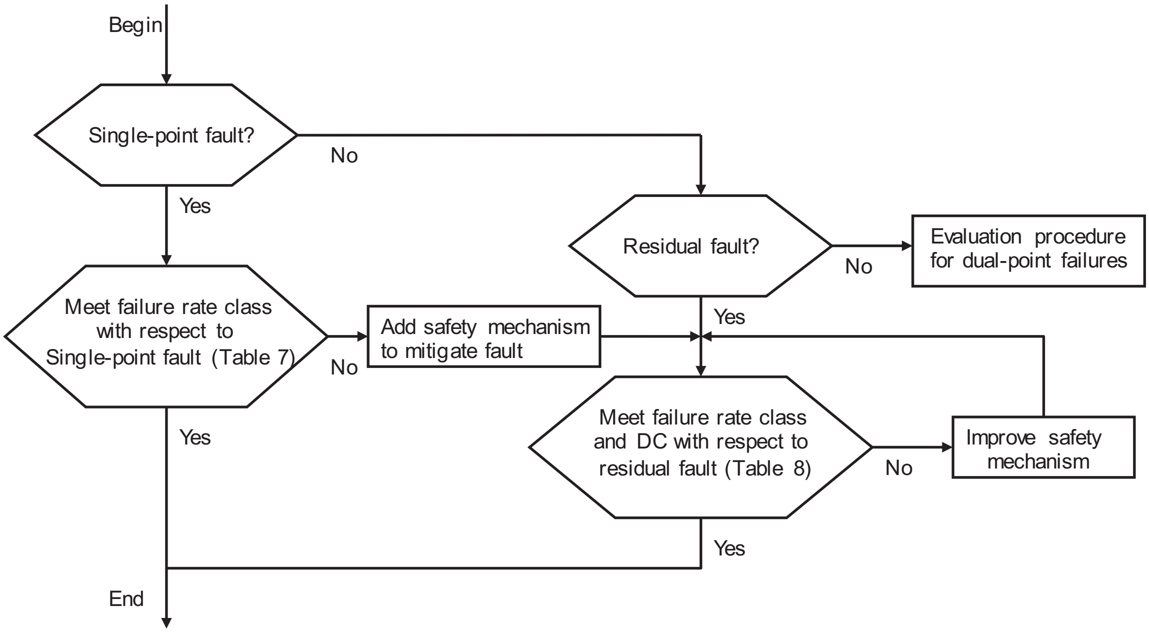

Begin
Single-point fault?
No
Yes
Residual fault?
No
Evaluation procedure for dual-point failures
Meet failure rate class with respect to Single-point fault (Table 7)
No
Add safety mechanism to mitigate fault
Yes
Meet failure rate class and DC with respect to residual fault (Table 8)
No
Improve safety mechanism
Yes
Yes
End

Figure 3 — Evaluation procedure for single-point and residual faults

The procedure to be applied for dual-point failures is illustrated by the flowchart in Figure 4. Each dual-point failure is first evaluated regarding its plausibility. A dual-point failure is considered not plausible if both faults leading to the failure are detected or perceived in a sufficiently short time with sufficient coverage. If the dual-point failure is plausible, the faults causing it are then evaluated using criteria combining occurrence of the fault and coverage of the safety mechanisms. The evaluation procedures described in Figures 3 and 4 apply to the hardware parts (transistors, etc.) level.

NOTE For complex hardware parts like microcontrollers, it can be appropriate to apply this procedure on a more detailed level like CPU, RAM, ROM, etc.

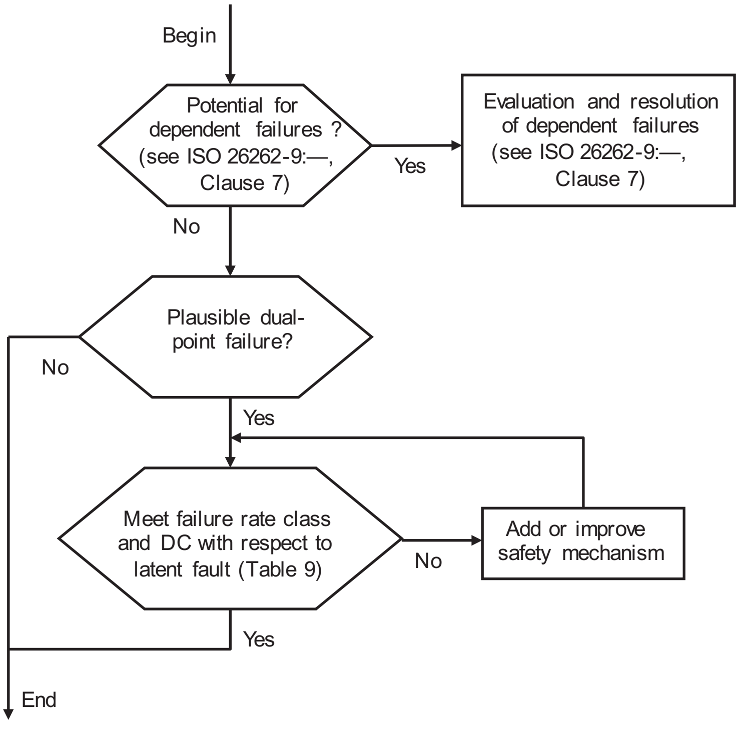

Begin
Potential for dependent failures ? (see ISO 26262-9:—, Clause 7)
Yes
Evaluation and resolution of dependent failures (see ISO 26262-9:—, Clause 7)
No
Plausible dual-point failure?
No
Yes
Meet failure rate class and DC with respect to latent fault (Table 9)
No
Add or improve safety mechanism
Yes
End

Figure 4 — Evaluation procedure for dual-point failures

9.4.3.2 This requirement applies to ASIL (B), C, and D of the safety goal. An individual evaluation of each single-point fault, residual fault and dual-point failure violating the considered safety goal shall be performed at the hardware part level. This evaluation shall provide evidence that each single-point fault, residual fault and dual-point failure violating the considered safety goal is acceptable in accordance with requirements 9.4.3.3 to 9.4.3.12.

NOTE 1 This analysis can be viewed as a review of cut sets where absence or incompleteness of coverage is treated as a fault.

NOTE 2 In most cases, multiple-point failures of a higher order than two are negligible. However, in some particular cases (very high failure rate or poor diagnostic coverage), it can be necessary to provide two redundant safety mechanisms. Therefore it is necessary to consider multiple-point failures of a higher order than two in the analysis when the technical safety concept is based on redundant safety mechanisms.

NOTE 3 For complex hardware parts like microcontrollers it can be appropriate to apply this procedure at a more detailed level like CPU, RAM, ROM, etc.

9.4.3.3 This requirement applies to ASIL (B), C, and D of the safety goal. The failure rate class ranking for a hardware part failure rate shall be determined as follows:

NOTE 1 The failure rate classes 1, 2 and 3 are introduced to address the failure occurrence rates. These classes are analogous to the occurrence levels 1, 2 and 3, respectively, used in an FMEA, where a 1 is assigned to failure modes which have the lowest occurrence rate.

a) the failure rate corresponding to failure rate class 1 shall be less than the target for ASIL D divided by 100; unless 9.4.3.4 is applied;

   NOTE 2 The target values given in Table 6 can be used.

b) the failure rate corresponding to failure rate class 2 shall be less than or equal to 10 times the failure rate corresponding to failure rate class 1;

c) the failure rate corresponding to failure rate class 3 shall be less than or equal to 100 times the failure rate corresponding to failure rate class 1; and

d) the failure rate corresponding to failure rate class i, $i > 3$ shall be less than or equal to $10^{(i-1)}$ times the failure rate corresponding to failure rate class 1.

   NOTE 3 The failure rate class assignment is based upon the hardware part failure rate.

   NOTE 4 For the case where a small number of parts (such as a microcontroller) have failure rates higher than the failure rate class i upper limit, then these parts can be assigned a class i occurrence if the resulting average failure rate of parts assigned class i is lower than failure rate class i's upper limit.

9.4.3.4 If a rationale is provided, the failure rate class ranking may be divided by a number lower than 100. In this case, it shall be ensured that a correct ranking is maintained while considering the single-point faults, residual faults and higher degree cut-sets together.

EXAMPLE The rationale can be based on the number of minimal cut-sets.

9.4.3.5 This requirement applies to ASIL (B), C and D of the safety goal. A single-point fault occurring in a hardware part shall only be considered as acceptable if the corresponding hardware part failure rate ranking complies with the targets given in Table 7.

Table 7 — Targets of failure rate classes of hardware parts regarding single-point faults

ASIL of the safety goal	Failure rate class
D	Failure rate class 1 + dedicated measures[a]
C	Failure rate class 2 + dedicated measures[a] or Failure rate class 1
B	Failure rate class 2 or Failure rate class 1

[a] The note in requirement 9.4.2.4 gives examples of dedicated measures.

NOTE When assessing the failure rate class, the proportion of safe faults of the hardware part can be considered.

9.4.3.6 This requirement applies to ASIL (B), C, and D of the safety goal. A residual fault occurring in a hardware part shall be considered acceptable if the failure rate class ranking complies with the targets given in Table 8 for the diagnostic coverage (with respect to residual faults) of the corresponding hardware part.

NOTE 1 The considered failure rate is the hardware part failure rate and does not take into account the effectiveness of the safety mechanisms.

Table 8 — Maximum failure rate classes for a given diagnostic coverage of the hardware part – residual faults

ASIL of the safety goal	Diagnostic coverage with respect to residual faults			
	≥99,9 %	≥99 %	≥90 %	<90 %
D	Failure rate class 4	Failure rate class 3	Failure rate class 2	Failure rate class 1 + dedicated measures[a]
C	Failure rate class 5	Failure rate class 4	Failure rate class 3	Failure rate class 2 + dedicated measures[a]
B	Failure rate class 5	Failure rate class 4	Failure rate class 3	Failure rate class 2

[a] The note in requirement 9.4.2.4 gives examples of dedicated measures.

NOTE 2 Table 8 specifies the connection between maximum failure rate class allowed given the target ASIL and the diagnostic coverage. Lower failure rate classes are acceptable but not required.

NOTE 3 “Lower failure rate classes” means failure rate classes with a lower number. For example, “lower failure rate classes” with respect to failure rate class 3 means failure rate classes 2 and 1.

NOTE 4 The proportion of safe faults of the hardware part can be taken into account when determining the coverage of the safety mechanisms. In this case, the calculation of the coverage is done analogously to the calculation of the single-point fault metric, but at the hardware part level instead of at the item level.

9.4.3.7 This requirement applies for ASIL C and D of the safety goal. For failure rate classes i, $i > 3$, a residual fault shall be considered as acceptable if the diagnostic coverage is greater than or equal to $[100 - 10^{(3-i)}]$ % for ASIL D or greater than or equal to $[100 - 10^{(4-i)}]$ % for ASIL C.

NOTE 1 The considered failure rate is the hardware part failure rate, and does not take into account the effectiveness of the safety mechanisms.

NOTE 2 The proportion of safe faults of the hardware part can be taken into account when determining the coverage of the safety mechanisms. In this case the calculation of the coverage is done analogously to the calculation of the single-point fault metric, but at the hardware part level instead of at the item level.

9.4.3.8 This requirement applies to ASIL D of the safety goal. A dual-point failure shall be considered plausible if

a) one or both hardware parts involved has a diagnostic coverage (with respect to the latent faults) of less than 90 %, or

b) one of the dual-point faults causing the dual-point failure remains latent for a time longer than the multiple-point fault detection interval as specified in requirement 6.4.8.

NOTE The proportion of safe faults of the hardware part can be taken into account when determining the coverage of the safety mechanisms. In this case the calculation of the coverage is done analogously to the calculation of the latent fault metric, but at the hardware part level instead of at the item level.

9.4.3.9 This requirement applies to ASIL C of the safety goal. A dual-point failure shall be considered plausible if

a) one or both hardware parts involved has a diagnostic coverage (with respect to the latent faults) of less than 80 %; or

b) one of the dual-point faults causing the dual-point failure remains latent for a time longer than the multiple-point fault detection interval as specified in requirement 6.4.8.

NOTE The proportion of safe faults of the hardware part can be taken into account when determining the coverage of the safety mechanisms. In this case the calculation of the coverage is done analogously to the calculation of the latent fault metric, but at the hardware part level instead of at the item level.

9.4.3.10 This requirement applies to ASIL C and D of the safety goal. A dual-point failure that is not plausible shall be considered compatible with the safety goal target and thus acceptable.

9.4.3.11 This requirement applies to ASIL C and D of the safety goal. A dual-point fault occurring in a hardware part and contributing to a plausible dual-point failure shall be considered acceptable if the corresponding hardware part complies with the targets for the failure rate class ranking and diagnostic coverage (with respect to latent faults) given in Table 9.

NOTE 1 The considered failure rate is the hardware part failure rate. Therefore, it does not consider the effectiveness of safety mechanisms.

Table 9 — Targets of failure rate class and coverage of hardware part regarding dual-point faults

ASIL of safety goal	Diagnostic coverage with respect to latent faults		
	>= 99 %	>= 90 %	<90 %
D	Failure rate class 4	Failure rate class 3	Failure rate class 2
C	Failure rate class 5	Failure rate class 4	Failure rate class 3

NOTE 2 Table 9 specifies the maximum failure rate class allowed given the target ASIL level and the level of diagnostic coverage achieved. Lower failure rate classes are acceptable but not required.

NOTE 3 “Lower failure rate classes” means failure rate classes with a lower number. For example, “lower failure rate classes” with respect to failure rate class 3 means failure rate classes 2 and 1.

NOTE 4 The proportion of safe faults of the hardware part can be taken into account when determining the coverage of the safety mechanisms. In this case the calculation of the coverage is done analogously to the calculation of the latent fault metric, but at the hardware part level instead of at the item level.

9.4.3.12 This requirement applies to ASIL (B), C, and D of the safety goal. The failure rate class ranking of the hardware part failure rate used in the analyses shall be justified by using sources of failure rates described in 8.4.3. If failure rates from multiple data sources are used in the analyses, then the rates shall be scaled as described in 9.4.2.7.

9.4.4 Verification review

This requirement applies to ASIL (B), C, and D of the safety goal. A verification review of the analysis resulting from the set of requirements 9.4.2 or 9.4.3 shall be performed in order to provide evidence of its technical correctness and completeness in accordance with ISO 26262-8:2011, Clause 9.

9.5 Work products

9.5.1 Analysis of safety goal violations due to random hardware failures resulting from requirements in 9.4.2 or in 9.4.3.

9.5.2 Specification of dedicated measures for hardware, if needed, including the rationale regarding the effectiveness of the dedicated measures, resulting from requirements 9.4.2.4, 9.4.2.5, 9.4.3.5 and 9.4.3.6.

9.5.3 Review report of evaluation of safety goal violations due to random hardware failures resulting from requirement 9.4.4.

10 Hardware integration and testing

10.1 Objectives

The objective of this clause is to ensure, by testing, the compliance of the developed hardware with the hardware safety requirements.

The requirements in 10.4.1 to 10.4.6 apply to the hardware of an element.

10.2 General

The activities described in this clause aim at integrating hardware elements and testing the hardware design to verify its compliance with the hardware safety requirements in accordance with the appropriate ASIL.

Hardware integration and testing differs from the qualification of hardware components activity of ISO 26262-8:2011, Clause 13, which gives evidence of the suitability of intermediate level hardware components and parts for their use as parts of items, systems or elements developed in compliance with ISO 26262.

10.3 Inputs of this clause

10.3.1 Prerequisites

The following information shall be available:

- safety plan (refined) in accordance with 5.5;
- item integration and testing plan (refined) in accordance with ISO 26262-4:2011, 5.5.3;
- hardware safety requirements specification in accordance with 6.5.1; and
- hardware design specification in accordance with 7.5.1.

10.3.2 Further supporting information

The following information can be considered:

- project plan (refined) (see ISO 26262-4:2011, 5.5.1); and
- hardware safety analysis report (see 7.5.2).

10.4 Requirements and recommendations

10.4.1 Hardware integration and testing activities shall be performed in accordance with ISO 26262-8:2011, Clause 9.

10.4.2 Hardware integration and testing activities shall be coordinated with the item integration and testing plan given in ISO 26262-4:2011, 5.5.5.

NOTE If ASIL decomposition is applied, as defined in ISO 26262-9:2011, Clause 5, the corresponding integration activities of the decomposed elements, and the subsequent activities, are applied at the ASIL before decomposition.

10.4.3 The test equipment shall be subject to the control of a monitoring quality system.

10.4.4 To enable the appropriate specification of test cases for the selected hardware integration tests, test cases shall be derived using an appropriate combination of methods listed in Table 10.

Table 10 — Methods for deriving test cases for hardware integration testing

Methods		ASIL			
		A	B	C	D
1a	Analysis of requirements	++	++	++	++
1b	Analysis of internal and external interfaces	+	++	++	++
1c	Generation and analysis of equivalence classes[a]	+	+	++	++
1d	Analysis of boundary values[b]	+	+	++	++
1e	Knowledge or experience based error guessing[c]	++	++	++	++
1f	Analysis of functional dependencies	+	+	++	++
1g	Analysis of common limit conditions, sequences and sources of dependent failures	+	+	++	++
1h	Analysis of environmental conditions and operational use cases	+	++	++	++
1i	Standards if existing[d]	+	+	+	+
1j	Analysis of significant variants[e]	++	++	++	++

[a] In order to derive the necessary test cases efficiently, analysis of similarities can be conducted.

[b] For example, values approaching and crossing the boundaries between specified values, and out of range values.

[c] "Error guessing tests" can be based on data collected through a lessons learned process, or expert judgment, or both. It can be supported by FMEA.

[d] Existing standards include ISO 16750 and ISO 11452.

[e] The analysis of significant variants includes worst-case analysis.

10.4.5 The hardware integration and testing activities shall verify the completeness and correctness of the implementation of the safety mechanisms with respect to the hardware safety requirements.

To achieve these objectives, the methods listed in Table 11 shall be considered.

Table 11 — Hardware integration tests to verify the completeness and correctness of the safety mechanisms implementation with respect to the hardware safety requirements

Methods		ASIL			
		A	B	C	D
1	Functional testing[a]	++	++	++	++
2	Fault injection testing[b]	+	+	++	++
3	Electrical testing[c]	++	++	++	++

[a] Functional testing aims at verifying that the specified characteristics of the item have been achieved. The item is given input data, which adequately characterises the expected normal operation. The outputs are observed and their response is compared with that given by the specification. Anomalies with respect to the specification and indications of an incomplete specification are analysed.

[b] Fault injection testing aims at introducing faults in the hardware product and analysing the response. This testing is appropriate whenever a safety mechanism is defined. Model-based fault injection (e.g. fault injection done at the gate-level netlist level) is also applicable, especially when fault injection testing is very difficult to do at the hardware product level. For example, showing the response of safety mechanisms to transient faults inside hardware parts, such as a microcontroller, is very difficult to do with fault insertion at the hardware product level since it would require irradiation tests.

[c] Electrical testing aims at verifying compliance with hardware safety requirements within the specified (static and dynamic) voltage range.

10.4.6 The hardware integration and testing activities shall verify robustness of hardware against external stresses.

To achieve these objectives, the methods listed in Table 12 shall be considered.

Table 12 — Hardware integration tests to verify robustness and operation under external stresses

Methods		ASIL			
		A	B	C	D
1a	Environmental testing with basic functional verification[a]	++	++	++	++
1b	Expanded functional test[b]	o	+	+	++
1c	Statistical test[c]	o	o	+	++
1d	Worst case test[d]	o	o	o	+
1e	Over limit test[e]	+	+	+	+
1f	Mechanical test[f]	++	++	++	++
1g	Accelerated life test[g]	+	+	++	++
1h	Mechanical Endurance test[h]	++	++	++	++
1i	EMC and ESD test[i]	++	++	++	++
1j	Chemical test [j]	++	++	++	++

[a] During environmental testing with basic functional verification the hardware is put under various environmental conditions during which the hardware requirements are assessed. ISO 16750-4 can be applied.

[b] Expanded functional testing checks the functional behaviour of the item in response to input conditions that are expected to occur only rarely (for instance extreme mission profile values), or that are outside the specification of the hardware (for instance, an incorrect command). In these situations, the observed behaviour of the hardware element is compared with the specified requirements.

[c] Statistical tests aim at testing the hardware element with input data selected in accordance with the expected statistical distribution of the real mission profile. The acceptance criteria are defined so that the statistical distribution of the results confirms the required failure rate.

[d] Worst-case testing aims at testing cases found during worst-case analysis. In such a test, environmental conditions are changed to their highest permissible marginal values defined by the specification. The related responses of the hardware are inspected and compared with the specified requirements.

[e] In over limit testing, the hardware elements are submitted to environmental or functional constraints increasing progressively to values more severe than specified until they stop working or they are destroyed. The purpose of this test is to determine the margin of robustness of the elements under test with respect to the required performance.

[f] Mechanical test applies to mechanical properties such as tensile strength.

[g] Accelerated life test aims at predicting the behaviour evolution of a product in its normal operational conditions by submitting it to stresses higher than those expected during its operational lifetime. Accelerated testing is based on an analytical model of failure mode acceleration.

[h] The aim of these tests is to study the mean time to failure or the maximum number of cycles that the element can withstand. Test can be performed up to failure or by damage evaluation.

[i] ISO 7637-2, ISO 7637-3, ISO 10605, ISO 11452-2 and ISO 11452-4 can be applied for EMC tests; ISO 16750-2 can be applied for ESD tests.

[j] For chemical tests, ISO 16750-5 can be applied.

10.5 Work products

10.5.1 Hardware integration and testing report resulting from requirements **10.4.1 to 10.4.6.**

Annex A
(informative)

Overview of and workflow of product development at the hardware level

Table A.1 provides an overview of objectives, prerequisites and work products of the particular phases of product development at the hardware level.

Table A.1 — Overview of product development at the hardware level

Clause	Objectives	Prerequisites	Work products
5 Initiation of product development at the hardware level	The objective of the initiation of the product development for the hardware is to determine and plan the functional safety activities during the individual subphases of hardware development. This also includes the necessary supporting processes described in ISO 26262-8. This planning of hardware-specific safety activities is included in the safety plan (see ISO 26262-2:2011, 6.4.3 and ISO 26262-4:2011, 5.4).	Project plan (refined) (in accordance with ISO 26262-4:2011, 5.5.1) Safety plan (refined) (in accordance with ISO 26262-4:2011, 5.5.2) Item integration and testing plan (refined) (in accordance with ISO 26262-4:2011, 5.5.5)	5.5 Safety plan (refined)
6 Specification of hardware safety requirements	The first objective of this clause is to specify the hardware safety requirements. They are derived from the technical safety concept and system design specification. The second objective is to verify that the hardware safety requirements are consistent with the technical safety concept and the system design specification. A further objective of this phase is to detail the hardware-software interface (HSI) specification initiated in ISO 26262-4:2011, Clause 7.	Safety plan (refined) (in accordance with 5.5) Technical safety concept (in accordance with ISO 26262-4:2011, 7.5.1) System design specification (in accordance with ISO 26262-4:2011, 7.5.2) Hardware software interface specification (in accordance with ISO 26262-4:2011, 7.5.3)	6.5.1 Hardware safety requirements specification (including test and qualification criteria) 6.5.2 Hardware-software interface specification (refined) 6.5.3 Hardware safety requirements verification report
7 Hardware design	The first objective of this clause is to design the hardware with respect to the system design specification and the hardware safety requirements. The second objective of this clause is to verify the hardware design against the system design specification and the hardware safety requirements.	Hardware safety requirements specification (in accordance with 6.5.1) Hardware-software interface specification (refined) (in accordance with 6.5.2) System design specification (in accordance with ISO 26262-4:2011, 7.5.2) Safety plan (refined) (in accordance with 5.5)	7.5.1 Hardware design specification 7.5.2 Hardware safety analysis report 7.5.3 Hardware design verification report 7.5.4 Specification of requirements related to production, operation, service and decommissioning

Table A.1 (*continued*)

Clause	Objectives	Prerequisites	Work products
8 Evaluation of the hardware architectural metrics	The objective of this clause is to evaluate the hardware architecture of the item against the requirements for fault handling as represented by the hardware architectural metrics.	Hardware safety requirements specification (in accordance with 6.5.1) Hardware design specification (in accordance with 7.5.1) Hardware safety analysis report (in accordance with 7.5.2)	8.5.1 Analysis of the effectiveness of the architecture of the item to cope with the random hardware failures 8.5.2 Review report of evaluation of the effectiveness of the architecture of the item to cope with the random hardware failures
9 Evaluation of safety goal violations due to random HW failures	The objective of the requirements in this clause is to make available criteria that can be used in a rationale that the residual risk of a safety goal violation, due to random hardware failures of the item, is sufficiently low.	Hardware safety requirements specification (in accordance with 6.5.1) Hardware design specification (in accordance with 7.5.1) Hardware safety analysis report (in accordance with 7.5.2)	9.5.1 Analysis of safety goal violations due to random hardware failures 9.5.2 Specification of dedicated measures for hardware, if needed, including the rationale regarding the effectiveness of the dedicated measures 9.5.3 Review report of evaluation of safety goal violations due to random hardware failures
10 Hardware integration and testing	The objective of this clause is to ensure, by testing, the compliance of the developed hardware with the hardware safety requirements. The requirements in 10.4.1 to 10.4.6 apply to the hardware of an element.	Safety plan (refined) (in accordance with 5.5) Item integration and testing plan (refined) (in accordance with ISO 26262-4:2011, 5.5.5) Hardware safety requirements specification (in accordance with 6.5.1) Hardware design specification (in accordance with 7.5.1)	10.5 Hardware integration and testing report

Annex B
(informative)

Failure mode classification of a hardware element

Failure modes of a hardware element can be classified as shown in Figure B.1. The flow diagram shown in Figure B.2 describes how a failure mode of a hardware element can be placed into one of these classifications.

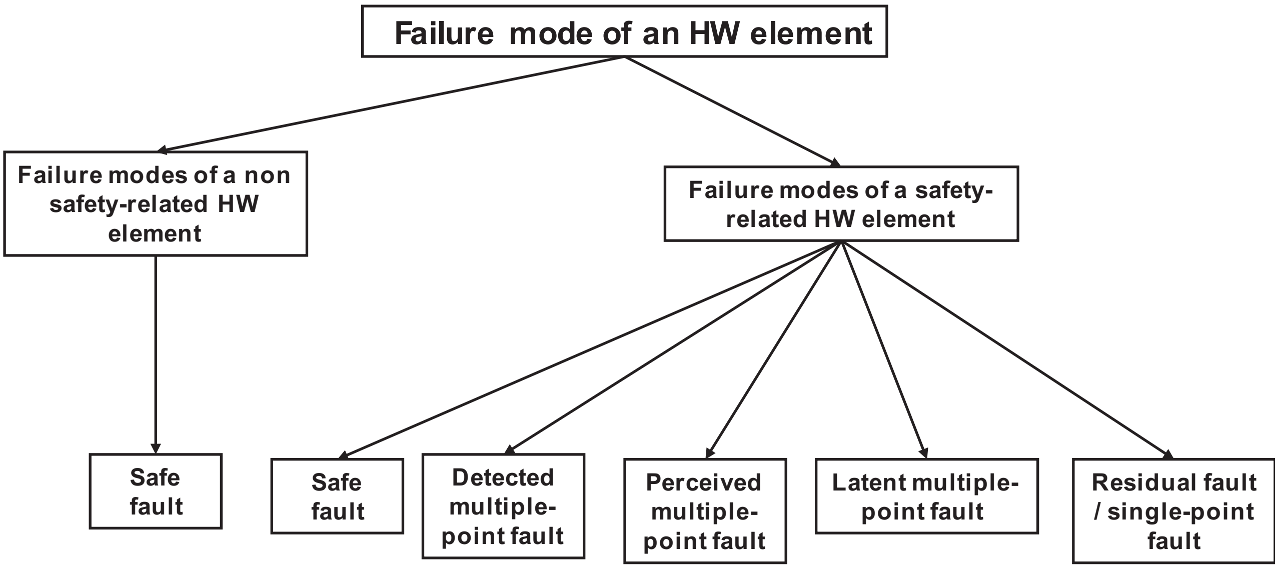
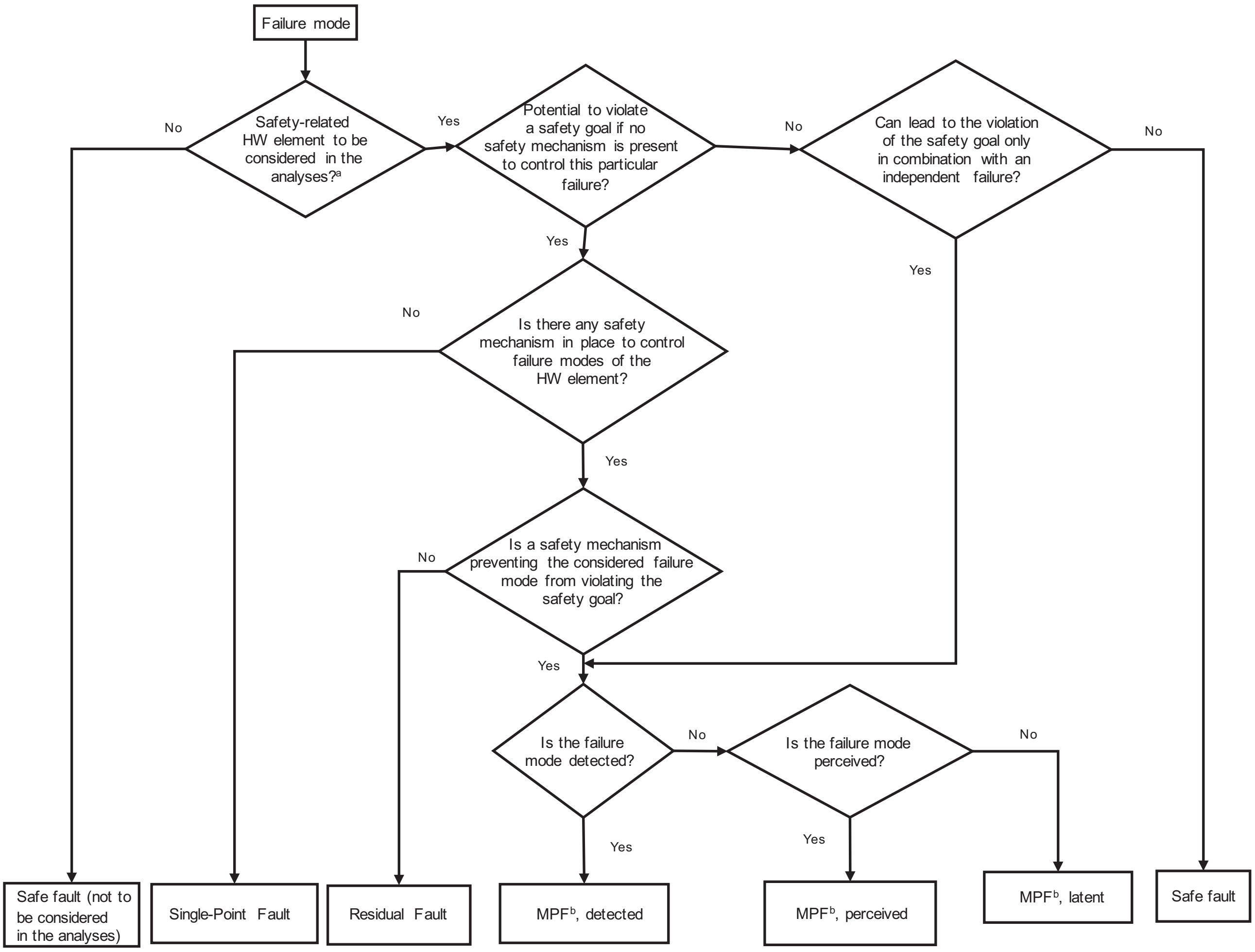

Failure mode of an HW element
Failure modes of a non safety-related HW element
Failure modes of a safety-related HW element
Safe fault
Safe fault
Detected multiple-point fault
Perceived multiple-point fault
Latent multiple-point fault
Residual fault / single-point fault

Figure B.1 — Failure mode classifications of a hardware element


Failure mode
Safety-related HW element to be considered in the analyses?[a]
No
Yes
Potential to violate a safety goal if no safety mechanism is present to control this particular failure?
No
Can lead to the violation of the safety goal only in combination with an independent failure?
No
Yes
Yes
No
Is there any safety mechanism in place to control failure modes of the HW element?
Yes
No
Is a safety mechanism preventing the considered failure mode from violating the safety goal?
Yes
Is the failure mode detected?
No
Is the failure mode perceived?
No
Yes
Yes
Safe fault (not to be considered in the analyses)
Single-Point Fault
Residual Fault
MPF[b], detected
MPF[b], perceived
MPF[b], latent
Safe fault

[a] The elements with failures that do not significantly increase the probability of the violation of a safety goal can be omitted from the analyses and their failure modes can be classified as safe faults, e.g. hardware elements whose faults only contribute to multiple-point failures with $n>2$ unless shown to be relevant in the technical safety concept.

[b] MPF stands for multiple-point fault.

NOTE 1 Multiple-point faults with $n>2$ are considered as safe faults unless shown to be relevant in the technical safety concept.

NOTE 2 The same fault can be placed in different classes when being considered for different safety goals.

Figure B.2 — Example of flow diagram for failure mode classification

Annex C
(normative)

Hardware architectural metrics

C.1 Fault classification and diagnostic coverage

C.1.1 This requirement applies to ASIL (B), C, and D of the safety goal. Hardware architectural metrics shall be defined for the hardware of an item and shall address only safety-related hardware elements that have the potential to contribute significantly to the violation of the safety goal.

EXAMPLE Hardware elements whose faults are multiple-point faults with n>2 can be omitted from the calculations unless shown to be relevant in the technical safety concept.

C.1.2 This requirement applies to ASIL (B), C, and D of the safety goal. Each fault occurring in a safety-related hardware element shall be classified, as illustrated in Figure B.1, as:

a) single-point fault;

b) residual fault;

EXAMPLE A hardware element that can have “open”, “short to ground”, and “short to high” faults, but only the “open” and “short to ground” faults are covered by safety mechanisms. The “short to high” fault is a residual fault, since it is not covered by a safety mechanism, if it leads to the violation of the specified safety goal.

c) multiple-point fault;

d) safe fault.

Figure C.1 gives a graphical representation of fault classification of safety-related hardware elements of an item:

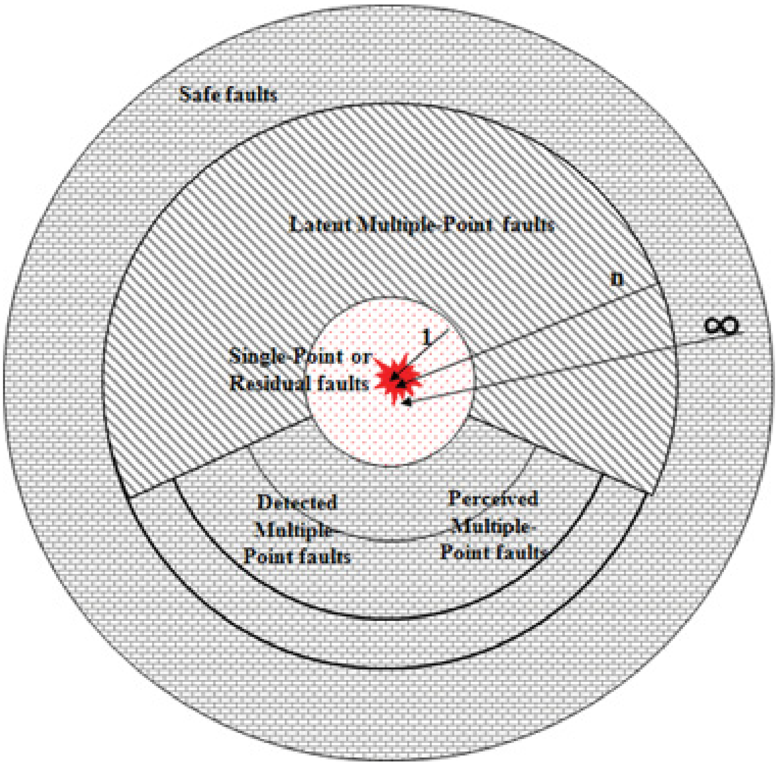

Safe faults
Latent Multiple-Point faults
n
∞
1
Single-Point or Residual faults
Detected Multiple-Point faults
Perceived Multiple-Point faults

Figure C.1 — Fault classification of safety-related hardware elements of an item

In this graphical representation:

- the distance n represents the number of independent faults present at the same time that cause a violation of the safety goal (n = 1 for single-point or residual faults, n = 2 for dual-point faults, etc.);
- faults with distance equal to n are located in the area between the circles n and n-1; and
- multiple-point faults of distance strictly higher than n=2 are to be considered as safe faults unless shown to be relevant in the technical safety concept.

NOTE 1 In the case of a transient fault, for which a safety mechanism restores the item to a fault free state, such a fault can be considered as a detected multiple-point fault even if the driver is never informed of its existence.

EXAMPLE In the case of an error correction code used to protect a memory against transient faults, the item is restored to a fault free state if the safety mechanism – in addition to delivering a correct value to the CPU - repairs the content of the flipped bit inside the memory array (e.g. by writing back the corrected value).

The failure rate, λ, of each safety-related hardware element can therefore be expressed according to Equation (C.1) (assuming all failures are independent and follow the exponential distribution), as follows:

$$\lambda = \lambda_{\mathrm{SPF}} + \lambda_{\mathrm{RF}} + \lambda_{\mathrm{MPF}} + \lambda_{\mathrm{S}} \qquad \text{(C.1)}$$

where

λ_{SPF} is the failure rate associated with hardware element single-point faults;

λ_{RF} is the failure rate associated with hardware element residual faults;

λ_{MPF} is the failure rate associated with hardware element multiple-point faults;

λ_{S} is the failure rate associated with hardware element safe faults.

The failure rate associated with hardware element multiple-point faults, λ_{MPF}, can be expressed according to Equation (C.2), as follows:

$$\lambda_{\text{MPF}} = \lambda_{\text{MPF,DP}} + \lambda_{\text{MPF,L}} \tag{C.2}$$

where

$\lambda_{\text{MPF,DP}}$ is the failure rate associated with hardware element perceived or detected multiple-point faults;

$\lambda_{\text{MPF,L}}$ is the failure rate associated with hardware element latent faults.

The failure rate assigned to residual faults can be determined using the diagnostic coverage of safety mechanisms that avoid single-point faults of the hardware element. Equation (C.3) gives a conservative estimation of the failure rate associated with the residual faults:

$$\begin{aligned} K_{\text{DC,RF}} &= \left(1 - \frac{\lambda_{\text{RF,est}}}{\lambda}\right) \times 100 \\ \lambda_{\text{RF}} \le \lambda_{\text{RF,est}} &= \lambda \times \left(1 - \frac{K_{\text{DC,RF}}}{100}\right) \end{aligned} \tag{C.3}$$

where

$\lambda_{\text{RF,est}}$ is the estimated failure rate with respect to residual faults;

$K_{\text{DC,RF}}$ is the diagnostic coverage with respect to residual faults, expressed as a percentage.

The failure rate assigned to latent faults can be determined using the diagnostic coverage of safety mechanisms that avoid latent faults of the hardware element. Equation (C.4) gives a conservative estimation of the failure rate associated with latent faults:

$$\begin{aligned} K_{\text{DC,MPF,L}} &= \left(1 - \frac{\lambda_{\text{MPF,L,est}}}{\lambda}\right) \times 100 \\ \lambda_{\text{MPF,L}} \le \lambda_{\text{MPF,L,est}} &= \lambda \times \left(1 - \frac{K_{\text{DC,MPF,L}}}{100}\right) \end{aligned} \tag{C.4}$$

where

$\lambda_{\text{MPF,L,est}}$ is the estimated failure rate with respect to latent faults;

$K_{\text{DC,MPF,L}}$ is the diagnostic coverage with respect to latent faults, expressed as a percentage.

NOTE 2 For this purpose, Annex D can be used as a basis for diagnostic coverage (DC) with the claimed DC supported by a proper rationale.

NOTE 3 If the above estimations are considered too conservative, then a detailed analysis of the failure modes of the hardware element can classify each failure mode into one of the fault classes (single-point faults, residual faults, latent, detected or perceived multiple-point faults or safe faults) with respect to the specified safety goal and determine the failure rates apportioned to the failure modes. Annex B describes a flow diagram that can be used to make the fault classification.

C.2 Single-point fault metric

C.2.1 This metric reflects the robustness of the item to single-point and residual faults either by coverage from safety mechanisms or by design (primarily safe faults). A high single-point fault metric implies that the proportion of single-point faults and residual faults in the hardware of the item is low.

C.2.2 This requirement applies to ASIL (B), C, and D of the safety goal. The calculation in Equation (C.5) shall be used to determine the single-point fault metric:

$$1-\frac{\sum\limits_{\text{SR,HW}}(\lambda_{\text{SPF}}+\lambda_{\text{RF}})}{\sum\limits_{\text{SR,HW}}\lambda}=\frac{\sum\limits_{\text{SR,HW}}(\lambda_{\text{MPF}}+\lambda_{\text{S}})}{\sum\limits_{\text{SR,HW}}\lambda} \tag{C.5}$$

where $\sum\limits_{\text{SR,HW}}\lambda_{\text{x}}$ is the sum of λ_{x} of the safety-related hardware elements of the item to be considered for the metrics.

NOTE 1 Only the safety-related hardware elements of the item whose failures have the potential to contribute significantly to the violation of the safety goal are considered for this metric.

EXAMPLE Hardware elements whose faults are multiple-point faults with $n > 2$ can be omitted from the calculations unless shown to be relevant in the technical safety concept.

NOTE 2 Figure C.2 gives a graphical representation of the single-point fault metric.

NOTE 3 An example of calculation of “latent-fault metric” is given in Annex E.

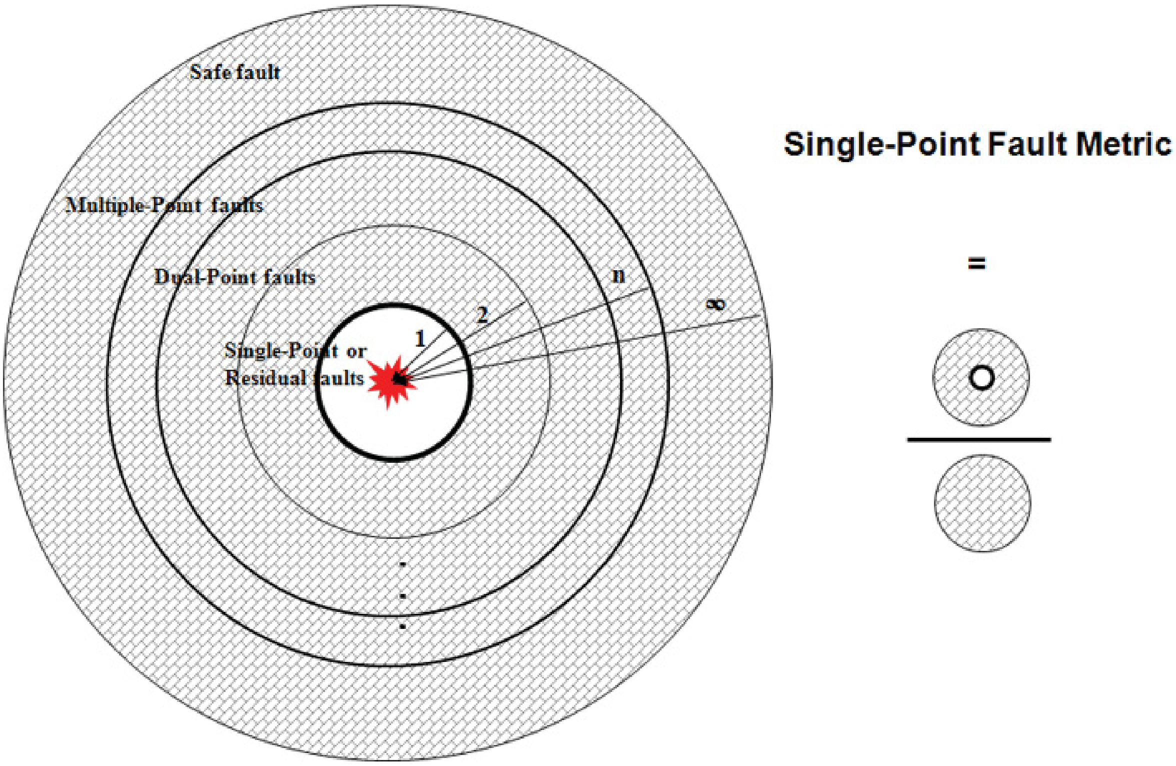


Figure C.2 — Graphical representation of the single-point fault metric

C.3 Latent-fault metric

C.3.1 This metric reflects the robustness of the item to latent faults either by coverage of faults in safety mechanisms or by the driver recognizing that the fault exists before the violation of the safety goal, or by design (primarily safe faults). A high latent-fault metric implies that the proportion of latent faults in the hardware is low.

C.3.2 This requirement applies to ASIL (B), (C), and D of the safety goal. The calculation in Equation (C.6) shall be used to determine the latent-fault metric:

$$1-\frac{\sum\limits_{\mathrm{SR,HW}}(\lambda_{\mathrm{MPF,latent}})}{\sum\limits_{\mathrm{SR,HW}}(\lambda-\lambda_{\mathrm{SPF}}-\lambda_{\mathrm{RF}})}=\frac{\sum\limits_{\mathrm{SR,HW}}(\lambda_{\mathrm{MPF,perceived\ or\ detected}}+\lambda_{\mathrm{S}})}{\sum\limits_{\mathrm{SR,HW}}(\lambda-\lambda_{\mathrm{SPF}}-\lambda_{\mathrm{RF}})} \qquad \text{(C.6)}$$

where $\sum\limits_{\mathrm{SR,HW}}\lambda_{\mathrm{x}}$ is the sum of λ_{x} of the safety-related hardware elements of the item to be considered for the metrics.

NOTE 1 Only the safety-related hardware elements of the item whose failures have the potential to contribute significantly to the violation of the safety goal are considered for this metric.

EXAMPLE Hardware elements whose faults are multiple-point faults with $n > 2$ can be omitted from the calculations unless shown to be relevant in the technical safety concept.

NOTE 2 Figure C.3 gives a graphical representation of the latent-fault metric.

NOTE 3 An example of calculation of “latent-fault metric” is given in Annex E.

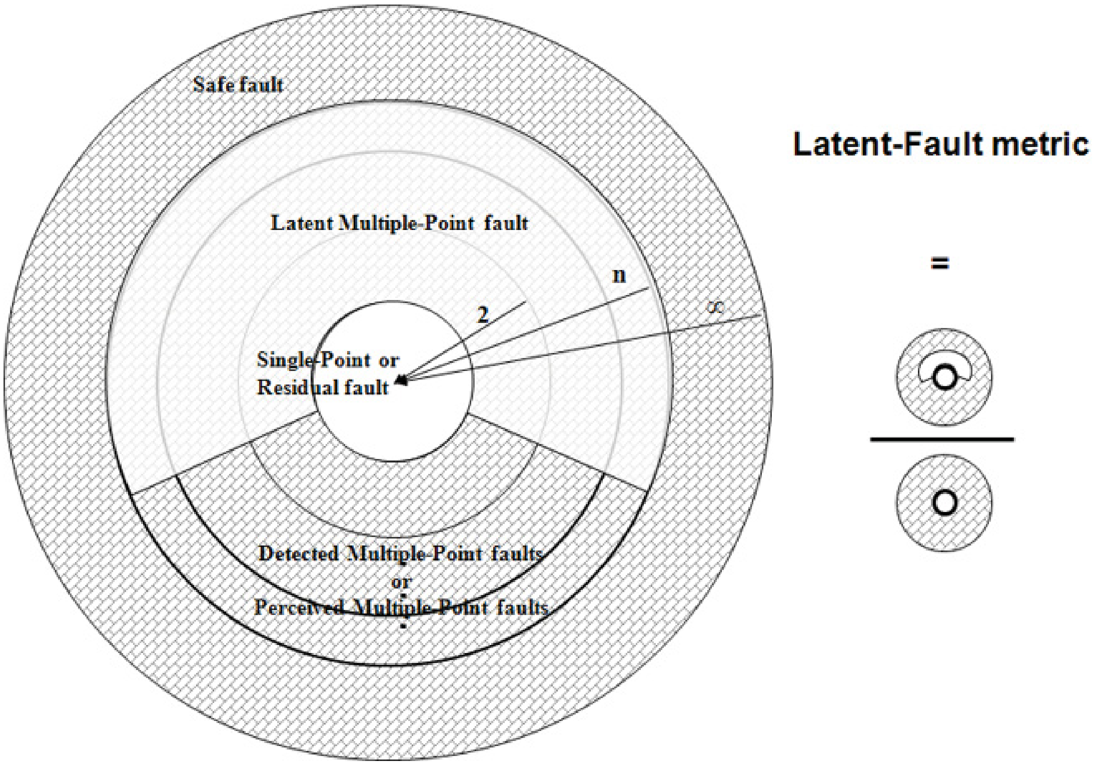

Safe fault
Latent Multiple-Point fault
n
2
∞
Single-Point or Residual fault
Detected Multiple-Point faults
or
Perceived Multiple-Point faults
Latent-Fault metric
=

Figure C.3 — Graphical representation of the latent-fault metric

Annex D
(informative)

Evaluation of the diagnostic coverage

D.1 General

This annex is intended to be used as:

a) an evaluation of the diagnostic coverage to produce a rationale for:

   1) the compliance with the single-point fault and latent-fault metrics defined in Clause 8;

   2) the compliance with the evaluation of the safety goal violations due to random hardware failures as defined in Clause 9;

b) a guideline in order to choose appropriate safety mechanisms to be implemented in the E/E architecture to detect failures of elements.

Figure D.1 shows the generic hardware of an embedded system. Typical faults or failures of the hardware elements of this system are shown in Table D.1 which also includes guidelines for diagnostic coverage. Each element listed in the leftmost column is associated with one or more faults which are captured in the columns to the right of the element. The listing does not claim exhaustiveness and can be adjusted based on additional known faults or depending on the application.

Additional detail on the safety mechanisms associated with these element faults are referenced in each row (Tables D.2 to D.14). The effectiveness of these typical safety mechanisms for the given elements is categorized according to their ability to cover the listed faults to achieve low, medium or high diagnostic coverage of the element. These low, medium and high diagnostic coverage rankings correspond to typical coverage levels at 60 %, 90 % or 99 %, respectively.

The assignment of the faults and their corresponding safety mechanisms to diagnostic coverage levels can vary from that listed in Table D.1 depending on:

c) variations in the source of the fault type detected by the diagnostic

d) the effectiveness of the safety mechanism

e) the specific implementation of the safety mechanism

f) the execution timing of the safety mechanism (periodicity)

g) the hardware technologies implemented in the system

h) the probability of the failure modes, based on hardware in the system

i) a more detailed analysis of the faults and their classification into several subclasses with different diagnostic coverage levels.

In summary, Table D.1 provides guidelines which are adapted based on analysis of the system elements.

These guidelines do not address specific constraints that can be specified in the safety concepts in order to avoid the violation of the safety goals. These constraints, such as timing aspects (periodicity of diagnostic) for example, are not considered when evaluating the generic typical diagnostic coverage by the safety mechanism. They will be considered when evaluating the specific diagnostic coverage by a safety mechanism used in the item to avoid the violation of the safety goals.

EXAMPLE A safety mechanism can have a high generic typical diagnostic coverage in this annex but if the diagnostic test interval used is longer than the diagnostic test interval needed to comply with the relevant fault tolerant time interval, the specific diagnostic coverage with respect to the avoidance of violation of the safety goal, will be much lower.

Therefore Tables D.1 to D.14 can be used as a starting point to evaluate the diagnostic coverage of these safety mechanisms with the claimed DC supported by a proper rationale. In addition, the given information is intended to help define the faults or failure modes of the element; however the relevant failure modes are ultimately dependent on the application in which the elements are used.

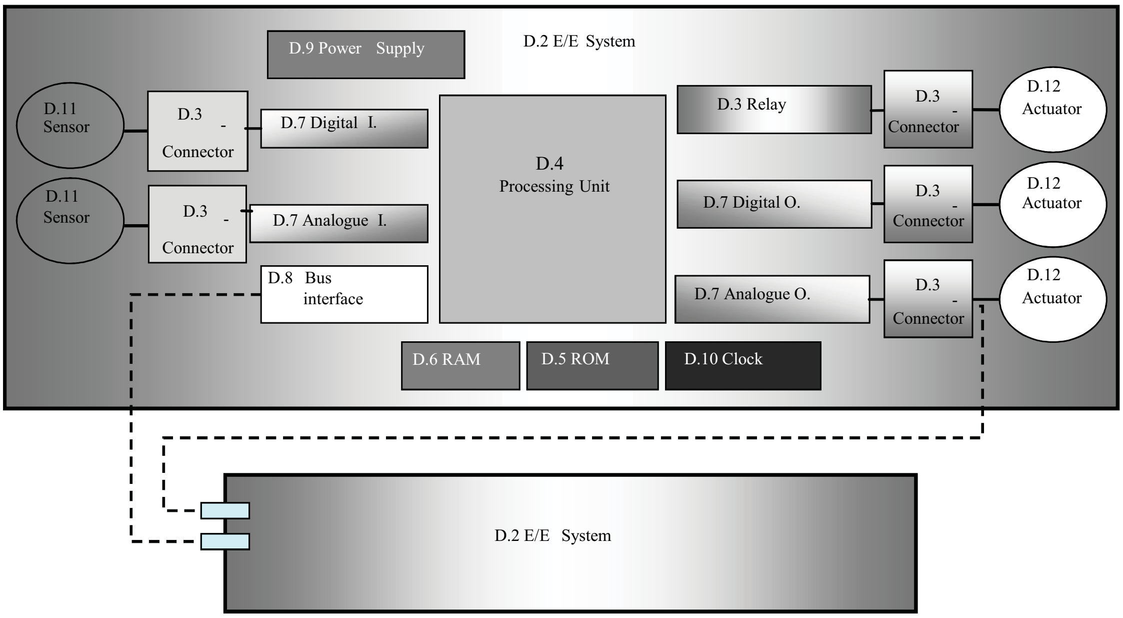


Figure D.1 — Generic hardware of a system

Tables D.2 to D.14 support the information of Table D.1 by giving guidelines on techniques for diagnostic tests. Tables D.1 to D.14 are not exhaustive and other techniques can be used, provided evidence is available to support the claimed diagnostic coverage. If justified, higher diagnostic coverage can be estimated, up to 100 % for simple or complex elements.

Table D.1 — Analyzed faults or failures modes in the derivation of diagnostic coverage

Element	See Tables	Analyzed failure modes for 60 %/90 %/99 % DC		
		Low (60 %)	Medium (90 %)	High (99 %)
General elements				
E.E Systems	D.2	No generic fault model available. Detailed analysis necessary.	No generic fault model available. Detailed analysis necessary.	No generic fault model available. Detailed analysis necessary.
Electrical elements				
Relays	D.3	Does not energize or de-energize. Welded contacts	Does not energize or de-energize. Individual contacts welded	Does not energize or de-energize. Individual contacts welded
Harnesses including splice and connectors		Open Circuit Short Circuit to Ground	Open Circuit Short Circuit to Ground (d.c Coupled) Short Circuit to Vbat Short Circuit between neighbouring pins	Open Circuit Contact Resistance Short Circuit to Ground (d.c Coupled) Short Circuit to Vbat Short Circuit between neighbouring pins Resistive drift between pins
Sensors including signal switches	D.11	No generic fault model available. Detailed analysis necessary. Typical failure modes to be covered include Out-of-range Stuck in range	No generic fault model available. Detailed analysis necessary. Typical failure modes to be covered include Out-of-range Offsets Stuck in range	No generic fault model available. Detailed analysis necessary. Typical failure modes to be covered include Out-of-range Offsets Stuck in range Oscillations
Final elements (actuators, lamps, buzzer, screen…)	D.12	No generic fault model available. Detailed analysis necessary.	No generic fault model available. Detailed analysis necessary.	No generic fault model available. Detailed analysis necessary.
General semiconductor elements				
Power supply	D.9	Under and over Voltage	Drift Under and over Voltage	Drift and oscillation Under and over Voltage Power spikes
Clock	D.10	Stuck-at[a]	d.c. fault model[b]	d.c. fault model[b] Incorrect frequency Period jitter
Non-volatile memory	D.5	Stuck-at[a] for data and addresses and control interface, lines and logic	d.c. fault model[b] for data and addresses (includes address lines within same block) and control interface, lines and logic	d.c. fault model[b] for data, addresses (includes address lines within same block) and control interface, lines and logic

Table D.1 (*continued*)

Element		See Tables	Analyzed failure modes for 60 %/90 %/99 % DC		
			Low (60 %)	**Medium (90 %)**	**High (99 %)**
Volatile memory		D.6	Stuck-at[a] for data, addresses and control interface, lines and logic	d.c. fault model[b] for data, addresses (includes address lines within same block and inability to write to cell) and control interface, lines and logic Soft error model[c] for bit cells	d.c. fault model[b] for data, addresses (includes address lines within same block and inability to write to cell) and control interface, lines and logic Soft error model[c] for bit cells
Digital I/O		D.7	Stuck-at[a] (including signal lines outside of the microcontroller)	d.c. fault model[b] (including signal lines outside of the microcontroller)	d.c. fault model[b] (including signal lines outside of the microcontroller) Drift and oscillation
Analogue I/O			Stuck-at[a] (including signal lines outside of the microcontroller)	d.c. fault model[b] (including signal lines outside of the microcontroller) Drift and oscillation	d.c. fault model[b] (including signal lines outside of the microcontroller) Drift and oscillation
Specific semiconductor elements					
Processing units	ALU - Data Path	D.4/D.13	Stuck-at[a]	Stuck-at[a] at gate level	d.c. fault model[b] Soft error model[c] (for sequential parts)
	Registers (general purpose registers bank, DMA transfer registers…), internal RAM	D.4	Stuck-at[a]	Stuck-at[a] at gate level Soft error model[c]	d.c. fault model[b] including no, wrong or multiple addressing of registers Soft error model[c]
	Address calculation (Load/Store Unit, DMA addressing logic, memory and bus interfaces)	D.4/D.5/D.6	Stuck-at[a]	Stuck-at[a] at gate level Soft error model[c] (for sequential parts)	d.c. fault model[b] including no, wrong or multiple addressing Soft error model[c] (for sequential parts)
	Interrupt handling	D.4/D.10	Omission of or continuous interrupts	Omission of or continuous interrupts Incorrect interrupt executed	Omission of or continuous interrupts Incorrect interrupt executed Wrong priority Slow or interfered interrupt handling causing missed or delayed interrupts service
	Control logic (Sequencer, coding and execution logic including flag registers and stack control)	D.4/D.10	No code execution Execution too slow Stack overflow/underflow	Wrong coding or no execution Execution too slow Stack overflow/underflow	Wrong coding, wrong or no execution Execution out of order Execution too fast or too slow Stack overflow/underflow
	Configuration Registers	D.4	—	Stuck-at[a] wrong value	Corruption of registers (soft errors) Stuck-at[a] fault model
	Other sub-elements not belonging to previous classes	D.4/D.13	Stuck-at[a]	Stuck-at[a] at gate level	d.c. fault model[b] Soft error model[c] (for sequential part)

Table D.1 (*continued*)

Element		See Tables	Analyzed failure modes for 60 %/90 %/99 % DC		
			Low (60 %)	**Medium (90 %)**	**High (99 %)**
Communication	On-chip communication including bus-arbitration	D.14	Stuck-at[a] (data, control, address and arbitration signals)	d.c. fault model[b] (data, control, address and arbitration signals) Time out No or continuous arbitration	d.c. fault model[b] (data, control, address and arbitration signals) Time out No or continuous or wrong arbitration Soft errors (for sequential part)
	Data transmission (to be analysed with ISO 26262-6:2011, Annex D)	D.8	Failure of communication peer Message corruption Message delay Message loss Unintended message repetition	Previous + Resequencing Insertion of message	Previous + Masquerading

NOTE 1 Higher DC can be claimed based on analysis. Likewise, lower coverage would result if the dominant failure mode is not listed.

NOTE 2 Transient faults are considered when shown to be relevant due, for instance, to the technology used.

NOTE 3 Failure modes for Processing Units can be adjusted to recognize a.c. fault models, such as transition faults (slow to rise and slow to fall nodes at application frequency) and path delays. Faults of this type are expected to be more prevalent with smaller process geometries. Usually tests for these types of faults are done at start-up, or power-down, or both, due to their intrusive nature and their ability to detect failures early with margin tests. Since they are hard to quantify, these failure modes are generally not included in failure rate calculations.

NOTE 4 If properly exercised, methods derived from stuck-at simulations (e.g. N-detect testing), but executed at application conditions, are known to be effective for d.c. fault and transition models as well.

[a] "Stuck-at": is a fault category that can be described with continuous "0" or "1" or "on" at the pins of an element. It is valid only for elements which have element level pin interfaces.

[b] "d.c. fault model" ("direct current fault model") includes the following failure modes: stuck-at faults, stuck-open, open or high impedance outputs, as well as short circuits between signal lines. It is not intended here to require an exhaustive analysis, for example to require the exhaustive analysis of bridging faults that can affect any theoretical combination of any signal inside a microcontroller or in a complex PCB. The analysis focuses on main signals or on very highly coupled interconnections identified with a layout level analysis.

[c] "soft error model": soft errors (e.g. bit flips) are the results of transient faults caused by alpha particles from package decay, neutrons, etc. These transient faults are also referred as Single Event Upset (SEU) and Single Event Transient (SET).

Table D.2 — Systems

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	D.2.1.1	Low	Depends on diagnostic coverage of failure detection
Comparator	D.2.1.2	High	Depends on the quality of the comparison
Majority voter	D.2.1.3	High	Depends on the quality of the voting
Dynamic principles	D.2.2.1	Medium	Depends on diagnostic coverage of failure detection
Analogue signal monitoring in preference to digital on/off states	D.2.2.2	Low	—
Self-test by software cross exchange between two independent units)	D.2.3.3	Medium	Depends on the quality of the self test

Table D.3 — Electrical elements

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	D.2.1.1	High	Depends on diagnostic coverage of failure detection
NOTE This table deals only with safety mechanisms dedicated to electrical elements. General techniques like a technique based on a data comparison (see D.2.1.2) are also able to detect failures of electrical elements but are not integrated in this table (already included in Table D.2).			

Table D.4 — Processing units

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Self-test by software: limited number of patterns (one channel)	D.2.3.1	Medium	Depends on the quality of the self test
Self-test by software cross exchange between two independent units	D.2.3.3	Medium	Depends of the quality of the self test
Self-test supported by hardware (one-channel)	D.2.3.2	Medium	Depends on the quality of the self test
Software diversified redundancy (one hardware channel)	D.2.3.4	High	Depends on the quality of the diversification. Common mode failures can reduce diagnostic coverage
Reciprocal comparison by software	D.2.3.5	High	Depends on the quality of the comparison
HW redundancy (e.g. Dual Core Lockstep, asymmetric redundancy, coded processing)	D.2.3.6	High	It depends on the quality of redundancy. Common mode failures can reduce diagnostic coverage
Configuration Register Test	D.2.3.7	High	Configuration registers only
Stack over/under flow Detection	D.2.3.8	Low	Stack boundary test only
Integrated Hardware consistency monitoring	D.2.3.9	High	Coverage for illegal hardware exceptions only
NOTE This table deals only with safety mechanisms dedicated to processing units. General techniques like one based on data comparison (see D.2.1.2) are also able to detect failures of electrical elements but are not integrated in this table (already included in Table D.2).			

Table D.5 — Non-volatile memory

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Parity bit	D.2.5.2	Low	—
Memory monitoring using error-detection-correction codes (EDC)	D.2.4.1	High	The effectiveness depends on the number of redundant bits. Can be used to correct errors
Modified checksum	D.2.4.2	Low	Depends on the number and location of bit errors within test area
Memory Signature	D.2.4.3	High	—
Block replication	D.2.4.4	High	—

Table D.6 — Volatile memory

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
RAM pattern test	D.2.5.1	Medium	High coverage for stuck-at failures. No coverage for linked failures. Can be appropriate to run under interrupt protection
RAM March test	D.2.5.3	High	Depends on the write read order for linked cell coverage. Test generally not appropriate for run time
Parity bit	D.2.5.2	Low	—
Memory monitoring using error-detection-correction codes (EDC)	D.2.4.1	High	The effectiveness depends on the number of redundant bits. Can be used to correct errors
Block replication	D.2.4.4	High	Common failure modes can reduce diagnostic coverage
Running checksum/CRC	D.2.5.4	High	The effectiveness of the signature depends on the polynomial in relation to the block length of the information to be protected. Care needs to be taken so that values used to determine checksum are not changed during checksum calculation Probability is 1/maximum value of checksum if random pattern is returned

Table D.7 — Analogue and digital I/O

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring (Digital I/O)[a]	D.2.1.1	Low	Depends on diagnostic coverage of failure detection
Test pattern	D.2.6.1	High	Depends on type of pattern
Code protection for digital I/O	D.2.6.2	Medium	Depends on type of coding
Multi-channel parallel output	D.2.6.3	High	—
Monitored outputs	D.2.6.4	High	Only if dataflow changes within diagnostic test interval
Input comparison/voting (1oo2, 2oo3 or better redundancy)	D.2.6.5	High	Only if dataflow changes within diagnostic test interval

[a] Digital I/O can be periodic.

Table D.8 — Communication bus (serial, parallel)

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
One-bit hardware redundancy	D.2.7.1	Low	—
Multi-bit hardware redundancy	D.2.7.2	Medium	—
Read back of sent message	D.2.7.9	Medium	—
Complete hardware redundancy	D.2.7.3	High	Common mode failures can reduce diagnostic coverage
Inspection using test patterns	D.2.7.4	High	—
Transmission redundancy	D.2.7.5	Medium	Depends on type of redundancy. Effective only against transient faults
Information redundancy	D.2.7.6	Medium	Depends on type of redundancy
Frame counter	D.2.7.7	Medium	—
Timeout monitoring	D.2.7.8	Medium	—
Combination of information redundancy, frame counter and timeout monitoring	D.2.7.6, D.2.7.7 and D.2.7.8	High	For systems without hardware redundancy or test patterns, high coverage can be claimed for the combination of these safety mechanisms

Table D.9 — Power supply

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Voltage or current control (input)	D.2.8.1	Low	—
Voltage or current control (output)	D.2.8.2	High	—

Table D.10 — Program sequence monitoring/Clock

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Watchdog with separate time base without time-window	D.2.9.1	Low	—
Watchdog with separate time base and time-window	D.2.9.2	Medium	Depends on time restriction for the time-window
Logical monitoring of program sequence	D.2.9.3	Medium	Only effective against clock failures if external temporal events influence the logical program flow. Provides coverage for internal hardware failures (such as interrupt frequency errors) that can cause the software to run out of sequence
Combination of temporal and logical monitoring of program sequence	D.2.9.4	High	—
Combination of temporal and logical monitoring of program sequences with time dependency	D.2.9.5	High	Provides coverage for internal hardware failures that can cause the software to run out of sequence. When implemented with asymmetrical designs, provides coverage regarding communication sequence between main and monitoring device NOTE Method to be designed to account for execution jitter from interrupts, CPU loading, etc.

Table D.11 — Sensors

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	D.2.1.1	Low	Depends on diagnostic coverage of failure detection
Test pattern	D.2.6.1	High	—
Input comparison/voting (1oo2, 2oo3 or better redundancy)	D.2.6.5	High	Only if dataflow changes within diagnostic test interval
Sensor valid range	D.2.10.1	Low	Detects shorts to ground or power and some open circuits
Sensor correlation	D.2.10.2	High	Detects in range failures
Sensor rationality check	D.2.10.3	Medium	—

Table D.12 — Actuators

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
Failure detection by on-line monitoring	D.2.1.1	Low	Depends on diagnostic coverage of failure detection
Test pattern	D.2.6.1	High	—
Monitoring (i.e. coherence control)	D.2.11.1	High	Depends on diagnostic coverage of failure detection

Table D.13 — Combinatorial and sequential logic

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable (d.c. fault model)	Notes
Self-test by software	D.2.3.1	Medium	—
Self-test supported by hardware (one-channel)	D.2.3.2	High	Effectiveness depends on the type of self-test. Gate level is an appropriate level for this test

Table D.14 — On-chip communication

Safety mechanism/measure	See overview of techniques	Typical diagnostic coverage considered achievable	Notes
One-bit hardware redundancy	D.2.7.1	Low	—
Multi-bit hardware redundancy	D.2.7.2	Medium	Multi-bit redundancy can achieve high coverage by proper interleaving of data, address and control lines, and if combined with some complete redundancy, e.g. for the arbiter.
Complete hardware redundancy	D.2.7.3	High	Common failure modes can reduce diagnostic coverage
Test pattern	D.2.6.1	High	Depends on type of pattern
NOTE This table provides coverage for communication buses within microprocessors.			

D.2 Overview of techniques for embedded diagnostic self-tests

D.2.1 Electrical

Global objective: To control failures in electromechanical elements.

D.2.1.1 Failure detection by on-line monitoring

NOTE 1 This technique/measure is referenced in Tables D.2, D.3, D.7, D.11 and D.12.

Aim: To detect failures by monitoring the behaviour of the system in response to the normal (on-line) operation.

Description: Under certain conditions, failures can be detected using information about (for example) the time behaviour of the system. For example, if a switch is normally actuated and does not change state at the expected time, a failure will have been detected. It is not usually possible to localize the failure.

NOTE 2 In general, there is no specific hardware element for the realisation of the on-line monitoring diagram. On-line monitoring detects abnormal behaviour of the system with respect to certain conditions of activation. For example, if such parameter is inverted when the vehicle speed is different from zero, then detection of incoherence between this parameter and vehicle speed leads to failure detection.

D.2.1.2 Comparator

NOTE This technique/measure is referenced in Table D.2.

Aim: To detect, as early as possible, (non-simultaneous) failures in independent hardware or software.

Description: The output signals of independent hardware or output information of independent software, are compared cyclically or continuously by a comparator. Detected differences lead to a failure message. For instance: two processing units exchange data (including results, intermediate results and test data) reciprocally. A comparison of the data is carried out using software in each unit and detected differences lead to a failure message.

D.2.1.3 Majority voter

NOTE 1 This technique/measure is referenced in Table D.2.

Aim: To detect and mask failures in one of at least three channels.

Description: A voting unit using the majority principle (2 out of 3, 3 out of 3, or m out of n) is used to detect and mask failures.

NOTE 2 Unlike the comparator, the majority voter technique increases the availability by ensuring the functionality of the redundant channel even after the loss of one channel.

D.2.2 Electronic

Global objective: To control failure in solid-state elements.

D.2.2.1 Dynamic principles

NOTE This technique/measure is referenced in Table D.2.

Aim: To detect static failures by dynamic signal processing.

Description: A forced change of otherwise static signals (internally or externally generated) helps to detect static failures in elements. This technique is often associated with electromechanical elements.

D.2.2.2 Analogue signal monitoring in preference to digital on/off states

NOTE This technique/measure is referenced in Table D.2.

Aim: To improve confidence in measured signals.

Description: Wherever there is a choice, analogue signals are used in preference to digital on/off states. For example, trip or safe states are represented by analogue signal levels, usually with signal level tolerance monitoring. In the case of a digital signal, it is possible to monitor it with an analogue input. The technique gives continuity monitoring and a higher level of confidence in the transmitter, reducing the required frequency of the periodic test performed to detect failures of the transmitter sensing function.

D.2.3 Processing units

Global objective: To detect failures in processing units which lead to incorrect results.

D.2.3.1 Self-test by software

NOTE This technique/measure is referenced in Tables D.4 and D.13.

Aim: To detect, as early as possible, failures in the processing unit and other sub-elements consisting of physical storage (for example, registers) or functional units (for example, instruction decoder or an EDC coder/decoder), or both, by means of software.

Description: The failure detection is realised entirely by software which perform self-tests using a data pattern, or set of data patterns, to test the physical storage (for example, data and address registers) or the functional units (for example, the instruction decoder) or both.

EXAMPLE 1 The processing unit is tested for functional correctness by applying at least one pattern per instruction. Instructions not executed in the safety-related path can be omitted from the test but coverage can be limited as not all gates of the processing unit will be tested. In general it is possible that not all dedicated and special purpose registers, core timers, and exceptions can be covered. Coverage for order dependencies, such as pipelines, or timing related fault modes can be limited. Determining the actual coverage of the tested gates (in contrast to covered instructions) typically requires extensive fault simulation. This test provides very limited or no coverage for soft errors.

EXAMPLE 2 In the case of sub-elements like an EDC coder/decoder, the software can read pre-written intentionally corrupted words to test the behaviour of the EDC logic. Corrupted words can also be written by the software test itself if the EDC and memory interface have an HW switch to access both data and code bits. Coverage depends on the amount and richness of patterns. This test provides no coverage for soft errors.

D.2.3.2 Self-test supported by hardware (one-channel)

NOTE This technique/measure is referenced in Tables D.4 and D.13.

Aim: To detect, as early as possible, failures in the processing unit and other sub-elements, using special hardware that increases the speed and extends the scope of failure detection.

Description: Additional special hardware facilities support self-test functions to detect failures in the processing unit and other sub-elements (for example an EDC coder/decoder) at a gate level. The test can achieve high coverage. Typically only run at the initialization or power-down of the processing unit due to its intrusive nature. Typical usage is for multipoint fault detection.

EXAMPLE In the case of sub-elements like an EDC coder/decoder, a special HW mechanism, like a logic BIST, can be added to generate inputs to the coder-decoder and check for expected results. Typically inputs are generated by random pattern generators (e.g. MISR). Its coverage depends on the amount and richness of patterns – but usually the coverage is quite high due to the automatic pattern generation. This test provides no coverage for soft errors.

D.2.3.3 Self-test by software cross exchanged between two independent units

NOTE This technique/measure is referenced in Tables D.2 and D.4.

Aim: To detect, as early as possible, failures in the processing unit consisting of physical storage (for example registers) and functional units (for example, instruction decoder).

Description: The failure detection is realised entirely by means of two or more processing units each executing additional software functions which perform self-tests (for example walking-bit pattern) to test the physical storage (data and address registers) and the functional units (for example instruction decoder). The processing units exchange the results. This test provides very limited or no coverage for soft errors.

D.2.3.4 Software diversified redundancy (one hardware channel)

NOTE 1 This technique/measure is referenced in Table D.4.

Aim: To detect, as early as possible, failures in the processing unit, by dynamic software comparison.

Description: The design consists of two redundant diverse software implementations in one hardware channel. In some cases, using different hardware resources (e.g. different RAM, ROM memory ranges) can increase the diagnostic coverage.

One implementation, referred to as the primary path, is responsible for the calculations that if calculated erroneously can cause a hazard. The second implementation, referred to as the redundant path, is responsible for verifying the primary path's calculations and taking action if a failure is detected. Often the redundant path is implemented using separate algorithm designs and code to provide for software diversity. Once both paths are complete, a comparison of the output data of the two redundant software implementations is carried out. Detected differences lead to a failure message (see Figure D.2). The design includes methods to coordinate the two paths and to resynchronise the paths for transient errors.

Generally the comparison involves some type of hysteresis and filtering to allow for minor differences due to the diverse software paths. Examples of algorithm diversity are: A+B=C versus C-B=A and one path using normal calculations and the other path using two's complement mathematics. A redundant path can be as simple as a magnitude or rate-limit check on the calculation of the primary path.

NOTE 2 Due to the potential common cause failures between the primary and redundant paths, an additional watchdog processor can be used to verify the operation of the primary controller via a question and response diagnostic (see Reference [21]).

Another version of this safety mechanism is to implement the redundant path as an exact copy of the primary path (or to execute the primary path twice). This version, without software redundancy, only provides coverage for soft errors. Medium coverage can be achieved if the code is executed a third time with known inputs generating outputs to be verified versus a set of expected outputs. This technique results in a very easy pass-fail criterion (compared results are expected to agree exactly) and easy implementation (the redundant path does not need to be designed) but the concept includes the preservation of history terms (e.g. dynamic states, integrators, rate-limits, etc.).

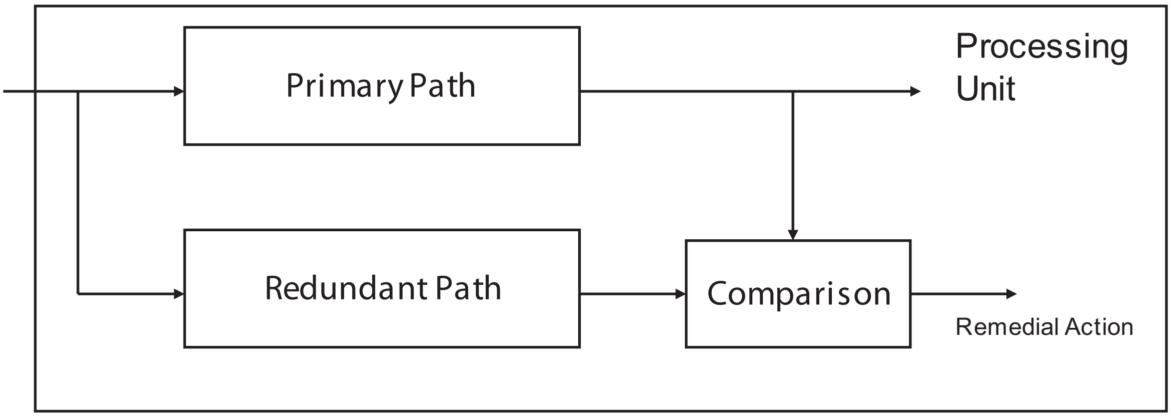


Figure D.2 — Redundant software comparison same processing unit

D.2.3.5 Reciprocal comparison by software in separate processing units

NOTE This technique/measure is referenced in Table D.4.

Aim: To detect, as early as possible, failures in the processing unit, by dynamic software comparison.

Description: Two processing units exchange data (including results, intermediate results and test data) reciprocally. A comparison of the data is carried out using software in each unit and detected differences lead to a failure message (see Figure D.3). This approach allows for hardware and software diversity if different processor types are used as well as separate algorithm designs, code and compilers. The design includes methods to avoid false error detections due to differences between the processors (e.g. loop jitter, communication delays, processor initialization).

Paths can be implemented using separate cores of a dual core processor. In this case, the method includes analysis to understand common cause failures modes, due to the shared die and package, of the two cores.

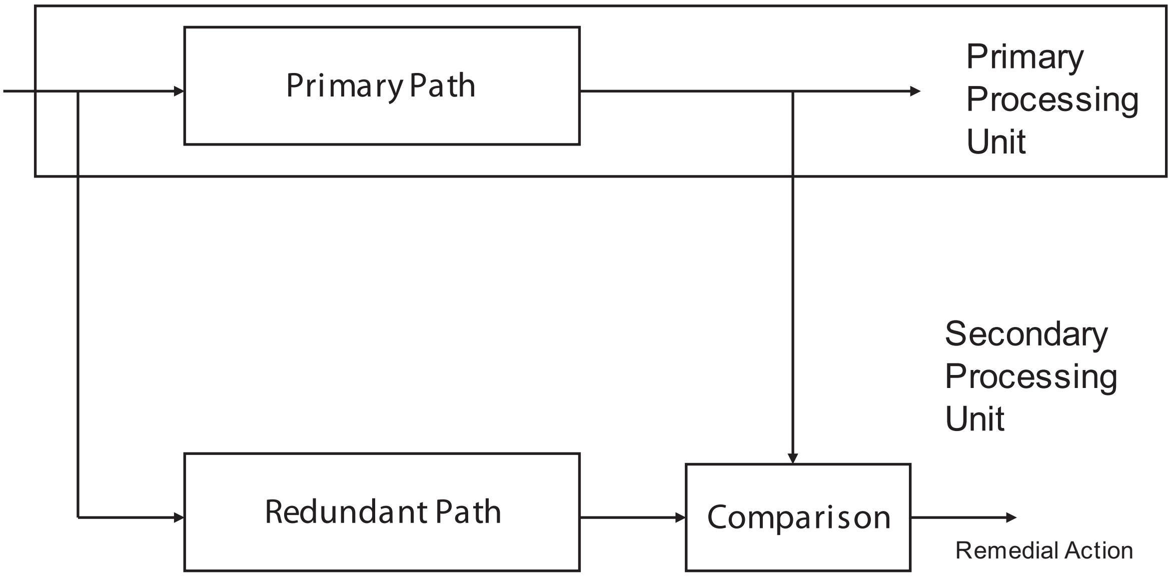


Figure D.3 — Redundant software comparison different processing units

D.2.3.6 HW redundancy (e.g. Dual Core Lockstep, asymmetric redundancy, coded processing)

NOTE This technique/measure is referenced in Table D.4.

Aim: To detect, as early as possible, failures in the processing unit, by step-by-step comparison of internal or external results or both produced by two processing units operating in lockstep.

Description: In one version of this type of diagnostic technique, the Dual Core Lockstep, two symmetrical processing units are contained on one die (see Reference [22]). The processing units run duplicate operations in lockstep (or delayed by a fixed period) and the results are compared. Any mismatch results in an error condition and usually a reset condition. This is very effective for transient errors and for ALU type failures. Depending on the level of redundancy, coverage can be extended to the memory addressing lines and configuration registers. The technique has the advantage that no separate code for the parallel path is required but has the disadvantage of having two processing units providing only the performance of a single processing unit. In good designs, common cause failures are understood and addressed (for example, common clock failure). This approach by itself does not provide diagnostic coverage for systematic errors.

Other types of HW redundancies are possible, such as asymmetric redundancy. In those architectures (e.g. Reference [25]), a diverse and dedicated processing unit is tightly coupled with the main processing units by means of an interface enabling a step-by-step comparison of internal and external results. This is very effective for both a d.c. fault model and for soft errors: moreover, the interface reduces complexity and shortens error detection latency, for example, for faults affecting the processing unit registers bank. No separate code is needed for the parallel path and the dedicated processing unit can be smaller than the main one. The hardware diversity provides effective coverage for common cause failures and systematic failures. The disadvantage of this approach is that it can require a detailed analysis to prove the diagnostic coverage.

Coded processing is also possible: processing units can be designed with special failure-recognising or failure correcting circuit techniques. These approaches can guarantee high coverage for very small processors with limited functionalities or they can be suitable for processor sub-units like ALU (e.g. Reference [26]). Hardware and software coding can be combined using approaches like the Vital Coded Processor (see Reference [27]). A detailed analysis can be needed to prove the diagnostic coverage.

D.2.3.7 Configuration register test

NOTE This technique/measure is referenced in Table D.4.

Aim: To detect, as early as possible, failures in the configuration resisters of a processing unit. Failures can be hardware related (stuck values or soft errors induced bit flips) or software related (incorrect value stored or register corrupted by software error).

Description: Configuration register settings are read and then compared to an encoding of the expected settings (e.g. a mask). If the settings do not match, the registers are reloaded with their intended value. If the error persists for a pre-determined number of checks the fault condition is reported.

D.2.3.8 Stack over/under flow detection

NOTE This technique/measure is referenced in Table D.4.

Aim: To detect, as early as possible, stack over or under flows.

Description: The boundaries of the stack in volatile memory are loaded with predefined values. Periodically, the values are checked and if they have changed, an over or under flow is detected. A test is not needed if writes outside stack boundaries are controlled by a memory management unit.

D.2.3.9 Integrated hardware consistency monitoring

NOTE This technique/measure is referenced in Table D.4.

Aim: To detect, as early as possible, illegal conditions in the processing unit.

Description: Most processors are equipped with mechanisms that trigger hardware exceptions when errors are detected (division by zero and invalid op-codes, for example). Interrupt processing of these errors can then be used to trap these conditions to isolate the system from their effects. Typically, hardware monitoring is used to detect systematic failures but can also be used to detect certain kinds of random hardware faults. The technique provides low coverage for some coding errors and is good design practice.

D.2.4 Non-volatile memory

Global objective: The detection of information corruption in the non-volatile memory.

NOTE Depending on the type of memory implemented, a single fault can affect multiple memory locations. For example, an open row select line would prevent reading an entire row of memory. This type of failure can be easier to detect if multiple locations are tested.

D.2.4.1 Memory monitoring using error-detection-correction codes (EDC)

NOTE 1 This technique/measure is referenced in Tables D.5 and D.6.

Aim: To detect each single-bit failure, each two-bit failure, some three-bit failures, and some all-bit failures in a word (typically 32, 64 or 128 bits).

Description: Every word of memory is extended by several redundant bits to produce a modified Hamming code with a Hamming distance of at least 4. Every time a word is read, checking of the redundant bits can determine whether or not a corruption has taken place. If a difference is found, a failure message is produced.

The procedure can also be used to detect addressing failures, by calculating the redundant bits for the concatenation of the data word and its address. Otherwise for addressing failures, the probability of detection is dependent on the number of EDC bits for random data returned (for example, address line open or address line shorted to another address line such that an average of the two cells is returned). The coverage is 0 % if the addressing error leads to a completely different cell selected.

For RAM cell write-enable failure, EDC can provide high coverage if the cell cannot be initialized. The coverage is 0 % if the write-enable failure affects the entire cell after it has been initialized.

NOTE 2 This technology is often referred to as ECC (Error Correcting Code).

D.2.4.2 Modified checksum

NOTE This technique/measure is referenced in Table D.5.

Aim: To detect each single bit failure.

Description: A checksum is created by a suitable algorithm which uses each of the words in a block of memory. The checksum can be stored as an additional word in ROM, or an additional word can be added to the memory block to ensure that the checksum algorithm produces a predetermined value. In a later memory test, a checksum is created again using the same algorithm, and the result is compared with the stored or defined value. If a difference is found, a failure message is produced (see Reference [20]). The probability of a missed detection is 1/(2^size of checksum) if a random result is returned. If certain data disturbances are more probable, some checksums can provide a better detection ratio than the one for random results.

D.2.4.3 Memory signature

NOTE 1 This technique/measure is referenced in Table D.5.

Aim: To detect each one-bit failure and most multi-bit failures.

Description: The contents of a memory block are compressed (using either hardware or software) into one or more bytes using, for example, a cyclic redundancy check (CRC) algorithm. A typical CRC algorithm treats the whole contents of the block as byte-serial or bit-serial data flow, on which a continuous polynomial division is carried out using a polynomial generator. The remainder of the division represents the compressed memory contents – it is the “signature” of the memory – and is stored. The signature is computed once again in later tests and compared with one already stored. A failure message is produced if there is a difference.

CRCs are particularly effective in detecting burst errors. The effectiveness of the signature depends on the polynomial in relation to the block length of the information to be protected. The probability of a missed detection is 1/(2^size of checksum) if a random result is returned (see Reference [20]).

NOTE 2 Use of an 8 bit CRC is not generally considered the state of the art for memory sizes above 4k.

D.2.4.4 Block replication (for example double memory with hardware or software comparison)

NOTE This technique/measure is referenced in Tables D.5 and D.6.

Aim: To detect each bit failure.

Description: The address space is duplicated in two memories. The first memory is operated in the normal manner. The second memory contains the same information and is accessed in parallel to the first. The outputs are compared and a failure message is produced if a difference is detected. Dependent on memory subsystem design, storage of inverse data in one of the two memories can enhance diagnostic coverage. Coverage can be reduced if failure modes (such as common address lines, write-enables) exist that are common to both blocks or if physical placement of memory cells makes logically distant cells physical neighbours.

D.2.5 Volatile memory

Global objective: Detecting failures during addressing, writing, storing and reading.

NOTE Depending on the type of memory implemented, a single fault can affect multiple memory locations. For example an open row select line would prevent reading an entire row of memory. This type of failure can be easier to detect if multiple locations are tested.

D.2.5.1 RAM Pattern test

NOTE 1 This technique/measure is referenced in Table D.6.

Aim: To detect predominantly static bit failures.

Description: A bit pattern followed by the complement of that bit pattern is written into the cells of memory.

RAM locations are generally tested individually. The cell content is stored and then all 0s are written to the cell. The cell contents are then verified by a read back of the 0 values. The procedure is repeated by writing all 1s to the cell and reading the contents back. If a transition failure from 1 to 0 is a failure mode of concern, an additional write and read of 0s can be performed. Finally, original contents of the cell are restored (see Reference [20], Section 4.2.1). The test is effective at detecting stuck-at and transition failures but cannot detect most soft errors, addressing faults and linked cell faults.

NOTE 2 The test is often implemented in the background with interrupt suppression during the test of each individual location.

NOTE 3 Because the implementation includes a read of a just written value, optimizing compilers have a tendency to optimize out the test. If an optimizing compiler is used, good design practice is to verify the test code by an assembler-level code inspection.

NOTE 4 Some RAMs can fail such that the last memory access operation is echoed back as a read. If this is a plausible failure mode, the diagnostic can test two locations together, first writing a 0 to 1 and then a 1 to the next and then verifying a 0 is read from the first location.

D.2.5.2 Parity bit

NOTE This technique/measure is referenced in Tables D.5 and D.6.

Aim: To detect a single corrupted bit or an odd number of corrupted bits failures in a word (typically 8 bits, 16 bits, 32 bits, 64 bits or 128 bits).

Description: Every word of the memory is extended by one bit (the parity bit) which completes each word to an even or odd number of logical 1s. The parity of the data word is checked each time it is read. If the wrong number of 1s is found, a failure message is produced. The choice of even or odd parity ought to be made such that, whichever of the zero word (nothing but 0s) or the one word (nothing but 1s) is the more unfavourable in the event of a failure, then that word is not a valid code.

Parity can also be used to detect addressing failure, when the parity is calculated for the concatenation of the data word and its address. Otherwise, for addressing failures, there is a 50 % probability of detection for random data returned (for example, address line open or address line shortened to another address line such that an average of the two cells is returned). The coverage is 0 % if the addressing error leads to a completely different cell selected.

For RAM cell write-enable failure, parity can detect 50 % of failures if the cell is unable to be initialized. The coverage is 0 % if the write-enable failure affects entire cell after it has been initialized.

D.2.5.3 RAM March test

NOTE 1 This technique/measure is referenced in Table D.6.

Aim: To detect predominantly persistent bit failures, bit transition failures, addressing failures and linked cell failures.

Description: A pattern of 0s and 1s is written into the cells of memory in a specific pattern and verified in a specific order.

A March test consists of a finite sequence of March elements; while a March element is a finite sequence of operations applied to every cell in the memory array before proceeding to the next cell. For example, an operation can consist of writing a 0 into a cell, writing a 1 into a cell, reading an expected 0 from a cell, and reading an expected 1 from a cell. A failure is detected if the expected “1” is not read. The coverage level for linked cells depends on the write/read order.

Reference [20], Chapter 4, lists a number of different March tests designed to detect various RAM failure modes: stuck-at faults, transition faults (inability to transition from a one to a zero or a zero to a one but not both), address faults and linked cell faults. These types of tests are not effective for soft error detection.

NOTE 2 These tests can usually only be run at initialization or shutdown.

D.2.5.4 Running checksum/CRC

NOTE This technique/measure is referenced in Table D.6.

Aim: To detect single bit, and some multiple bit, failures in RAM.

Description: A checksum/CRC is created by a suitable algorithm which uses each of the words in a block of memory. The checksum is stored as an additional word in RAM. As the memory block is updated, the RAM checksum/CRC is also updated by removing the old data value and adding in the new data value to be stored to the memory location. Periodically, a checksum/CRC is calculated for the data block and compared to the stored checksum/CRC. If a difference is found, a failure message is produced. The probability of a missed detection is 1/size of checksum/CRC if a random result is returned. DC can be reduced as memory size increases.

D.2.6 I/O-units and interfaces

Global objective: To detect failures in input and output units (digital, analogue) and to prevent the sending of inadmissible outputs to the process.

D.2.6.1 Test pattern

NOTE This technique/measure is referenced in Tables D.7, D.11, D.12 and D.14.

Aim: To detect static failures (stuck-at failures) and cross-talk.

Description: This is a dataflow-independent cyclical test of input and output units. It uses a defined test pattern to compare observations with the corresponding expected values. Test coverage is dependent on the degree of independence between the test pattern information, the test pattern reception, and the test pattern evaluation. In a good design, the functional behaviour of the system is not unacceptably influenced by the test pattern.

D.2.6.2 Code protection

NOTE This technique/measure is referenced in Table D.7.

Aim: To detect random hardware and systematic failures in the input/output dataflow.

Description: This procedure protects the input and output information from both systematic and random hardware failures. Code protection gives dataflow-dependent failure detection of the input and output units, based on information redundancy, or time redundancy, or both. Typically, redundant information is superimposed on input data, or output data, or both. This gives a means to monitor the correct operation of the input or output circuits. Many techniques are possible, for example a carrier frequency signal can be superimposed on the output signal of a sensor and the logic unit can then check for the presence of the carrier frequency, or redundant code bits can be added to an output channel to allow the monitoring of the validity of a signal passing between the logic unit and final actuator.

D.2.6.3 Multi-channel parallel output

NOTE This technique/measure is referenced in Table D.7.

Aim: To detect random hardware failures (stuck-at failures), failures caused by external influences, timing failures, addressing failures, drift failures and transient failures.

Description: This is a dataflow-dependent multi-channel parallel output with independent outputs for the detection of random hardware failures. Failure detection is carried out via external comparators. If a failure occurs, the system can possibly be switched off directly. This measure is only effective if the dataflow changes during the diagnostic test interval.

D.2.6.4 Monitored outputs

NOTE This technique/measure is referenced in Table D.7.

Aim: To detect individual failures, failures caused by external influences, timing failures, addressing failures, drift failures (for analogue signals) and transient failures.

Description: This is a dataflow-dependent comparison of outputs with independent inputs to ensure compliance with a defined tolerance range (time, value). A detected failure cannot always be related to the defective output. This measure is only effective if the dataflow changes during the diagnostic test interval.

D.2.6.5 Input comparison/voting

NOTE This technique/measure is referenced in Tables D.7 and D.11.

Aim: To detect individual failures, failures caused by external influences, timing failures, addressing failures, drift failures (for analogue signals) and transient failures.

Description: This is a dataflow-dependent comparison of independent inputs to ensure compliance with a defined tolerance range (time, value). There will be 1 out of 2, 2 out of 3 or better redundancy. This measure is only effective if the dataflow changes during the diagnostic test interval.

D.2.7 Communication bus

Global objective: To detect failures in the information transfer.

D.2.7.1 One-bit hardware redundancy

NOTE This technique/measure is referenced in Tables D.8 and D.14.

Aim: To detect each odd-bit failure, i.e. 50 % of all the possible bit failures in the data stream.

Description: The communication bus is extended by one line (bit) and this additional line (bit) is used to detect failures by parity checking.

EXAMPLE Parity bit as implemented in a standard UART.

D.2.7.2 Multi-bit hardware redundancy

NOTE This technique/measure is referenced in Tables D.8 and D.14.

Aim: To detect failures during the communication on a bus and in serial transmission links.

Description: The communication bus is extended by two or more lines and these additional lines are used in order to detect failures by using Hamming code techniques.

D.2.7.3 Complete hardware redundancy

NOTE This technique/measure is referenced in Tables D.8 and D.14.

Aim: To detect failures during the communication by comparing the signals on two buses.

Description: The bus is duplicated and the additional lines are used to detect failures.

EXAMPLE Dual channel FlexRay implementation: the bus is duplicated and the additional lines (bits) are used in order to detect failures.

D.2.7.4 Inspection using test patterns

NOTE This technique/measure is referenced in Table D.8.

Aim: To detect static failures (stuck-at failure) and cross-talk.

Description: This is a dataflow-independent cyclical test of data paths. It uses a defined test pattern to compare observations with the corresponding expected values.

Test coverage is dependent on the degree of independence between the test pattern information, the test pattern reception, and the test pattern evaluation. In a good design, the functional behaviour of the system is not unacceptably influenced by the test pattern.

D.2.7.5 Transmission redundancy

NOTE This technique/measure is referenced in Table D.8.

Aim: To detect transient failures in bus communication.

Description: The information is transferred several times in sequence. The technique is only effective in detecting transient failures.

D.2.7.6 Information redundancy

NOTE 1 This technique/measure is referenced in Table D.8.

Aim: To detect failures in bus communication.

Description: Data is transmitted in blocks, together with a calculated checksum or CRC (cyclic redundancy check) (see References [28] and [29]) for each block. The receiver then re-calculates the checksum of the received data and compares the result with the received checksum. For CRC coverage depends on the length of the data to be covered, the size of the CRC (number of bits) and the polynomial. The CRC can be designed to address the more probable communication failure modes of the underlying hardware (for example burst errors).

The message ID can be included in the checksum/CRC calculation to provide coverage for corruptions in this part of the message (masquerading).

a) Low diagnostic coverage: Hamming distance of 2 or less

EXAMPLE 1 CRC value for message information embedded in message; with a CRC size of 5 bits and a polynomial 0x12 results in a Hamming distance of 2 for a data length of less than 2 048 bits. The transmitter includes the CRC value mentioned and the receiver confirms the data after calculating and comparing the CRC value.

b) Medium diagnostic coverage: Hamming distance of 3 or more

EXAMPLE 2 CRC value for message information embedded in message; with a CRC size of 8 bits and a polynomial 0x97 results in a Hamming distance of 4 for a data length of less than 119 bits. The transmitter includes the CRC value mentioned and the receiver confirms the data after calculating and comparing the CRC value (typically used in a LIN bus).

EXAMPLE 3 CRC value for message information and message ID embedded in the message; with a CRC size of 10 bits and a polynomial 0x319 results in a Hamming distance of 4 for a data length of less than 501 bits. The transmitter includes the CRC value mentioned and the receiver confirms the data after calculating and comparing the CRC value.

EXAMPLE 4 CRC value for message information and message ID embedded in the message; with a CRC size of 15 bits and a polynomial 0x4599 results in a Hamming distance of 5 for a data length of less than 127 bits. As well, burst errors of length up to 15 can be detected. The transmitter includes the CRC value mentioned and the receiver confirms the data after calculating and comparing the CRC value (as used in CAN).

EXAMPLE 5 CRC value for message information embedded in the message, with a CRC size of 24 bits and a polynomial 0x5D6DCB results in a Hamming distance of the CRC of 6 for a data length of less than or equal to 248 bytes and a Hamming distance of the CRC of 4 for a data length of greater than 248 bytes. The transmitter includes the CRC value mentioned and the receiver confirms the data after calculating and comparing the CRC value (as used in FlexRay for the frame CRC).

EXAMPLE 6 CRC value for message header including the message ID embedded in the message; with a CRC size of 11 bits and a polynomial 0x385 results in a Hamming distance of 6 for a data length of less than or equal to 20 bits. The transmitter includes the CRC value mentioned and the receiver confirms the data after calculating and comparing the CRC value (as used in FlexRay for the header CRC).

NOTE 2 High coverage can be reached concerning data and ID corruption, however, overall high coverage cannot be reached by checking only the coherence of the data and the ID with a signature, whatever the efficiency of the signature. Specifically, a signature does not cover the message loss or the unintended message repetition.

NOTE 3 If a checksum algorithm has a Hamming distance of less than 3, a high coverage concerning data and ID corruption can still be claimed if supported by a proper rationale.

D.2.7.7 Frame counter

NOTE This technique/measure is referenced in Table D.8.

Aim: To detect frame losses. A frame is a coherent set of data sent from one controller to other controller(s). The unique frame is identified by a message ID.

Description: Each unique safety-related frame includes a counter as part of the message which is transmitted on the bus. The counter is incremented (with roll-over) during the creation of each successive frame transmitted. The receiver is then able to detect any frame loss or non-refreshment by verifying that the counter is incrementing by one.

A special version of the frame counter would be to include separate signal counters tied to the refreshment of safety-related data. In this situation, if a frame contained more than one piece of safety-related data, an individual counter for each piece of safety-related data is provided.

D.2.7.8 Timeout monitoring

NOTE This technique/measure is referenced in Table D.8.

Aim: To detect loss of data between the sending node and the receiving node.

Description: The receiver monitors each expected safety-related message ID for time between the receipt of valid frames with this message ID. A failure would be indicated by too long a period elapsing between messages. This is intended to detect continuous loss of a communications channel or the continuous loss of one a specific message (no frames received for a specific message ID).

D.2.7.9 Read back of sent message

NOTE 1 This technique/measure is referenced in Table D.8.

Aim: To detect failures in bus communication.

Description: The transmitter reads back its sent message from the bus and compares it with the original message.

NOTE 2 This safety mechanism is used by CAN.

NOTE 3 High coverage can be reached concerning data and ID corruption, however, overall high coverage cannot be reached by checking only the coherence of the data and the ID. Other failure modes like the unintended message repetition are not necessarily covered by this safety mechanism.

D.2.8 Power supply

Global objective: To detect failures caused by a defect in the power supply.

D.2.8.1 Voltage or current control (input)

NOTE This technique/measure is referenced in Table D.9.

Aim: To detect as soon as possible wrong behaviour of input current or voltage values.

Description: Monitoring of input voltage or current.

D.2.8.2 Voltage or current control (output)

NOTE This technique/measure is referenced in Table D.9.

Aim: To detect as soon as possible wrong behaviour of output current or voltage values.

Description: Monitoring of output voltage or current.

D.2.9 Temporal and logical program sequence monitoring

NOTE This group of techniques and measures is referenced in Table D.10.

Global objective: To detect a defective program sequence. A defective program sequence exists if the individual elements of a program (for example, software modules, subprograms or commands) are processed in the wrong sequence or period of time, or if the clock of the processor is faulty.

D.2.9.1 Watchdog with separate time base without time-window

NOTE This technique/measure is referenced in Table D.10.

Aim: To monitor the behaviour and the plausibility of the program sequence.

Description: External timing elements with a separate time base (for example, watchdog timers) are periodically triggered to monitor the processor's behaviour and the plausibility of the program sequence. It is important that the triggering points are correctly placed in the program. The watchdog is not triggered at a fixed period, but a maximum interval is specified.

D.2.9.2 Watchdog with separate time base and time-window

NOTE This technique/measure is referenced in Table D.10.

Aim: To monitor the behaviour and the plausibility of the program sequence.

Description: External timing elements with a separate time base (for example watchdog timers) are periodically triggered to monitor the processor's behaviour and the plausibility of the program sequence. It is important that the triggering points are correctly placed in the program (e.g. not in an interrupt service routine).

A lower and upper limit is given for the watchdog timer. If the program sequence takes a longer or shorter time than expected, action is taken.

D.2.9.3 Logical monitoring of program sequence

NOTE This technique/measure is referenced in Table D.10.

Aim: To monitor the correct sequence of the individual program sections.

Description: The correct sequence of the individual program sections is monitored using software (counting procedure, key procedure) or using external monitoring facilities (References [23, 24]). It is important that the checking points are placed in the program so that paths which can result in a hazard if they fail to complete or execute out of sequence, due to a single or multiple-point fault, are monitored. The sequences can be updated between each function call or more tightly integrated into the program execution.

D.2.9.4 Combination of temporal and logical monitoring of program sequences

NOTE This technique/measure is referenced in Table D.10.

Aim: To monitor the behaviour and the correct sequence of the individual program sections.

Description: A temporal facility (for example a watchdog timer) monitoring the program sequence is retriggered only if the sequence of the program sections is also executed correctly. This is a combination of the technique in D.2.9.3 and either D.2.9.1 or D.2.9.2.

D.2.9.5 Combination of temporal and logical monitoring of program sequences with time dependency

NOTE This technique/measure is referenced in Table D.10.

Aim: To monitor the behaviour, correct sequencing and the execution time interval of the individual program sections.

Description: A Program Flow Monitoring strategy is implemented where software update points are expected to occur within a relative time window. The PFM sequence result and time calculation are monitored by external monitoring facilities.

D.2.10 Sensors

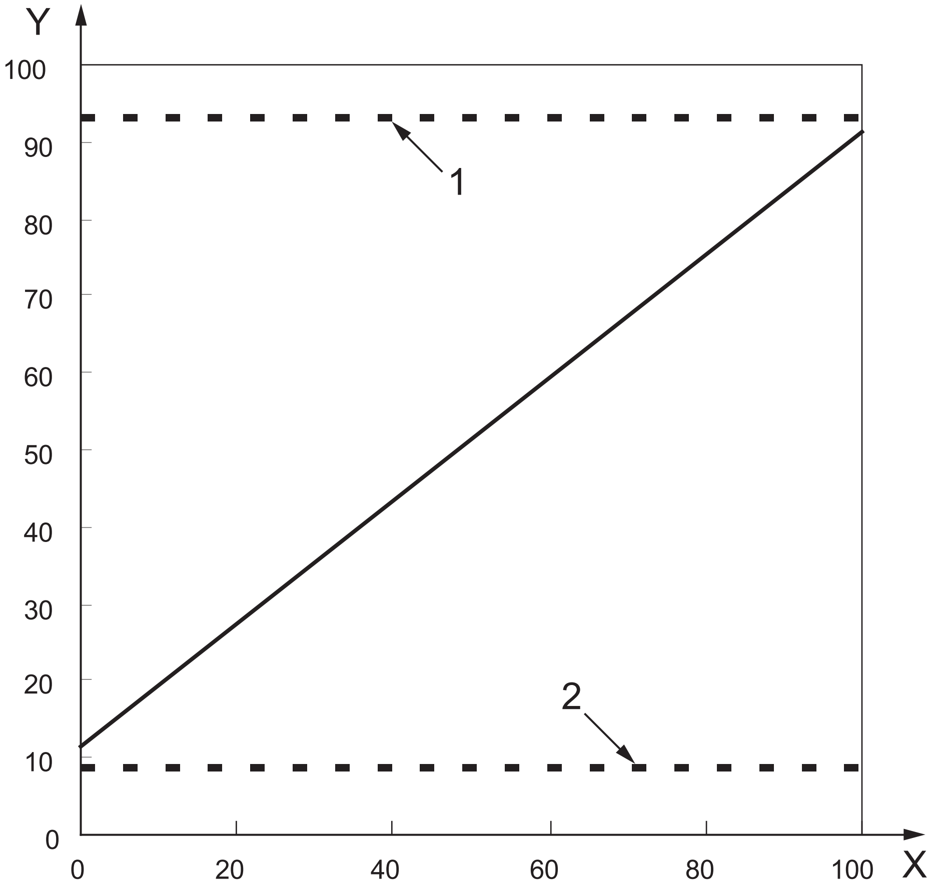
Global objective: To control failures in the sensors of the system.

D.2.10.1 Sensor valid range

NOTE This technique/measure is referenced in Table D.11.

Aim: To detect sensor shorts to ground or power and some open circuits.

Description: Limit valid reading to the middle part of the sensor electrical range (see Figure D.4 for example). If a sensor reading is in an invalid region, this indicates an electrical problem with the sensor such as a short to power or to ground. Typically used with sensors read by the ECU using ADCs.



Key

X physical sensor reading, in %

Y measured sensor reading, in % of reference voltage

1 out-of-range high

2 out-of-range low

Figure D.4 — Sensor with out-of-range region

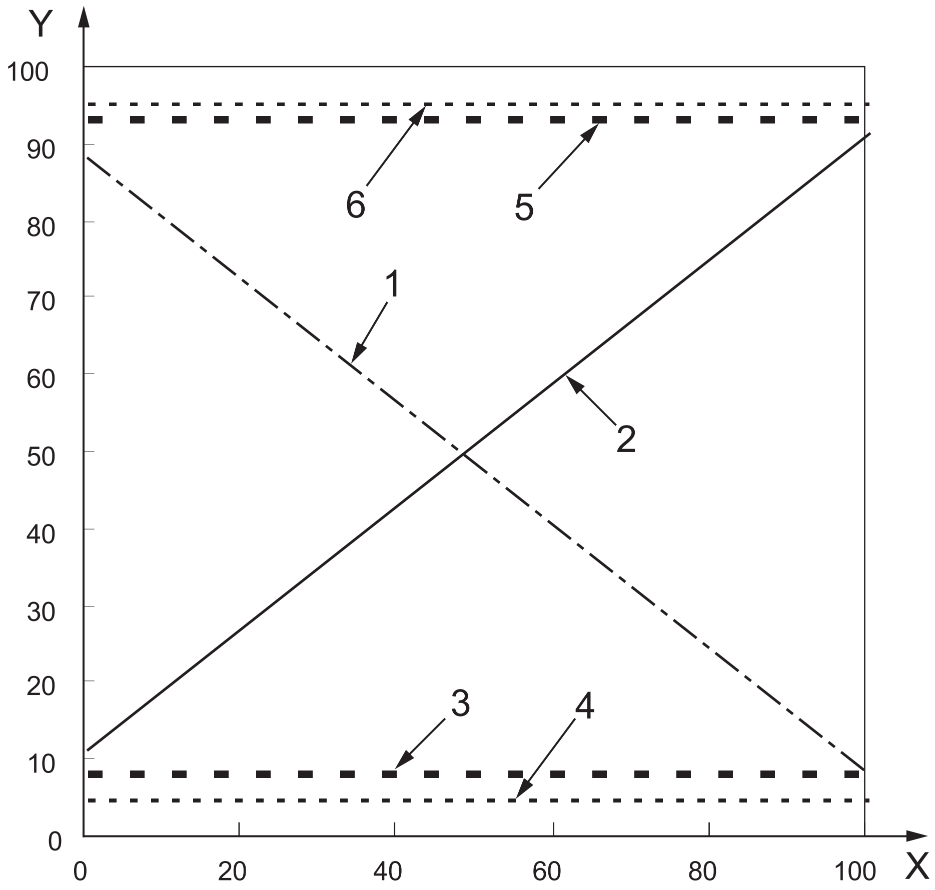
D.2.10.2 Sensor correlation

NOTE This technique/measure is referenced in Table D.11.

Aim: To detect sensor-in-range drifts, offsets or other errors using a redundant sensor.

Description: Comparison of two identical or similar sensors to detect in-range failures such as drifts, offsets or stuck-at failures. See Figure D.5 for an example with two equal but opposite slope sensors. Note, that the out-of-range region is different for each sensor. Typically used with sensors read by the ECU using ADCs.

For the example of Figure D.5, sensors would be converted to equal slope and compared to agree within a threshold. The threshold is selected taking into account the ADC tolerance and the variation in the electrical elements. Both sensors are sampled by the ECU at as close to the same time as possible to avoid false failures due to the sensor readings dynamically changing.



Key

X physical sensor reading, in %

Y measured sensor reading, in % of reference voltage

1 sensor 1

2 sensor 2

3 out-of-range sensor 1 low

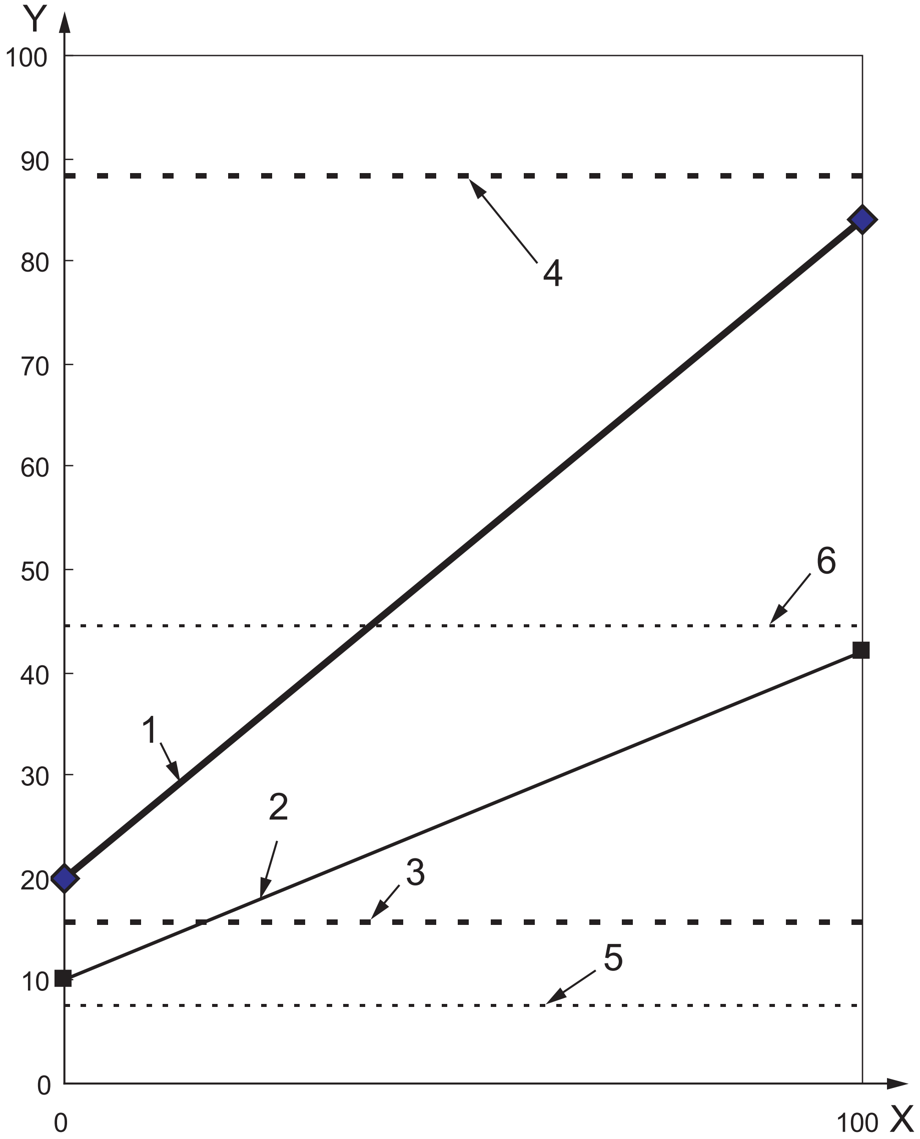
4 out-of-range sensor 2 low

5 out-of-range sensor 1 high

6 out-of-range sensor 2 high

Figure D.5 — Equal and opposite slope sensors with out-of-range regions

Equal slope sensor based diagnostics do not detect situations where the two sensors are shorted together yielding correlated readings at the crossing point or common cause failures where a single component, e.g. the ADC, corrupts both sensor results in a similar way. An alternative design based on one full and one half slope sensor is given in Figure D.6.



Key

X physical sensor reading, in %

Y measured sensor reading, in % of reference voltage

1 sensor 1

2 sensor 2

3 out-of-range sensor 1 low

4 out-of-range sensor 1 high

5 out-of-range sensor 2 low

6 out-of-range sensor 2 high

Figure D.6 — One full and one half slope sensor with out-of-range regions

D.2.10.3 Sensor rationality check

NOTE This technique/measure is referenced in Table D.11.

Aim: To detect sensor-in-range drifts, offsets or other errors using multiple diverse sensors.

Description: Comparison of two (or more) sensors measuring different properties to detect in-range failures such as drifts, offsets or stuck-at failures. The sensor measurements are converted to equivalent values using a model to provide values that can be compared.

EXAMPLE The comparison of gasoline engine throttle position, manifold pressure and mass air flow sensors after each is converted to an air flow reading. The usage of diverse sensors reduces the problem of systematic faults.

D.2.11 Actuators

Global objective: To control failures in the final elements of the system.

D.2.11.1 Monitoring

NOTE 1 This technique/measure is referenced in Table D.12.

Aim: To detect the incorrect operation of an actuator.

Description: The operation of the actuator is monitored.

NOTE 2 Monitoring can be done at the actuator level by physical parameter measurements (which can have high coverage) but also at the system level regarding the actuator failure effect.

EXAMPLE 1 For a cooling radiator fan, monitoring at system level uses a temperature sensor to detect failure of the cooling radiator fan. Monitoring of physical parameters measures the voltage, or the current, or both, on the inputs of the cooling radiator fan.

EXAMPLE 2 Feedback control is used to move a throttle blade to a desired position. The actual position is measured and compared to the expected throttle position determined from the commanded throttle position and a model of the desired performance. If the two values differ from each other, after taking into account hysteresis, an error can be declared.

Annex E
(informative)

Example calculation of hardware architectural metrics: "single-point fault metric" and "latent-fault metric"

This annex gives an example of calculating the single-point fault metric and the latent-fault metric for each safety goal of the item as required in alternative a) of 8.4.7 and 8.4.8.

The system for this example realises two functions implemented on a single ECU.

Function 1 has one input (temperature measured via sensor R3) and one output (valve 2 controlled by I71) and its behaviour is to open valve 2 when the temperature is higher than 90 °C.

If no current flows through I71, valve 2 is open.

The associated safety goal 1 is "valve 2 shall not be closed for longer than x ms when the temperature is higher than 100 °C". The safety goal is assigned ASIL B. The safe state is: valve2 open.

The value of sensor R3 is read by the microcontroller ADC. R3 resistance decreases as the temperature rises. There is no monitoring on this input. The output stage controlling T71 is monitored by the analogue input InADC1 (Safety mechanism SM1 in the tables). In this example, we will assume that the safety mechanism SM1 is able to detect certain failure modes of T71 with a 90 % diagnostic coverage with respect to the violation of the safety goal. If a failure is detected by SM1, the safe state is activated but no lamp is switched on. Therefore, the diagnostic coverage with respect to latent faults is claimed to be only 80 % (the driver will notice the failure through the functionality degradation).

Function 2 has two inputs (wheel speed measured via sensors I1 and I2 generating pulses) and one output (valve 1 controlled by I61) and its behaviour is to open valve 1 when the vehicle speed is higher than 90 km/h.

If no current flows through I61, valve 1 is open.

The associated safety goal 2 is "valve 1 shall not be closed for longer than y ms when the speed is higher than 100 km/h". The safety goal is assigned ASIL C. The safe state is: valve 1 open.

The values of sensors I1 and I2 pulses are read by the microcontroller. The wheel speed is computed using the mean value given by the sensors. The safety mechanism 2 (Safety Mechanism SM2 in the tables) compares both inputs. It detects the failures of each input with a 99 % diagnostic coverage. In the case of an inconsistency, Out.1 is set to 0. This opens valve1 (A "0" voltage on transistor opens the gate. A "0" voltage on I61 opens valve 1). Therefore, 99 % of the faults that have the potential to violate the safety goal are detected and lead to the safe state. When the safe state is activated, the lamp L1 is on. Therefore, these faults are 100 % perceived. The remaining 1 % of faults are residual faults and not latent faults.

The output stage controlling T61 is monitored by the analogue input InADC2 (Safety mechanism SM3 in the tables). The wheel speed is computed using the mean value given by the sensors.

The microcontroller has no internal redundancy. If no detailed information about the ratio of safe faults of a complex part is available, a conservative ratio of 50 % safe faults can be assumed. A global coverage of 90 % with respect to the violation of the safety goal, through internal self tests and the external watchdog (Safety Mechanism SM4 in the tables) is also assumed. The watchdog gets a live signal via the output 0 of the microcontroller. When the watchdog is no longer refreshed, its output goes low. A fault detection by SM4 (watchdog and microcontroller self tests) switches both functions to their safe state and switches L1 on. Therefore the diagnostic coverage with respect to the latent faults is claimed to be 100 %.

L1 is an LED on the dashboard, it is lit upon detection of a multiple-point failure, of which only a proportion can be detected, and indicates to the driver that the safe state of function 1 (valve 1 open) has been activated.

NOTE 1 The harness failures are not considered in this example.

NOTE 2 The fault model used for a given electronic part can differ depending on the application.

EXAMPLE 1 The fault model of a resistor depends if the hardware part is used in a digital input (such as R11, R12, R13...) or an analogue input (such as R3). In the first case the fault model can be “open/closed” whereas in the second case it can be “open/closed/drift”.

NOTE 3 The first metric only uses the failure mode coverage of the safety mechanisms that aim at preventing the violation of the safety goal. The second metric only uses the failure mode coverage of the safety mechanisms that aim at preventing the failure mode from being latent.

EXAMPLE 2 The failure mode “open” of R21 has the potential to violate safety goal 2 in the absence of a safety mechanism. Safety mechanism 3 detects this failure mode with a failure mode coverage of 99 % and switches the system into a safe state. When detecting this failure mode, an alert is displayed; the failure mode coverage with respect to latent failures is 100 %.

NOTE 4 In this example, assumptions on the failure mode distribution of the hardware elements have been considered. If no particular failure mode distribution can be argued or referenced, an equal distribution of the failure modes can be assumed.

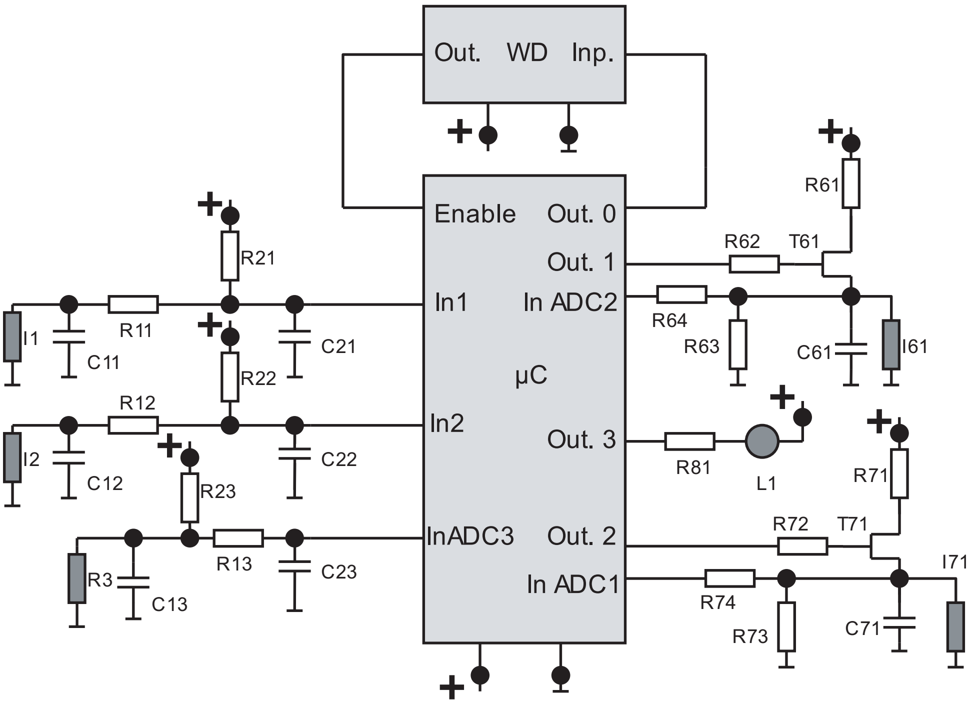


Figure E.1 — Example diagram

Please note that in the following tables, the coverage by a safety mechanism of a specific failure mode of a hardware element is called “failure mode coverage”.

Component Name	Failure rate/FIT	Safety-related component to be considered in the calculations?	Failure Mode	Failure rate distribution	Failure mode that has the potential to violate the safety goal in absence of safety mechanisms?	Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal?	Failure mode coverage wrt. violation of safety goal	Residual or Single-Point Fault failure rate/FIT	Failure mode that may lead to the violation of safety goal in combination with an independent failure of another component?	Detection means? Safety mechanism(s) allowing to prevent the failure mode from being latent?	Failure mode coverage with respect to latent failures	Latent Multiple-Point Fault failure rate/FIT
R3 note 1	3	YES	open	30 %	X	none	0 %	0,9				
			closed	10 %								
			drift 0,5	30 %								
			drift 2	30 %	X		0 %	0,9				
R13 note 1, note 2 and note 7	2	YES	open	90 %	X	none	0 %	1,8				
			closed	10 %	X		0 %	0,2				
R23 note 1	2	YES	open	90 %		none						
			closed	10 %	X		0 %	0,2				
C13 note 3 and note 7	2	YES	open	20 %	X	none	0 %	0,4				
			closed	80 %								
C23	2	NO	open	20 %								
			closed	80 %								
WD	20	YES	Out. Stuck at 1	50 %					X	none	0 %	10
			Out. Stuck at 0	50 %								
T71	5	YES	open circuit	50 %		SM1				SM1		
			short circuit	50 %	X		90 %	0,25	X		80 %	0,45
R71 note 2 and note 7	2	YES	open	90 %						none		
			closed	10 %					X		0 %	0,2
R72 note 2 and note 7	2	YES	open	90 %						none		
			closed	10 %					X		0 %	0,2
R73	2	NO	open	90 %								
			closed	10 %								
R74 note 2 and note 7	2	YES	open	90 %					X	none	0 %	1,8
			closed	10 %					X		0 %	0,2
171	5	NO	open	70 %								
			closed	20 %								
C71 note 3	2	YES	open	20 %					X	none		0,4
			closed	80 %								
R81	2	NO	open	90 %								
			closed	10 %								
L1	10	NO	open	90 %								
			closed	10 %								
μC	100	YES	All	50 %	X	SM4	90 %	5	X	SM4	100 %	0
			All	50 %								
							Σ	9,65			Σ	13,25

Total failure rate	163	Single-Point Fault Metric = 1-(9,65/142) = 93,2 %	Latent Fault Metric = 1-(13,25/(142-9,65)) = 90,0 %
Total Safety Related	142		
Total Not Safety Related	21		

Figure E.2 — Safety goal 1 (*continued*)

Safety goal 1 is assigned ASIL B, which has, if Table 4 is used, a single-point fault metric recommendation of ≥90 %, and, if Table 5 is used, a latent-fault metric recommendation of ≥60 %. The single-point fault metric recommendation is satisfied by the calculated metric of 93,2 % and the latent-fault metric recommendation is satisfied by the value of 90 %.

NOTE 1 The failure modes “open” on R3 and R13 and “closed” on R23 are single-point faults. They lead directly to the violation of the safety goal and no safety mechanism covers faults of these hardware parts.

NOTE 2 The purpose of this hardware part is electrical protection. The closed failure mode means loss of protection.

NOTE 3 The purpose of this hardware part is ESD protection. The open failure mode means loss of protection.

NOTE 4 The purpose of this hardware part is electrical protection. One failure mode is the loss of electrical protection. The other mode has the potential to violate the safety goal in the absence of safety mechanisms.

NOTE 5 The elements with failures that do not have the potential to significantly contribute to the violation of the safety goal are not considered in the calculations. Here, L1 and R81 are elements which implement a safety mechanism to prevent dual-point faults from being latent. The multiple-point faults with n>2 are considered to be safe faults.

NOTE 6 The faults that lead directly to the violation of the safety goal (single-point faults or residual faults) cannot contribute anymore to the latent faults population. Therefore, for instance, the failure rate of the latent failure mode “closed gate” of T71 is computed as follows:

$$\lambda_{\mathrm{MPF,L}} = \left[\left(\lambda_{\mathrm{T71}} \times \mathrm{FailureModeDistrib}_{\mathrm{closed\,gate}} \right) - \lambda_{\mathrm{T71}_{\mathrm{RF}}} \right] \times \left(1 - \mathrm{FMC}_{\mathrm{LatentFaults}} \right) = \left[\left(5 \times 0{,}1 \right) - 0{,}05 \right] \times \left(1 - 0{,}8 \right) = 0{,}09$$

NOTE 7 The classification of the failure modes leading to the loss of ESD or electrical protection is based on a case-by-case analysis and takes into consideration the likelihood of the ESD or electrical stress and the characterized effects of the ESD or electrical stress with respect to the safety goal. If for example the ESD event is likely to occur during the vehicle lifetime and its effects can lead to the violation of the safety goal in the absence of the given protection, then the failure mode leading the loss of the protection is classified as a single-point fault. This annex is an example on how to handle those cases within the metrics. In practice ESD or EMI stresses do not have this impact on typical designs similar to that of the example.

Figure E.2 — Safety goal 1

Component Name	Failure rate/FIT	Safety-related component to be considered in the calculation?	Failure Mode	Failure rate distribution	Failure mode that has the potential to violate the safety goal in absence of safety mechanisms?	Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal?	Failure mode coverage wrt. violation of safety goal	Residual or Single-Point Fault failure rate/FIT	Failure mode that may lead to the violation of safety goal in combination with an independent failure of another component?	Detection means? Safety mechanism(s) allowing to prevent the failure mode from being latent?	Failure mode coverage wrt. Latent failures	Latent Multiple-Point Fault failure rate/FIT
R11 note 1, note 6 and note 7	2	YES	open	90 %	X	SM2	99 %	0,018	X	SM2	100 %	0
			closed	10 %	X		99 %	0,002	X		100 %	0
R12 note 1, note 6 and note 7	2	YES	open	90 %	X	SM2	99 %	0,018	X	SM2	100 %	0
			closed	10 %	X		99 %	0,002	X		100 %	0
R21 note 2	2	YES	open	90 %	X	SM2	99 %	0,018	X	SM2	100 %	0
			closed	10 %	X		99 %	0,002	X		100 %	0
R22 note 2	2	YES	open	90 %	X	SM2	99 %	0,018	X	SM2	100 %	0
			closed	10 %	X		99 %	0,002	X		100 %	0
C11 note 1, note 6 and note 7	2	YES	open	20 %	X	SM2	99 %	0,004	X	SM2	100 %	0
			closed	80 %	X		99 %	0,016	X		100 %	0
C12 note 1, note 6 and note 7	2	YES	open	20 %	X	SM2	99 %	0,004	X	SM2	100 %	0
			closed	80 %	X		99 %	0,016	X		100 %	0
C21	2	YES	open	20 %		SM2				SM2		
			closed	80 %	X		99 %	0,016	X		100 %	0
C22	2	YES	open	20 %		SM2				SM2		
			closed	80 %	X		99 %	0,016	X		100 %	0
I1	4	YES	open	70 %	X	SM2	99 %	0,028	X	SM2	100 %	0
			closed	20 %	X		99 %	0,008	X		100 %	0
			drift 0,5	5 %	X		99 %	0,002	X		100 %	0
			drift 2	5 %								
I2	4	YES	open	70 %	X	SM2	99 %	0,028	X	SM2	100 %	0
			closed	20 %	X		99 %	0,008	X		100 %	0
			drift 0,5	5 %	X		99 %	0,002	X		100 %	0
			drift 2	5 %								
WD	20	YES	Out. Stuck at 1	50 %					X	none	0 %	10
			Out. Stuck at 0	50 %								
T61	5	YES	open circuit	50 %		SM3				SM3		
			short circuit	50 %	X		90 %	0,25	X		100 %	0
R61 note 3 and note 6	2	YES	open	90 %						none		
			closed	10 %					X		0 %	0,2
R62 note 3 and note 6	2	YES	open	90 %						none		
			closed	10 %					X		0 %	0,2
R63	2	NO	open	90 %								
			closed	10 %								
R64 note 1 and note 6	2	YES	open	90 %					X	none	0 %	1,8
			closed	10 %					X		0 %	0,2
I61	5	NO	open	70 %								
			closed	20 %								
C61 note 4 and note 6	2	YES	open						X	none	0 %	0,4
			closed	80 %								
R81	2	NO	open	90 %								
			closed	10 %								
L1	10	NO	open	90 %								
			closed	10 %								
µC	100	YES	All	50 %	X	SM4	90 %	5	X	SM4	100 %	0
			All	50 %								
							∑	5,48			∑	12,80

Total failure rate	176	Single-Point Fault Metric = 1-(5,48/157) = 96,5 %	Latent Fault Metric = 1-(13,99/(157-5,48)) = 91,6 %
Total Safety Related	157		
Total Not Safety Related	19		

Figure E.3 — Safety goal 2 (*continued*)

Safety goal 2 is assigned ASIL C, which has, if Table 4 is used, a single-point fault metric requirement of ≥97 %, and, if Table 5 is used, a latent-fault metric recommendation of ≥80 %. The single-point fault metric requirement is not satisfied by the calculated metric of 96,5 % and the latent-fault metric recommendation is satisfied by the value of 91,6 %.

NOTE 1 The purpose of this hardware part is electrical protection. One failure mode is the loss of electrical protection. The other mode has the potential to violate the safety goal in absence of safety mechanisms.

NOTE 2 Both failure modes have the potential to violate the safety goal in absence of safety mechanisms as in both cases, no speed pulses are transmitted. This leads to a wrong speed acquisition. The sensor is an open-collector sensor.

NOTE 3 The purpose of this hardware part is electrical protection. The close failure mode means loss of protection.

NOTE 4 The purpose of this hardware part is ESD protection. The open failure mode means loss of protection.

NOTE 5 The elements with failures that do not have the potential to significantly contribute to the violation of the safety goal are not considered in the calculations. Here, L1 and R81 are elements which implement a safety mechanism to prevent dual-point faults from being latent. The multiple-point faults with $n>2$ are considered to be safe faults.

NOTE 6 The classification of the failure modes leading to the loss of ESD or electrical protection is based on a case-by-case analysis and takes into consideration the likelihood of the ESD or electrical stress and the characterized effects of the ESD or electrical stress with respect to the safety goal. If for example the ESD event is likely to occur during the vehicle lifetime and its effects can lead to the violation of the safety goal in the absence of the given protection, then the failure mode leading the loss of the protection is classified as a single-point fault. This annex is an example on how to handle those cases within the metrics. In practice ESD or EMI stresses do not have this impact on typical designs similar to that of the example. Moreover it is considered here that SM4 does not cover these failure modes even if they can lead to some damage to the microcontroller.

NOTE 7 The loss of electrical protection will cause a wrong input value and will be detected by SM2 and therefore will not be latent.

Figure E.3 — Safety goal 2

Annex F
(informative)

Application of scaling factors

The scaling factor is a factor that is used to combine failure rates from multiple sources in the calculations of the Probabilistic Metric for random Hardware Failures (PMHF).

A target value, defined in 9.4.2.1, for the PMHF for each safety goal is derived from one of the three sources of reference:

- Table 6, or
- field data of similar well-trusted design principles, or
- quantitative analysis techniques applied on similar well-trusted design principles using failure rates as given in 8.4.3.

To verify that the hardware design meets the defined target values, failure rates are calculated based on the hardware parts involved. The failure rate of hardware parts can be estimated based on one of the three sources described in 8.4.3:

a) hardware part failure rates data from a recognized industry source, or

b) statistics based on field returns or tests (with an adequate confidence level), or

c) expert judgment founded on an engineering approach based on quantitative and qualitative arguments.

Therefore, in the calculations, different failure rate sources can be used for different hardware parts of the item.

Let T_a, T_b and T_c be the three possible sources for the definition of the target values for the PMHF and F_a, F_b and F_c be the three possible sources for the estimation of a hardware part failure rate. Let $\pi_{Fi \rightarrow Fj}$ be the scaling factor between sources F_i and F_j. This factor can be used to scale a hardware part failure rate based on F_i to a failure rate based on F_j, as defined in Equation (F.1):

$$\pi_{Fi \rightarrow Fj} = \frac{\lambda_{k.Fj}}{\lambda_{k.Fi}} \qquad \text{(F.1)}$$

where

$\lambda_{k.Fj}$ is the failure rate for a hardware part using F_j as the source for the failure rate;

$\lambda_{k.Fi}$ is the failure rate for the same hardware part using F_i as the source for the failure rate.

In this case, knowing the corresponding scaling factor enables the scaling of a similar hardware part failure rate based on F_i to a failure rate based on F_j, as defined in Equation (F.2):

$$\lambda_{l,Fj} = \pi_{Fi \rightarrow Fj} \times \lambda_{l,Fi} \qquad \text{(F.2)}$$

Table F.1 shows the possible combinations of target values and failure rates.

NOTE 1 The targets of Table 6 are based on calculations using handbook data and under the assumption that handbook data are very pessimistic.

NOTE 2 If the source of data for the target and for the hardware part failure rate are similar, then no scaling is necessary.

Table F.1 — Possible combinations of sources of target values and failure rates to produce consistent failure rates for use in calculations

Data source for failure rates of hardware parts	Data source for target value		
	Table 6 9.4.2.1 a)	Field data 9.4.2.1 b)	Quantitative analysis 9.4.2.1 c)
Standard database 8.4.3 a)	$\lambda_{k,Fa}$ [a]	$\lambda_{k,Fb} = \pi_{Fa\to Fb} \times \lambda_{k,Fa}$	[b]
Statistics 8.4.3 b)	$\lambda_{k,Fa} = \pi_{Fb\to Fa} \times \lambda_{k,Fb}$	$\lambda_{k,Fb}$	[b]
Expert judgment 8.4.3 c)	$\lambda_{k,Fa} = \pi_{Fc\to Fa} \times \lambda_{k,Fc}$	$\lambda_{k,Fb} = \pi_{Fc\to Fb} \times \lambda_{k,Fc}$	[b]

[a] For some types of hardware parts, different handbooks can give different estimates of the failure rate of the same type of hardware part. Therefore the scaling factor can be used to scale the failure rates of a hardware part using different handbooks.

[b] To have a consistent approach, failure rates have the same origin as the failure rates used in the calculation of the target value.

The scaling is possible if sufficient evidence is available that a factor exists between two possible sources of target values.

For example, if sufficient data exist about a “predecessor” system whose failure rate can be considered representative of that expected for the item under consideration.

EXAMPLE 1 Evidence can be made available that $\frac{10^{-8}}{h}$ with a 99 % level of confidence is similar to $\frac{10^{-9}}{h}$ with a 70 % level of confidence. Therefore, failure rates based on a recognized industry source considered with a 99 % level of confidence can be scaled to failure rates based on statistics with a 70 % level of confidence using the scaling factor $\pi_{Fa\to Fb} = \left(\frac{10^{-9}}{h}\right) \div \left(\frac{10^{-8}}{h}\right) = \frac{1}{10}$ or the other way round.

NOTE 3 Based on experience, a 99 % level of confidence can be considered for failure rates based on recognized industry sources as referred to in 8.4.3.

EXAMPLE 2 From a previous design, calculated failure rates from a data handbook and warranty data have been obtained. We know that

$$\frac{\lambda_{\text{handbook}}}{\lambda_{\text{warranty}}} = \pi_{Fb\to Fa} = 10 \qquad \text{(F.3)}$$

where

$\lambda_{\text{handbook}}$ is the calculated failure rates from a data handbook,

$\lambda_{\text{warranty}}$ is the calculated failure rates from warranty data;

$\pi_{Fb\to Fa}$ is the resulting scaling factor.

If in a new design, we use the handbook data to determine the failure rates except for one hardware part (hardware part 1) for which we have only warranty data, then we can determine the handbook scaled data for this hardware part:

$$\lambda_{1,\text{handbook}} = \pi_{Fb \rightarrow Fa} \times \lambda_{1,\text{warranty}} \tag{F.4}$$

where

$\lambda_{1,\text{handbook}}$ is the failure rate of the hardware part 1 using handbook data;

$\lambda_{1,\text{warranty}}$ is the failure rate of the hardware part 1 using warranty data.

For instance, if $\lambda_{1,\text{warranty}} = \frac{9 \times 10^{-9}}{h}$, then $\lambda_{1,\text{handbook}}$ can be calculated as $\left(9 \times 10^{-9}\right) \times 10 = \frac{9 \times 10^{-8}}{h}$.

Using this $\lambda_{1,\text{handbook}}$, a consistent evaluation of the violation of the safety goal due to random hardware failures can be done.

Bibliography

[1] ISO 7637-2, *Road vehicles — Electrical disturbances from conduction and coupling — Part 2: Electrical transient conduction along supply lines only*

[2] ISO 7637-3, *Road vehicles — Electrical disturbances from conduction and coupling — Part 3: Electrical transient transmission by capacitive and inductive coupling via lines other than supply lines*

[3] ISO 10605, *Road vehicles — Test methods for electrical disturbances from electrostatic discharge*

[4] ISO 11452-2, *Road vehicles — Component test methods for electrical disturbances from narrowband radiated electromagnetic energy — Part 2: Absorber-lined shielded enclosure*

[5] ISO 11452-4, *Road vehicles — Component test methods for electrical disturbances from narrowband radiated electromagnetic energy — Part 4: Harness excitation methods*

[6] ISO 16750-2, *Road vehicles — Environmental conditions and testing for electrical and electronic equipment — Part 2: Electrical loads*

[7] ISO 16750-4, *Road vehicles — Environmental conditions and testing for electrical and electronic equipment — Part 4: Climatic loads*

[8] ISO 16750-5, *Road vehicles — Environmental conditions and testing for electrical and electronic equipment — Part 5: Chemical loads*

[9] IEC 61508 (all parts), *Functional safety of electrical/electronic/programmable electronic safety-related systems*

[10] IEC 61709, *Electronic components — Reliability — Reference conditions for failure rates and stress models for conversion*

[11] IEC 62061:2005, *Safety of machinery — Functional safety of safety-related electrical, electronic and programmable electronic control systems*

[12] IEC/TR 62380, *Reliability data handbook — Universal model for reliability prediction of electronics components, PCBs and equipment*

[13] EN 50129:2003, *Railway applications — Communication, signalling and processing systems — Safety related electronic systems for signalling*

[14] MIL HDBK 217 F notice 2, *Military handbook: Reliability prediction of electronic equipment*

[15] MIL HDBK 338, *Military handbook: Electronic reliability design handbook*

[16] NPRD 95, Non-electronic Parts Reliability Data

[17] RIAC FMD 97, Failure Mode / Mechanism Distributions

[18] RIAC HDBK 217 Plus, Reliability Prediction Models

[19] UTE C80-811, Reliability methodology for electronic systems

[20] VAN DE GOOR, A.J.: Testing Semiconductor Devices, Theory and Practice, A.J. van de Goor/ComTex Publishing, 1999

[21] SUNDARAM, P. and D'AMBROSIO, J.G., *Controller Integrity in Automotive Failsafe System Architectures*, SAE 2006 World Congress, 2006-01-0840

[22] FRUELING, T., *Delphi Secured Microcontroller Architecture*, SAE 2000 World Congress, SAE# 2000-01-1052

[23] MAHMOOD, A. and MCCLUSKEY, E.J., “Concurrent Error Detection Using Watchdog Processors – A Survey”, *IEEE Trans. Computers*, 37(2), 160-174 (1988)

[24] LEAPHART, E., CZERNY, B., D'AMBROSIO, J., et al, *Survey of Software Failsafe Techniques for Safety-Critical Automotive Applications*, SAE 2005 World Congress, 2005-01-0779

[25] MARIANI, R., FUHRMANN, P., VITTORELLI, B., *Cost-effective Approach to Error Detection for an Embedded Automotive Platform*, 2006-01-0837, SAE 2006 World Congress & Exhibition, April 2006, Detroit, MI, USA”

[26] PATEL, J., FUNG, L. “Concurrent Error Detection in ALU's by Recomputing with Shifted Operands”, *IEEE Transactions on Computers*, Vol. C-31, pp.417-422, July 1982

[27] FORIN, P., *Vital Coded Microprocessor: Principles and Application for various Transit Systems*, Proc. IFAC-GCCT, Paris, France, 1989

[28] RAMABADRAN, T.V.; Gaitonde, S.S. (1988). “A tutorial on CRC computations”. *IEEE Micro* 8 (4): 62–75, 1988

[29] Koopman, Philip; Chakravarty, Tridib (2004). *Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks* The International Conference on Dependable Systems and Networks, DSN-2004, http://www.ece.cmu.edu/~koopman/roses/dsn04/koopman04_crc_poly_embedded.pdf

ICS 43.040.10

Price based on 76 pages