COMMISSION ELECTROTECHNIQUE INTERNATIONALE

CEI IEC 61508-6

INTERNATIONAL ELECTROTECHNICAL COMMISSION

Functional safety of electrical/electronic/ programmable electronic safety-related systems

Part 6: Guidelines on the application of parts 2 and 3

Contents

Figures

Tables

FUNCTIONAL SAFETY OF ELECTRICAL/ELECTRONIC/PROGRAMMABLE ELECTRONIC SAFETY-RELATED SYSTEMS

Part 6: Guidelines on the application of parts 2 and 3

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC national committees). The object of the IEC is to promote international cooperation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes international standards. Their preparation is entrusted to technical committees; any IEC national committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.

2) The formal decisions or agreements of the IEC on technical matters, prepared by technical committees on which all the national committees having a special interest therein are represented, express, as nearly as possible, an international consensus of opinion on the subjects dealt with.

3) They have the form of recommendations for international use published in the form of standards, technical reports or guides and they are accepted by the national committees in that sense.

4) In order to promote international unification, IEC national committees undertake to apply IEC international standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC standard and the corresponding national or regional standard shall be clearly indicated in the latter.

5) Attention is drawn to the possibility that some of the elements of IEC 61508 may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

6) The IEC has not laid down any procedure concerning marking as an indication of approval and has no responsibility when an item of equipment is declared to comply with one of its standards.

IEC 61508-6 has been prepared by sub-committee 65A: System aspects, of IEC technical committee 65: Industrial process measurement and control.

The text of this part is based on the following documents:

FDIS	Report on voting
65A/xxx	65A/xxx

Full information on the voting for the approval of this standard can be found in the voting report indicated in the above table.

Annexes A to E are for information only.

IEC 61508 consists of the following parts, under the general title “Functional safety of electrical/ electronic/programmable electronic safety-related systems”:

— Part 1: General requirements;

— Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems;

— Part 3: Software requirements;

— Part 4: Definitions and abbreviations;

— Part 5: Examples of methods for the determination of safety integrity levels;

— Part 6: Guidelines on the application of parts 2 and 3;

— Part 7: Overview of techniques and measures.

Introduction

Systems comprised of electrical and/or electronic components have been used for many years to perform safety functions in most application sectors. Computer-based systems (generically referred to as programmable electronic systems (PESs)) are being used in all application sectors to perform non-safety functions and, increasingly, to perform safety functions. If computer system technology is to be effectively and safely exploited, it is essential that those responsible for making decisions have sufficient guidance on the safety aspects on which to make those decisions.

This standard sets out a generic approach for all safety lifecycle activities for systems comprised of electrical and/or electronic and/or programmable electronic components (electrical/electronic/ programmable electronic systems (E/E/PESs)) that are used to perform safety functions. This unified approach has been adopted in order that a rational and consistent technical policy be developed for all electrically-based safety-related systems. A major objective is to facilitate the development of application sector standards.

In most situations, safety is achieved by a number of protective systems which rely on many technologies (for example mechanical, hydraulic, pneumatic, electrical, electronic, programmable electronic). Any safety strategy must therefore consider not only all the elements within an individual system (for example sensors, controlling devices and actuators) but also all the safety-related systems making up the total combination of safety-related systems. Therefore, while this standard is concerned with electrical/electronic/programmable electronic (E/E/PE) safety-related systems, it may also provide a framework within which safety-related systems based on other technologies may be considered.

It is recognised that there is a great variety of E/E/PES applications in a variety of application sectors and covering a wide range of complexity, hazard and risk potentials. In any particular application, the exact prescription of safety measures will be dependent on many factors specific to the application. This standard, by being generic, will enable such a prescription to be formulated in future application sector international standards.

This standard:

— considers all relevant overall, E/E/PES and software safety lifecycle phases (for example, from initial concept, through design, implementation, operation and maintenance to decommissioning) when E/E/PESs are used to perform safety functions;

— has been conceived with a rapidly developing technology in mind – the framework is sufficiently robust and comprehensive to cater for future developments;

— enables application sector international standards, dealing with safety-related E/E/PESs, to be developed – the development of application sector international standards, within the framework of this standard, should lead to a high level of consistency (for example, of underlying principles, terminology etc) both within application sectors and across application sectors; this will have both safety and economic benefits;

— provides a method for the development of the safety requirements specification necessary to achieve the required functional safety for E/E/PE safety-related systems;

— uses safety integrity levels for specifying the target level of safety integrity for the safety functions to be implemented by the E/E/PE safety-related systems;

— adopts a risk-based approach for the determination of the safety integrity level requirements;

— sets numerical target failure measures for E/E/PE safety-related systems which are linked to the safety integrity levels;

— sets a lower limit on the target failure measures, in a dangerous mode of failure, that can be claimed for a single E/E/PE safety-related system; for E/E/PE safety-related systems operating in:

  — a low demand mode of operation, the lower limit is set at an average probability of failure of 10^{-5} to perform its design function on demand,

  — a high demand or continuous mode of operation, the lower limit is set at a probability of a dangerous failure of 10^{-9} per hour;

  NOTE A single E/E/PE safety-related system does not necessarily mean a single-channel architecture.

— adopts a broad range of principles, techniques and measures to achieve functional safety for E/E/PE safety-related systems, but does not use the concept of fail safe, which may be of value when the failure modes are well defined and the level of complexity is relatively low – the concept of fail safe was considered inappropriate because of the full range of complexity of E/E/PE safety-related systems that are within the scope of the standard.

FUNCTIONAL SAFETY OF ELECTRICAL/ELECTRONIC/PROGRAMMABLE ELECTRONIC SAFETY-RELATED SYSTEMS

Part 6: Guidelines on the application of parts 2 and 3

1 Scope

1.1 This part contains information and guidelines on parts 2 and 3.

— Annex A gives a brief overview of the requirements of parts 2 and 3 and sets out the functional steps in their application.

— Annex B gives an example technique for calculating the probabilities of failure and should be read in conjunction with 7.4.4 of part 2 and annex D.

— Annex C gives a worked example of calculating diagnostic coverage and should be read in conjunction with 7.4.4.6 and 7.4.4.7 of part 2.

— Annex D gives a methodology for quantifying the effect of hardware-related common cause failures on the probability of failure.

— Annex E gives worked examples of the application of the software safety integrity tables specified in annex A of part 3 for safety integrity levels 2 and 3.

1.2 Parts 1, 2, 3 and 4 of this standard are basic safety publications, although this status does not apply in the context of low complexity E/E/PE safety-related systems (see 3.4.4 of part 4). As basic safety publications, they are intended for use by Technical Committees in the preparation of standards in accordance with the principles contained in ISO/IEC Guide 104 and ISO/IEC Guide 51. One of the responsibilities of a Technical Committee is, wherever applicable, to make use of basic safety publications in the preparation of its own publications. IEC 61508 is also intended for use as a stand-alone standard.

1.3 Figure 1 shows the overall framework for parts 1 to 7 of this standard and indicates the role that part 6 plays in the achievement of functional safety for E/E/PE safety-related systems.

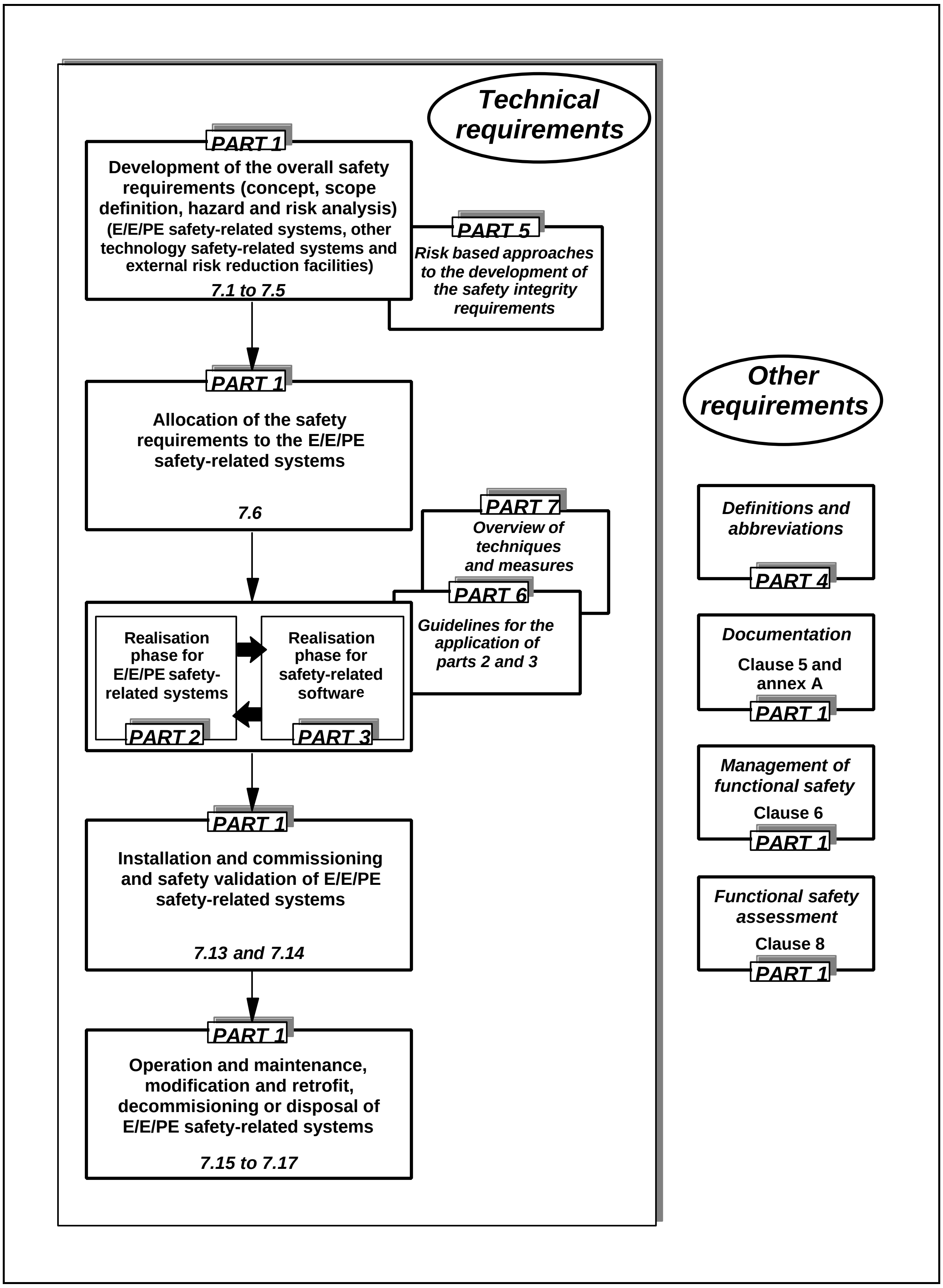

Technical requirements
PART 1
Development of the overall safety requirements (concept, scope definition, hazard and risk analysis) (E/E/PE safety-related systems, other technology safety-related systems and external risk reduction facilities)
7.1 to 7.5
PART 5
Risk based approaches to the development of the safety integrity requirements
PART 1
Allocation of the safety requirements to the E/E/PE safety-related systems
7.6
PART 7
Overview of techniques and measures
PART 6
Guidelines for the application of parts 2 and 3
Realisation phase for E/E/PE safety-related systems
PART 2
Realisation phase for safety-related software
PART 3
PART 1
Installation and commissioning and safety validation of E/E/PE safety-related systems
7.13 and 7.14
PART 1
Operation and maintenance, modification and retrofit, decommisioning or disposal of E/E/PE safety-related systems
7.15 to 7.17
Other requirements
Definitions and abbreviations
PART 4
Documentation
Clause 5 and annex A
PART 1
Management of functional safety
Clause 6
PART 1
Functional safety assessment
Clause 8
PART 1

Figure 1 — Overall framework of this standard

2 Definitions and abbreviations

For the purposes of this standard, the definitions and abbreviations given in part 4 apply.

Annex A
(informative)

Application of parts 2 and 3

A.1 General

Machinery, process plant and other equipment may, in the case of malfunction (for example by failures of electro-mechanical, electronic and/or programmable electronic devices), present risks to people from hazardous events such as fires, explosions, radiation overdoses, machinery traps etc. Failures can arise from either physical faults in the device (for example causing random hardware failures), or from systematic faults (for example human errors made in the specification and design of a system cause systematic failure under some particular combination of inputs), or from some environmental condition.

Part 1 provides an overall framework based on a risk approach for the prevention and/or control of failures in electro-mechanical, electronic, or programmable electronic devices.

The overall goal is to ensure that plant and equipment can be safely automated. A key objective of this standard is to prevent:

— failures of control systems triggering other events, which in turn could lead to danger (for example fire, release of toxic materials, repeat stroke of a machine etc); and

— undetected failures in protection systems (for example in an emergency shut down system), making the systems unavailable when needed for a safety action.

Part 1 requires that a hazard and risk analysis at the process/machine level is carried out to determine the amount of risk reduction necessary to meet the risk criteria for the application. Risk is based on the assessment of both the consequence (or severity) and the frequency (or probability) of the hazardous event.

Part 1 further requires that the amount of risk reduction established by the risk analysis is used to determine if one or more safety-related systems[1] are required and what safety functions (each with a specified safety integrity[2]) they are needed for.

Parts 2 and 3 take the safety functions and safety integrity requirements allocated to any system, *designated* as a E/E/PE safety-related system, by the application of part 1 and establish requirements for safety lifecycle activities which:

— are to be applied during the specification, design and modification of the hardware and software; and

— focus on means for preventing and/or controlling random hardware and systematic failures (the E/E/PES and software safety lifecycles[3]).

[1] Systems necessary for functional safety and containing one or more electrical (electro-mechanical), electronic or programmable electronic (E/E/PE) devices are *designated* as E/E/PE safety-related systems and include *all* equipment necessary to carry out the required safety function (see 3.4.1 of part 4).

[2] Safety integrity is specified as one of four discrete levels. Safety integrity level 4 is the highest and safety integrity level 1 the lowest (see 7.6.2.9 of part 1).

[3] To enable the requirements of this standard to be clearly structured, a decision was made to order the requirements using a development process model in which each stage follows in a defined order with little iteration (sometimes referred to as a waterfall model). However, it is stressed that any lifecycle approach can be used provided a statement of equivalence is given in the safety plan for the project (see clause 6 of part 1).

Parts 2 and 3 do not give guidance on which level of safety integrity is appropriate for a given required tolerable risk. This decision will depend upon many factors, including the nature of the application, the extent to which other systems carry out safety functions and social and economic factors (see parts 1 and 5).

The requirements of parts 2 and 3 include:

— the application of measures and techniques[1], which are graded against the safety integrity level, for the avoidance of systematic failures[2] by preventative methods; and

— the control of systematic failures (including software failures) and random hardware failures by design features such as fault detection, redundancy and software architectural features (for example diversity).

In part 2, assurance that the safety integrity target has been satisfied for dangerous random hardware failures is based on:

— hardware fault tolerance requirements (see tables 2 and 3 of part 2); and

— the diagnostic coverage and frequency of proof tests of subsystems and components, by carrying out a reliability analysis using *appropriate* data.

In both parts 2 and 3, assurance that the safety integrity target has been satisfied for systematic failures is gained by:

— the correct application of safety management procedures;

— the use of competent staff;

— the application of the specified safety lifecycle activities, including the specified techniques and measures[3]; and

— an independent functional safety assessment[4].

The overall goal is to ensure that remaining systematic faults will, commensurate with the safety integrity level, not cause a failure of the E/E/PE safety-related system.

Part 2 has been developed to provide requirements for achieving safety integrity in the hardware[5] of the E/E/PE safety-related systems including sensors and final elements. Techniques and measures against both random hardware failures and systematic hardware failures are required. These involve an appropriate combination of fault avoidance and failure control measures as indicated above. Where manual action is needed for functional safety, requirements are given for the operator interface. Also diagnostic test techniques and measures, based on software and hardware (for example diversity), to detect random hardware failures are specified in part 2.

---

[1] The required techniques and measures for each safety integrity level are shown in the tables in annexes A and B of parts 2 and 3.

[2] Systematic failures cannot usually be quantified. Causes include: specification and design faults in hardware and software; failure to take account of the environment (for example temperature); and operation-related faults (for example poor interface).

[3] Alternative measures to those specified in the standard are acceptable provided justification is documented during safety planning (see clause 6).

[4] Independent assessment does not always imply third party assessment - see clause 8 of part 1.

[5] Including fixed built-in software or software equivalents (also called firmware), such as application specific integrated circuits.

Part 3 has been developed to provide requirements for achieving safety integrity for the software –both embedded (including diagnostic fault detection services) and application software. Part 3 requires a combination of fault avoidance (quality assurance) and fault tolerance approaches (software architecture), as there is no known way to prove the absence of faults in reasonably complex safety-related software, especially the absence of specification and design faults. Part 3 requires the adoption of such software engineering principles as: top down design; modularity; verification of each phase of the development lifecycle; verified software modules and software module libraries; and clear documentation to facilitate verification and validation. The different levels of software require different levels of assurance that these and related principles have been correctly applied.

Close co-operation between the E/E/PES developer and the software developer is needed, particularly in developing the architecture of the programmable electronics where trade-offs between hardware and software architectures need to be considered for their safety impact (see figure 4 of part 2).

A.2 Functional steps

The functional steps in the application of part 2 are shown in figures A.1 and A.2. The functional steps in the application of part 3 are shown in figure A.3.

Functional steps for part 2 (see figures A.1 and A.2) are as follows.

a) Obtain the allocation of safety requirements (see part 1). Update the safety planning as appropriate during E/E/PES development.

b) Determine the requirements for E/E/PES safety, including the safety integrity requirements, for each safety function (see 7.2 of part 2). Allocate requirements to software and pass to software supplier and/or developer for the application of part 3.

c) Start the phase of planning for E/E/PES safety validation (see 7.3 of part 2).

d) Specify the architecture (configuration) for the E/E/PES logic system, sensors and final elements (see 7.4.1, 7.4.2 and 7.4.3 of part 2). Review with the software supplier/developer the hardware and software architecture and the safety implications of the trade-offs between the hardware and software (see figure 4 of part 2). Iterate if required.

e) Develop a model for the hardware architecture for the E/E/PE safety-related system. Develop this model by examining each safety function separately and determine the subsystem (component) that will be used to carry out this function (see 7.4.4.1 and 7.4.4.2 of part 2).

f) Establish the system parameters for each of the subsystems (components) used in the E/E/PE safety-related system. For each of the subsystems (components), determine the following:

— the proof test interval for failures which are not automatically revealed;

— the mean time to restoration;

— the diagnostic coverage (see 7.4.4.5 to 7.4.4.7 of part 2); and

— the probability of failure.

g) Create a reliability model for each of the safety functions that the E/E/PE safety-related system is required to carry out.

h) Calculate a reliability prediction for each subsystem using an appropriate technique. Compare the result with the target safety integrity level determined in b) above and the requirements of tables 2 and 3 of part 2 (see 7.4.5 of part 2). If the predicted safety integrity level does not meet the target safety integrity level and/or does not meet the requirements of tables 2 and 3 of part 2, then change:

— where possible, one or more of the subsystem parameters (go back to f) above); and/or

— the hardware architecture (go back to d) above).

NOTE A number of modelling methods are available and the analyst should choose w hich is the most appropriate (see 7.4.4.4 for a list of some methods that could be used).

i) Implement the design of the E/E/PE safety-related system. Select measures and techniques to control systematic hardware failures, failures caused by environmental influences and operational failures (see annex A of part 2).

j) Integrate the verified software (see part 3) onto the target hardware (see 7.5 of part 2 and annex B of part 2) and, in parallel, develop the procedures that users and maintenance staff will need to follow when operating the system (see 7.6 of part 2 and annex B of part 2). Include software aspects (see functional step f) for part 3 overleaf).

k) Together with the software developer (see 7.7 of part 3) , validate the E/E/PES (see 7.7 of part 2 and annex B of part 2).

l) Hand over the hardware and results of the E/E/PES safety validation to the system engineers for further integration into the overall system.

m) If maintenance/modification of the E/E/PES is required during operational life then re-activate this part 2 as appropriate (see 7.8 of part 2).

A number of activities run across the E/E/PES safety lifecycle. These include verification (see 7.9 of part 2) and functional safety assessment (see clause 8 of part 1).

In applying the above steps the E/E/PES safety techniques and measures appropriate to the required safety integrity level are selected. To aid in this selection, tables have been formulated, ranking the various techniques/measures against the four safety integrity levels (see annex B of part 2). Cross referenced to the tables is an overview of each technique and measure with references to further sources of information (see annexes A and B of part 7).

Annex B provides one possible technique for calculating the probabilities of failure for E/E/PE safety-related systems.

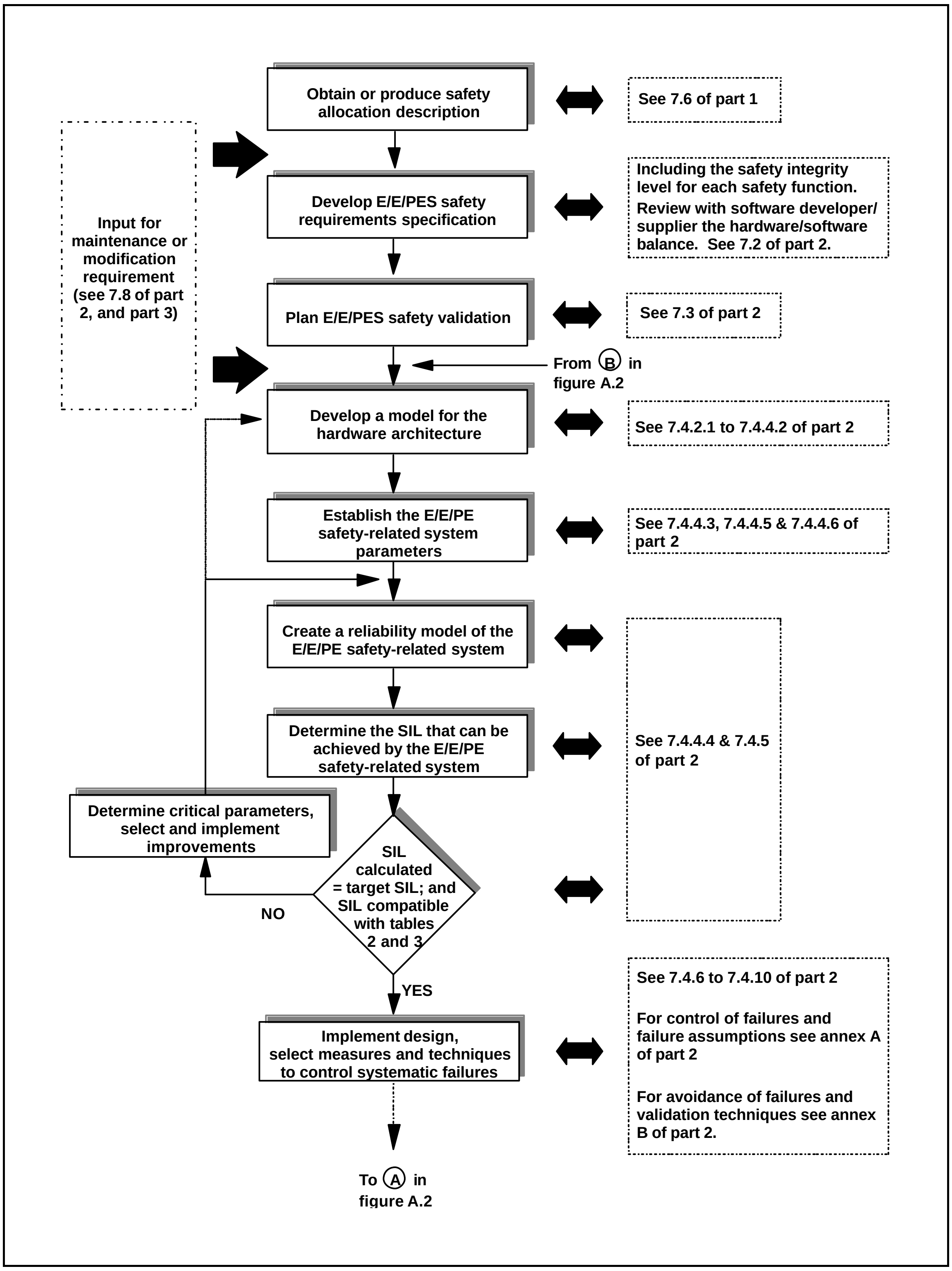

Obtain or produce safety allocation description
See 7.6 of part 1
Input for maintenance or modification requirement (see 7.8 of part 2, and part 3)
Develop E/E/PES safety requirements specification
Including the safety integrity level for each safety function.
Review with software developer/ supplier the hardware/software balance. See 7.2 of part 2.
Plan E/E/PES safety validation
See 7.3 of part 2
From B in figure A.2
Develop a model for the hardware architecture
See 7.4.2.1 to 7.4.4.2 of part 2
Establish the E/E/PE safety-related system parameters
See 7.4.4.3, 7.4.4.5 & 7.4.4.6 of part 2
Create a reliability model of the E/E/PE safety-related system
Determine the SIL that can be achieved by the E/E/PE safety-related system
See 7.4.4.4 & 7.4.5 of part 2
Determine critical parameters, select and implement improvements
SIL calculated = target SIL; and SIL compatible with tables 2 and 3
NO
YES
Implement design, select measures and techniques to control systematic failures
See 7.4.6 to 7.4.10 of part 2
For control of failures and failure assumptions see annex A of part 2
For avoidance of failures and validation techniques see annex B of part 2.
To A in figure A.2

Figure A.1 — Application of part 2

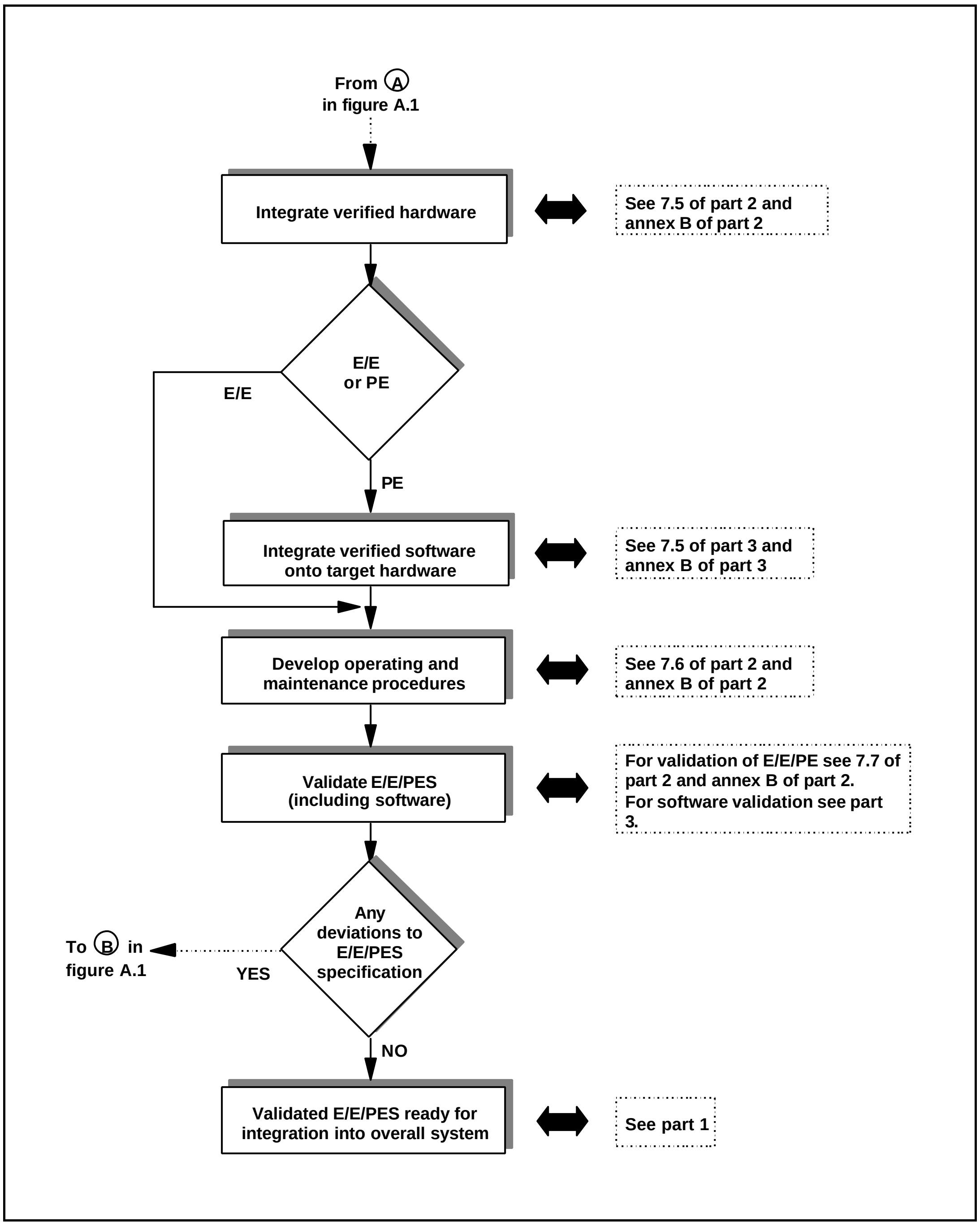

From Ⓐ in figure A.1
Integrate verified hardware
See 7.5 of part 2 and annex B of part 2
E/E or PE
E/E
PE
Integrate verified software onto target hardware
See 7.5 of part 3 and annex B of part 3
Develop operating and maintenance procedures
See 7.6 of part 2 and annex B of part 2
Validate E/E/PES (including software)
For validation of E/E/PE see 7.7 of part 2 and annex B of part 2. For software validation see part 3.
Any deviations to E/E/PES specification
To Ⓑ in figure A.1
YES
NO
Validated E/E/PES ready for integration into overall system
See part 1

Figure A.2 — Application of part 2 (continued)

Functional steps for part 3 (see figure A.3) are as follows.

a) Obtain the requirements for the E/E/PE safety-related systems and relevant parts of the safety planning (see 7.3 of part 2). Update the safety planning as appropriate during software development.

   NOTE Earlier lifecycle phases have already:

   — specified the required safety functions and their associated safety integrity levels (see 7.4 and 7.5 of part 1);

   — allocated the safety functions to designated E/E/PE safety-related systems (see 7.6 of part 1); and

   — allocated functions to software within each E/E/PE safety-related system (see 7.2 of part 2).

b) Determine the software architecture for all safety functions allocated to software (see 7.4 of part 3 and annex A of part 3).

c) Review with the E/E/PES supplier/developer the software and hardware architecture and the safety implications of the trade-offs between the software and hardware (see figure 4 of part 2). Iterate if required.

d) Start the planning for software safety verification and validation (see 7.3 and 7.9 of part 3).

e) Design, develop and verify/test the software according to the:

   — software safety planning,

   — software safety integrity level and

   — software safety lifecycle.

f) Complete final software verification activity and integrate the verified software onto the target hardware (see 7.5 of part 3), and in parallel develop the software aspects of the procedures that users and maintenance staff will need to follow when operating the system (see 7.6 of part 3, and part 2 functional step h) on previous page).

g) Together with the hardware developer (see 7.7 of part 2), validate the software in the integrated E/E/PE safety-related systems (see 7.7 of part 3).

h) Hand over the results of the software safety validation to the system engineers for further integration into the overall system.

i) If modification of the E/E/PES software is required during operational life then re-activate this part 3 phase as appropriate (see 7.8 of part 3).

A number of activities run across the software safety lifecycle. These include verification (see 7.9 of part 3) and functional safety assessment (see clause 8 of part 3).

In applying the above steps, software safety techniques and measures appropriate to the required safety integrity are selected. To aid in this selection, tables have been formulated ranking the various techniques/measures against the four safety integrity levels (see annex A of part 3). Cross referenced to the tables is an overview of each technique and measure with references to further sources of information (see annex C of part 7).

Worked examples in the application of the safety integrity tables are given in annex E, and part 7 includes a probabilistic approach to determining software safety integrity for pre-developed software (see annex D of part 7).

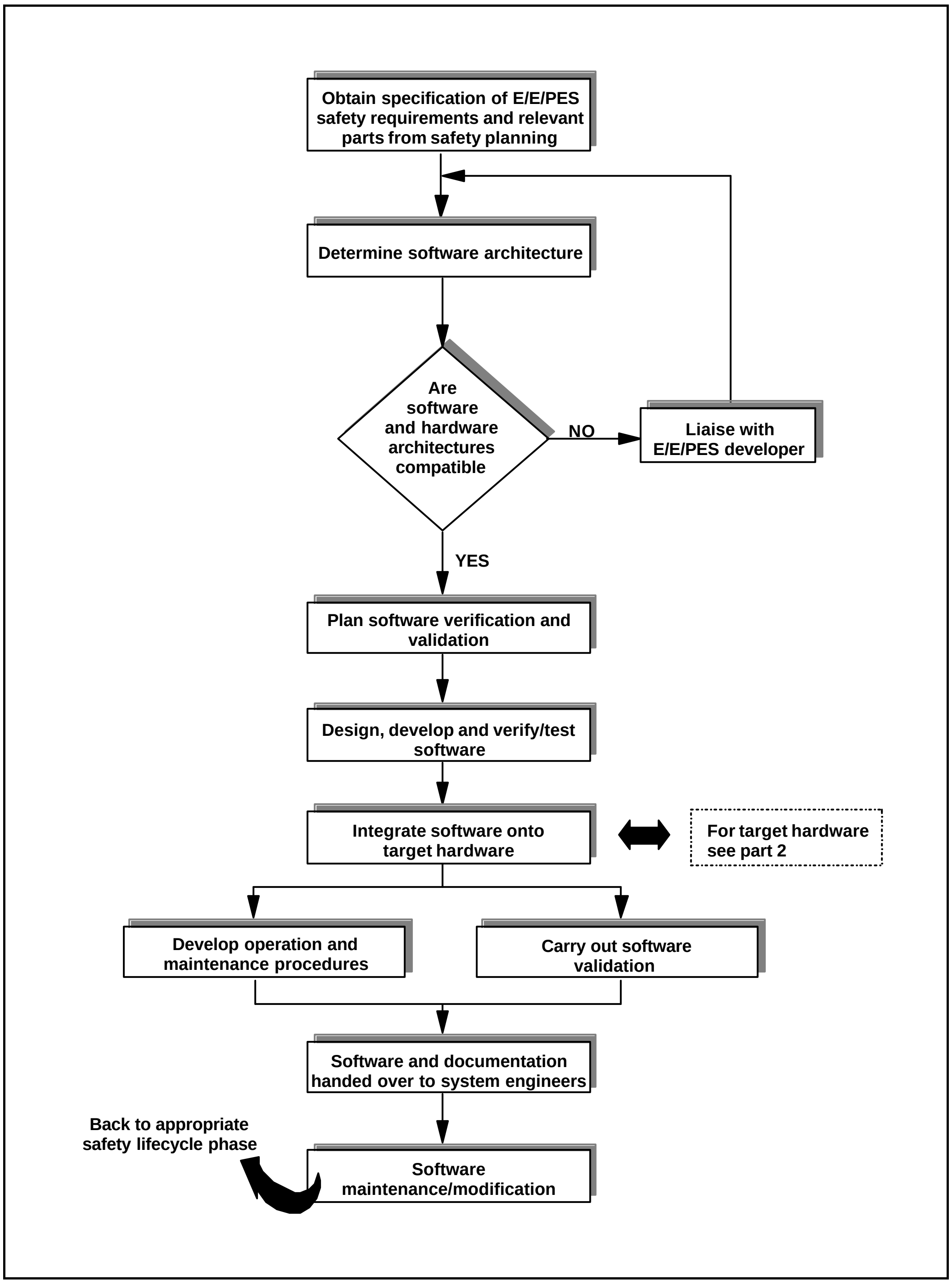

Obtain specification of E/E/PES safety requirements and relevant parts from safety planning
Determine software architecture
Are software and hardware architectures compatible
NO
Liaise with E/E/PES developer
YES
Plan software verification and validation
Design, develop and verify/test software
Integrate software onto target hardware
For target hardware see part 2
Develop operation and maintenance procedures
Carry out software validation
Software and documentation handed over to system engineers
Back to appropriate safety lifecycle phase
Software maintenance/modification

Figure A.3 — Application of part 3

Annex B
(informative)

Example technique for evaluating probabilities of failure

B.1 General

This annex provides one possible technique for calculating the probabilities of failure for E/E/PE safety related systems installed in accordance with parts 1 to 3. The information provided is informative in nature and should not be interpreted as the only evaluation technique that might be used. It does, however, provide a relatively simple approach for assessing the capability of E/E/PE safety-related systems.

There are a number of techniques available for the analysis of hardware safety integrity for E/E/PE safety-related systems. Two of the more common techniques are reliability block diagrams (see C.6.5 of part 7) and Markov models (see C.6.4 of part 7). Both methods, if correctly applied, will yield similar results, but in the case of complex programmable electronic subsystems (for example where multiple channel cross-voting and automatic testing are employed) there may be some loss of accuracy when reliability block diagrams are used compared to Markov models.

This loss of accuracy may not be significant in the context of the complete E/E/PE safety-related system and when the accuracy of the reliability data used in the analysis is taken into account. For example, field devices often predominate in the analysis of the hardware safety integrity for E/E/PE safety-related systems. Whether the loss of accuracy is significant can only be determined in the particular circumstances. In the case of complex programmable electronic subsystems, reliability block diagrams give results with more pessimistic hardware safety integrity values than Markov models (ie reliability block diagrams give a higher probability of failure).

This annex uses reliability block diagrams. The calculations are based on the following assumptions:

— component failure rates are constant over the life of the system;

— redundant components have similar failure rates;

— the sensor (input) component is from the actual sensing element up to (but not including) the first functional element which is combining the signal with the other sensors in the same voting group or into a specific protection function block (for example for two sensor channels, the configuration would be as shown in figure B.1);

— the logic system component is from the first functional element which is combining the input signals from other sensors in the same voting groups or into the same protection function block to the last function element which contains the same output for the logic groups or function block;

— the final element (output) component is after (ie not including) the output of the function element which contains the same output for the logic group or function block through to the final actuating elements within the safety system;

— the probabilities of failure are for a single channel of a (redundant) system (for example, if 2oo3 transmitters are used, the failure rate is for a single transmitter and the effect of 2oo3 is calculated separately);

— for each safety function, there is perfect proof testing and repair;

— the base β-factor for common cause failures, β, is two times the β-factor resulting from the diagnostic tests, β_D (for details on common cause failures see annex D);

— for each sub-system there is a single proof test interval and mean time to restoration;

— multiple repair teams are available to work on all known failures;

— the mean time to restoration is an order of magnitude less than the expected demand rate;

— for all components or subsystems operating in low demand mode of operation, and for 1oo2, 1oo2D and 2oo3 components or subsystems operating in high demand or continuous mode of operation, the fraction of failures specified by the diagnostic coverage will be both detected and repaired within the mean time to restoration used to determine hardware safety integrity requirements;

EXAMPLE If a mean time to restoration of 8 hours is assumed, this will include the diagnostic test interval which will typically be less than one hour, the remainder being the actual repair time.

NOTE For 1oo2, 1oo2D and 2oo3 components or subsystems, it is assumed that any repair is on-line. If an E/E/PE safety-related system is configured such that on any detected fault the EUC is put into a safe state, then the probability of failure on demand will be improved. The degree of improvement will be dependent on the diagnostic coverage.

— for 1oo1 and 2oo2 components or subsystems operating in high demand or continuous mode of operation, the E/E/PE safety-related system always shuts down safely after detecting a dangerous fault; to achieve this, the diagnostic test interval is a magnitude less than the expected demand rate;

— when a power supply failure removes power from a de-energize-to-trip E/E/PE safety-related system and initiates a system trip to a safe state, the power supply does not effect the average probability of failure on demand of the E/E/PE safety-related system; if the system is energized to trip, or the power supply has failure modes that can cause unsafe operation of the E/E/PE safety-related system, the power supply must be included in the evaluation;

— the abbreviated terms and considered values are as in table B.1.

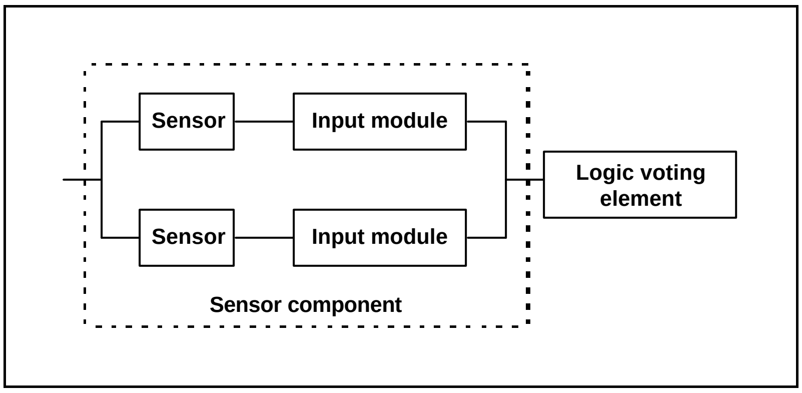


Figure B.1 — Example configuration for two sensor channels

Table B.1 — Terms and their ranges used in this annex
(applies to 1oo1, 1oo2, 2oo2, 1oo2D and 2oo3)

Abbreviation	**Term (units)**	**Parameter ranges in tables B.2 to B.5 and B.10 to B.13**
T_1	Proof test period (hour)	1 month (730 hours) - for high demand or continuous mode only 3 months (2190 hours) - for high demand or continuous mode only 6 months (4380 hours) 1 year (8760 hours) 2 years (17520 hours) - for low demand mode only 10 years (87600 hours) - for low demand mode only
MTTR	Mean time to restoration (hour)	8 hours
DC	Diagnostic coverage (percentage)	0% 60% 90% 99%
β	Fraction of failures having a common cause	1% 5% 10% (It is assumed $\beta = 2 \times \beta_D$)
λ	Average probability of failure (per demand or per hour)	0.1×10^{-6} 0.5×10^{-6} 1×10^{-6} 5×10^{-6} 10×10^{-6} 50×10^{-6}
λ_D	Probability of dangerous failure (per demand or per hour)	$0.5\ \lambda$ (assumes 50% dangerous failures and 50% safe failures)
λ_{DD}	Probability of detected dangerous failure (per demand or per hour)	(This is the sum of all the probabilities of detected dangerous failure within the channel)
λ_{DU}	Probability of undetected dangerous failure (per demand or per hour	(This is the sum of all the probabilities of undetected dangerous failure within the channel)
λ_{SD}	Probability of detected safe failure (per demand or per hour)	(This is the sum of all the probabilities of detected safe failure within the channel)
t_{DE}	Device equivalent mean down time (hour)	(This is the combined down time for all the components in a channel)
t_{SE}	System equivalent mean down time (hour)	(This is the combined down time for all the channels in a system)
PFD_{AVG}	Average probability of failure on demand	
λ_{AVG}	Average probability of failure per hour	

B.2 Average probability of failure per demand (for low demand mode of operation)

B.2.1 Procedure for calculations

B.2.1.1 General

The average probability of failure on demand of the E/E/PE safety-related system is determined by calculating and combining the average probability of failure on demand for all the subsystems which provide protection against a hazardous event. This is expressed by the following (see figure B.2):

$$PFD_{AVG} = \sum PFD_{SE} + \sum PFD_{LS} + \sum PFD_{PE}$$

where,

— PFD_{AVG} is the average probability of failure on demand of the E/E/PE safety-related system,

— PFD_{SE} is the probability of failure on demand of a sensor or input interface element,

— PFD_{LS} is the probability of failure on demand of a logic system element, and

— PFD_{FE} is the probability of failure on demand of an output interface element or final element.

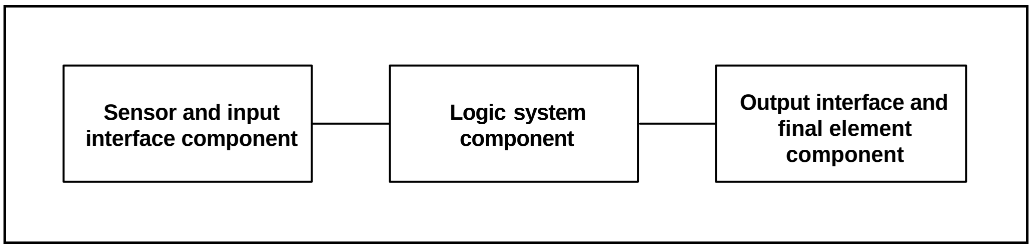

Sensor and input interface component
Logic system component
Output interface and final element component

Figure B.2 — Component structure

To determine the probability of failure on demand of each of the components (ie sensors, logic system and final elements), the following procedure should be adhered to for each component in turn.

a) Draw the block diagram showing the system input elements, logic system elements or system output elements. For example, system input elements may be sensors, barriers, input conditioning circuits; logic system elements may be processors and scanning devices; and system output elements may be output conditioning circuits, barriers and final elements. Represent each set of elements as 1oo1, 1oo2, 2oo2, 1oo2D or 2oo3.

b) Refer to the relevant table from tables B.2 to B.5 which are for 6 month, 1 year, 2 year and 10 year proof test intervals. These tables also assume an 8 hour mean time to restoration for each failure once it has been revealed.

c) For each sub-system element, select from the relevant table from tables B.2 to B.5:

— architecture (for example, 2oo3);

— diagnostic coverage (for example, 60%);

— the probability of failure per hour, λ, for the element (for example, 5.0E-06);

— the common cause failure β-factor (for example, 1%).

NOTE The options given for β in tables B.2 to B.5 (of 1%, 5% and 10%) represent the base β-factor used for all common cause failures. It is assumed that $\beta = 2 \times \beta_D$, ie that the base β-factor is twice the β-factor for failures that are undetected despite the diagnostic tests.

d) Obtain, from the relevant table from tables B.2 to B.5, the probability of failure on demand.

e) The combined probability of failure on demand of the sensor or final element component, PFD_{SE} or PFD_{FE}, is given in the following equations, where PFD_i and PFD_j is the probability of failure on demand of each voted group of sensors and final elements respectively:

$$PFD_{SE} = \sum_i PFD_i$$

$$PFD_{FE} = \sum_{j} PFD_{j}$$

B.2.2 Detailed tables for low demand mode of operation

Table B.2 — Average probability of failure on demand for a proof test interval of 6 months and a mean time to restoration of 8 hours

Architecture	DC	λ = 1.0E-07			λ = 5.0E-07			λ = 1.0E-06		
		β =1%	β =5%	β =10%	β =1%	β =5%	β =10%	β =1%	β =5%	β =10%
1oo1 (see note)	**0%**	1.1E-04			5.5E-04			1.1E-03		
	60%	4.4E-05			2.2E-04			4.4E-04		
	90%	1.1E-05			5.7E-05			1.1E-04		
	99%	1.5E-06			7.5E-06			1.5E-05		
1oo2	**0%**	2.2E-06	1.1E-05	2.2E-05	1.1E-05	5.5E-05	1.1E-04	2.4E-05	1.1E-04	2.2E-04
	60%	8.8E-07	4.4E-06	8.8E-06	4.5E-06	2.2E-05	4.4E-05	9.1E-06	4.4E-05	8.8E-05
	90%	2.2E-07	1.1E-06	2.2E-06	1.1E-06	5.6E-06	1.1E-05	2.3E-06	1.1E-05	2.2E-05
	99%	2.6E-08	1.3E-07	2.6E-07	1.3E-07	6.5E-07	1.3E-06	2.6E-07	1.3E-06	2.6E-06
2oo2 (see note)	**0%**	2.2E-04			1.1E-03			2.2E-03		
	60%	8.8E-05			4.4E-04			8.8E-04		
	90%	2.3E-05			1.1E-04			2.3E-04		
	99%	3.0E-06			1.5E-05			3.0E-05		
1oo2D	**0%**	2.2E-06	1.1E-05	2.2E-05	1.1E-05	5.5E-05	1.1E-04	2.4E-05	1.1E-04	2.2E-04
	60%	8.8E-07	4.4E-06	8.8E-06	4.4E-06	2.2E-05	4.4E-05	8.9E-06	4.4E-05	8.8E-05
	90%	2.2E-07	1.1E-06	2.2E-06	1.1E-06	5.6E-06	1.1E-05	2.2E-06	1.1E-05	2.2E-05
	99%	2.6E-08	1.3E-07	2.6E-07	1.3E-07	6.5E-07	1.3E-06	2.6E-07	1.3E-06	2.6E-06
2oo3	**0%**	2.2E-06	1.1E-05	2.2E-05	1.2E-05	5.6E-05	1.1E-04	2.7E-05	1.1E-04	2.2E-04
	60%	8.9E-07	4.4E-06	8.8E-06	4.6E-06	2.2E-05	4.4E-05	9.6E-06	4.5E-05	8.9E-05
	90%	2.2E-07	1.1E-06	2.2E-06	1.1E-06	5.6E-06	1.1E-05	2.3E-06	1.1E-05	2.2E-05
	99%	2.6E-08	1.3E-07	2.6E-07	1.3E-07	6.5E-07	1.3E-06	2.6E-07	1.3E-06	2.6E-06
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.2 *(concluded)*

Architecture	DC	λ = 5.0E-06			λ = 1.0E-05			λ = 5.0E-05		
		β =1%	β =5%	β =10%	β =1%	β =5%	β =10%	β =1%	β =5%	β =10%
1oo1 (see note)	**0%**	5.5E-03			1.1E-02			5.5E-02		
	60%	2.2E-03			4.4E-03			2.2E-02		
	90%	5.7E-04			1.1E-03			5.7E-03		
	99%	7.5E-05			1.5E-04			7.5E-04		
1oo2	**0%**	1.5E-04	5.8E-04	1.1E-03	3.7E-04	1.2E-03	2.3E-03	5.0E-03	8.8E-03	1.4E-02
	60%	5.0E-05	2.3E-04	4.5E-04	1.1E-04	4.6E-04	9.0E-04	1.1E-03	2.8E-03	4.9E-03
	90%	1.2E-05	5.6E-05	1.1E-04	2.4E-05	1.1E-04	2.2E-04	1.5E-04	6.0E-04	1.2E-03
	99%	1.3E-06	6.5E-06	1.3E-05	2.6E-06	1.3E-05	2.6E-05	1.4E-05	6.6E-05	1.3E-04
2oo2 (see note)	**0%**	1.1E-02			2.2E-02			>1E-01		
	60%	4.4E-03			8.8E-03			4.4E-02		
	90%	1.1E-03			2.3E-03			1.1E-02		
	99%	1.5E-04			3.0E-04			1.5E-03		
1oo2D	**0%**	1.5E-04	5.8E-04	1.1E-03	3.7E-04	1.2E-03	2.3E-03	5.0E-03	8.8E-03	1.4E-02
	60%	4.6E-05	2.2E-04	4.4E-04	9.5E-05	4.5E-04	8.9E-04	6.0E-04	2.3E-03	4.5E-03
	90%	1.1E-05	5.6E-05	1.1E-04	2.2E-05	1.1E-04	2.2E-04	1.1E-04	5.6E-04	1.1E-03
	99%	1.3E-06	6.5E-06	1.3E-05	2.6E-06	1.3E-05	2.6E-05	1.3E-05	6.5E-05	1.3E-04
2oo3	**0%**	2.3E-04	6.5E-04	1.2E-03	6.8E-04	1.5E-03	2.5E-03	1.3E-02	1.5E-02	1.9E-02
	60%	6.3E-05	2.4E-04	4.6E-04	1.6E-04	5.1E-04	9.4E-04	2.3E-03	3.9E-03	5.9E-03
	90%	1.2E-05	5.7E-05	1.1E-04	2.7E-05	1.2E-04	2.3E-04	2.4E-04	6.8E-04	1.2E-03
	99%	1.3E-06	6.5E-06	1.3E-05	2.7E-06	1.3E-05	2.6E-05	1.5E-05	6.7E-05	1.3E-04

NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.

Table B.3 — Average probability of failure on demand for a proof test interval of 1 year and mean time to restoration of 8 hours

Architectur e	DC	l = 1.0E-07			l = 5.0E-07			l = 1.0E-06		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	2.2E-04			1.1E-03			2.2E-03		
	60%	8.8E-05			4.4E-04			8.8E-04		
	90%	2.2E-05			1.1E-04			2.2E-04		
	99%	2.6E-06			1.3E-05			2.6E-05		
1oo2	**0%**	4.4E-06	2.2E-05	4.4E-05	2.3E-05	1.1E-04	2.2E-04	5.0E-05	2.2E-04	4.4E-04
	60%	1.8E-06	8.8E-06	1.8E-05	9.0E-06	4.4E-05	8.8E-05	1.9E-05	8.9E-05	1.8E-04
	90%	4.4E-07	2.2E-06	4.4E-06	2.2E-06	1.1E-05	2.2E-05	4.5E-06	2.2E-05	4.4E-05
	99%	4.8E-08	2.4E-07	4.8E-07	2.4E-07	1.2E-06	2.4E-06	4.8E-07	2.4E-06	4.8E-06
2oo2 (see note)	**0%**	4.4E-04			2.2E-03			4.4E-03		
	60%	1.8E-04			8.8E-04			1.8E-03		
	90%	4.5E-05			2.2E-04			4.5E-04		
	99%	5.2E-06			2.6E-05			5.2E-05		
1oo2D	**0%**	4.4E-06	2.2E-05	4.4E-05	2.3E-05	1.1E-04	2.2E-04	5.0E-05	2.2E-04	4.4E-04
	60%	1.8E-06	8.8E-06	1.8E-05	8.9E-06	4.4E-05	8.8E-05	1.8E-05	8.8E-05	1.8E-04
	90%	4.4E-07	2.2E-06	4.4E-06	2.2E-06	1.1E-05	2.2E-05	4.4E-06	2.2E-05	4.4E-05
	99%	4.8E-08	2.4E-07	4.8E-07	2.4E-07	1.2E-06	2.4E-06	4.8E-07	2.4E-06	4.8E-06
2oo3	**0%**	4.6E-06	2.2E-05	4.4E-05	2.7E-05	1.1E-04	2.2E-04	6.2E-05	2.4E-04	4.5E-04
	60%	1.8E-06	8.8E-06	1.8E-05	9.5E-06	4.5E-05	8.8E-05	2.1E-05	9.1E-05	1.8E-04
	90%	4.4E-07	2.2E-06	4.4E-06	2.3E-06	1.1E-05	2.2E-05	4.6E-06	2.2E-05	4.4E-05
	99%	4.8E-08	2.4E-07	4.8E-07	2.4E-07	1.2E-06	2.4E-06	4.8E-07	2.4E-06	4.8E-06
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.3 *(concluded)*

Architectur e	DC	l = 5.0E-06			l = 1.0E-05			l = 5.0E-05		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	1.1E-02			2.2E-02			>1E-01		
	60%	4.4E-03			8.8E-03			4.4E-02		
	90%	1.1E-03			2.2E-03			1.1E-02		
	99%	1.3E-04			2.6E-04			1.3E-03		
1oo2	**0%**	3.7E-04	1.2E-03	2.3E-03	1.1E-03	2.7E-03	4.8E-03	1.8E-02	2.4E-02	3.2E-02
	60%	1.1E-04	4.6E-04	9.0E-04	2.8E-04	9.7E-04	1.8E-03	3.4E-03	6.6E-03	1.1E-02
	90%	2.4E-05	1.1E-04	2.2E-04	5.1E-05	2.3E-04	4.5E-04	3.8E-04	1.3E-03	2.3E-03
	99%	2.4E-06	1.2E-05	2.4E-05	4.9E-06	2.4E-05	4.8E-05	2.6E-05	1.2E-04	2.4E-04
2oo2 (see note)	**0%**	2.2E-02			4.4E-02			>1E-01		
	60%	8.8E-03			1.8E-02			8.8E-02		
	90%	2.2E-03			4.5E-03			2.2E-02		
	99%	2.6E-04			5.2E-04			2.6E-03		
1oo2D	**0%**	3.7E-04	1.2E-03	2.3E-03	1.1E-03	2.7E-03	4.8E-03	1.8E-02	2.4E-02	3.2E-02
	60%	9.4E-05	4.5E-04	8.8E-04	2.0E-04	9.0E-04	1.8E-03	1.5E-03	5.0E-03	9.3E-03
	90%	2.2E-05	1.1E-04	2.2E-04	4.5E-05	2.2E-04	4.4E-04	2.3E-04	1.1E-03	2.2E-03
	99%	2.4E-06	1.2E-05	2.4E-05	4.8E-06	2.4E-05	4.8E-05	2.4E-05	1.2E-04	2.4E-04
2oo3	**0%**	6.8E-04	1.5E-03	2.5E-03	2.3E-03	3.8E-03	5.6E-03	4.8E-02	5.0E-02	5.3E-02
	60%	1.6E-04	5.1E-04	9.4E-04	4.8E-04	1.1E-03	2.0E-03	8.4E-03	1.1E-02	1.5E-02
	90%	2.7E-05	1.2E-04	2.3E-04	6.4E-05	2.4E-04	4.6E-04	7.1E-04	1.6E-03	2.6E-03
	99%	2.5E-06	1.2E-05	2.4E-05	5.1E-06	2.4E-05	4.8E-05	3.1E-05	1.3E-04	2.5E-04
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.4 — Average probability of failure on demand for a proof test interval of 2 years and a mean time to restoration of 8 hours

Architecture	DC	l = 1.0E-07			l = 5.0E-07			l = 1.0E-06		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	4.4E-04			2.2E-03			4.4E-03		
	60%	1.8E-04			8.8E-04			1.8E-03		
	90%	4.4E-05			2.2E-04			4.4E-04		
	99%	4.8E-06			2.4E-05			4.8E-05		
1oo2	**0%**	9.0E-06	4.4E-05	8.8E-05	5.0E-05	2.2E-04	4.4E-04	1.1E-04	4.6E-04	8.9E-04
	60%	3.5E-06	1.8E-05	3.5E-05	1.9E-05	8.9E-05	1.8E-04	3.9E-05	1.8E-04	3.5E-04
	90%	8.8E-07	4.4E-06	8.8E-06	4.5E-06	2.2E-05	4.4E-05	9.1E-06	4.4E-05	8.8E-05
	99%	9.2E-08	4.6E-07	9.2E-07	4.6E-07	2.3E-06	4.6E-06	9.2E-07	4.6E-06	9.2E-06
2oo2 (see note)	**0%**	8.8E-04			4.4E-03			8.8E-03		
	60%	3.5E-04			1.8E-03			3.5E-03		
	90%	8.8E-05			4.4E-04			8.8E-04		
	99%	9.6E-06			4.8E-05			9.6E-05		
1oo2D	**0%**	9.0E-06	4.4E-05	8.8E-05	5.0E-05	2.2E-04	4.4E-04	1.1E-04	4.6E-04	8.9E-04
	60%	3.5E-06	1.8E-05	3.5E-05	1.8E-05	8.8E-05	1.8E-04	3.6E-05	1.8E-04	3.5E-04
	90%	8.8E-07	4.4E-06	8.8E-06	4.4E-06	2.2E-05	4.4E-05	8.8E-06	4.4E-05	8.8E-05
	99%	9.2E-08	4.6E-07	9.2E-07	4.6E-07	2.3E-06	4.6E-06	9.2E-07	4.6E-06	9.2E-06
2oo3	**0%**	9.5E-06	4.4E-05	8.8E-05	6.2E-05	2.3E-04	4.5E-04	1.6E-04	5.0E-04	9.3E-04
	60%	3.6E-06	1.8E-05	3.5E-05	2.1E-05	9.0E-05	1.8E-04	4.7E-05	1.9E-04	3.6E-04
	90%	8.9E-07	4.4E-06	8.8E-06	4.6E-06	2.2E-05	4.4E-05	9.6E-06	4.5E-05	8.9E-05
	99%	9.2E-08	4.6E-07	9.2E-07	4.6E-07	2.3E-06	4.6E-06	9.3E-07	4.6E-06	9.2E-06

NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.

Table B.4 *(concluded)*

Architecture	DC	l = 5.0E-06			l = 1.0E-05			l = 5.0E-05		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	2.2E-02			4.4E-02			>1E-01		
	60%	8.8E-03			1.8E-02			8.8E-02		
	90%	2.2E-03			4.4E-03			2.2E-02		
	99%	2.4E-04			4.8E-04			2.4E-03		
1oo2	**0%**	1.1E-03	2.7E-03	4.8E-03	3.3E-03	6.5E-03	1.0E-02	6.6E-02	7.4E-02	8.5E-02
	60%	2.8E-04	9.7E-04	1.8E-03	7.5E-04	2.1E-03	3.8E-03	1.2E-02	1.8E-02	2.5E-02
	90%	5.0E-05	2.3E-04	4.5E-04	1.1E-04	4.6E-04	9.0E-04	1.1E-03	2.8E-03	4.9E-03
	99%	4.7E-06	2.3E-05	4.6E-05	9.5E-06	4.6E-05	9.2E-05	5.4E-05	2.4E-04	4.6E-04
2oo2 (see note)	**0%**	4.4E-02			8.8E-02			>1E-01		
	60%	1.8E-02			3.5E-02			>1E-01		
	90%	4.4E-03			8.8E-03			4.4E-02		
	99%	4.8E-04			9.6E-04			4.8E-03		
1oo2D	**0%**	1.1E-03	2.7E-03	4.8E-03	3.3E-03	6.5E-03	1.0E-02	6.6E-02	7.4E-02	8.5E-02
	60%	2.0E-04	9.0E-04	1.8E-03	4.5E-04	1.8E-03	3.6E-03	4.3E-03	1.1E-02	1.9E-02
	90%	4.4E-05	2.2E-04	4.4E-04	8.9E-05	4.4E-04	8.8E-04	4.7E-04	2.2E-03	4.4E-03
	99%	4.6E-06	2.3E-05	4.6E-05	9.2E-06	4.6E-05	9.2E-05	4.6E-05	2.3E-04	4.6E-04
2oo3	**0%**	2.3E-03	3.7E-03	5.6E-03	8.3E-03	1.1E-02	1.4E-02	>1E-01	>1E-01	>1E-01
	60%	4.8E-04	1.1E-03	2.0E-03	1.6E-03	2.8E-03	4.4E-03	3.2E-02	3.5E-02	4.0E-02
	90%	6.3E-05	2.4E-04	4.6E-04	1.6E-04	5.1E-04	9.4E-04	2.4E-03	4.0E-03	6.0E-03
	99%	4.8E-06	2.3E-05	4.6E-05	1.0E-05	4.7E-05	9.2E-05	6.9E-05	2.5E-04	4.8E-04

NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.

Table B.5 — Average probability of failure on demand for a proof test interval of 10 years and a mean time to restoration of 8 hours

Architecture	DC	l = 1.0E-07			l = 5.0E-07			l = 1.0E-06		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	2.2E-03			1.1E-02			2.2E-02		
	60%	8.8E-04			4.4E-03			8.8E-03		
	90%	2.2E-04			1.1E-03			2.2E-03		
	99%	2.2E-05			1.1E-04			2.2E-04		
1oo2	**0%**	5.0E-05	2.2E-04	4.4E-04	3.7E-04	1.2E-03	2.3E-03	1.1E-03	2.7E-03	4.8E-03
	60%	1.9E-05	8.9E-05	1.8E-04	1.1E-04	4.6E-04	9.0E-04	2.7E-04	9.6E-04	1.8E-03
	90%	4.4E-06	2.2E-05	4.4E-05	2.3E-05	1.1E-04	2.2E-04	5.0E-05	2.2E-04	4.4E-04
	99%	4.4E-07	2.2E-06	4.4E-06	2.2E-06	1.1E-05	2.2E-05	4.5E-06	2.2E-05	4.4E-05
2oo2 (see note)	**0%**	4.4E-03			2.2E-02			4.4E-02		
	60%	1.8E-03			8.8E-03			1.8E-02		
	90%	4.4E-04			2.2E-03			4.4E-03		
	99%	4.5E-05			2.2E-04			4.5E-04		
1oo2D	**0%**	5.0E-05	2.2E-04	4.4E-04	3.7E-04	1.2E-03	2.3E-03	1.1E-03	2.7E-03	4.8E-03
	60%	1.8E-05	8.8E-05	1.8E-04	9.4E-05	4.4E-04	8.8E-04	2.0E-04	9.0E-04	1.8E-03
	90%	4.4E-06	2.2E-05	4.4E-05	2.2E-05	1.1E-04	2.2E-04	4.4E-05	2.2E-04	4.4E-04
	99%	4.4E-07	2.2E-06	4.4E-06	2.2E-06	1.1E-05	2.2E-05	4.4E-06	2.2E-05	4.4E-05
2oo3	**0%**	6.2E-05	2.3E-04	4.5E-04	6.8E-04	1.5E-03	2.5E-03	2.3E-03	3.7E-03	5.6E-03
	60%	2.1E-05	9.0E-05	1.8E-04	1.6E-04	5.0E-04	9.3E-04	4.7E-04	1.1E-03	2.0E-03
	90%	4.6E-06	2.2E-05	4.4E-05	2.7E-05	1.1E-04	2.2E-04	6.3E-05	2.4E-04	4.5E-04
	99%	4.4E-07	2.2E-06	4.4E-06	2.3E-06	1.1E-05	2.2E-05	4.6E-06	2.2E-05	4.4E-05

NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.

Table B.5 *(concluded)*

Architecture	DC	l = 5.0E-06			l = 1.0E-05			l = 5.0E-05		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	>1E-01			>1E-01			>1E-01		
	60%	4.4E-02			8.8E-02			>1E-01		
	90%	1.1E-02			2.2E-02			>1E-01		
	99%	1.1E-03			2.2E-03			1.1E-02		
1oo2	**0%**	1.8E-02	2.4E-02	3.2E-02	6.6E-02	7.4E-02	8.5E-02	>1E-01	>1E-01	>1E-01
	60%	3.4E-03	6.6E-03	1.1E-02	1.2E-02	1.8E-02	2.5E-02	>1E-01	>1E-01	>1E-01
	90%	3.8E-04	1.2E-03	2.3E-03	1.1E-03	2.8E-03	4.9E-03	1.8E-02	2.5E-02	3.5E-02
	99%	2.4E-05	1.1E-04	2.2E-04	5.1E-05	2.3E-04	4.5E-04	3.8E-04	1.3E-03	2.3E-03
2oo2 (see note)	**0%**	>1E-01			>1E-01			>1E-01		
	60%	8.8E-02			>1E-01			>1E-01		
	90%	2.2E-02			4.4E-02			>1E-01		
	99%	2.2E-03			4.5E-03			2.2E-02		
1oo2D	**0%**	1.8E-02	2.4E-02	3.2E-02	6.6E-02	7.4E-02	8.5E-02	>1E-01	>1E-01	>1E-01
	60%	1.5E-03	4.9E-03	9.2E-03	4.2E-03	1.1E-02	1.9E-02	7.1E-02	9.9E-02	>1E-01
	90%	2.3E-04	1.1E-03	2.2E-03	4.7E-04	2.2E-03	4.4E-03	3.0E-03	1.2E-02	2.3E-02
	99%	2.2E-05	1.1E-04	2.2E-04	4.4E-05	2.2E-04	4.4E-04	2.2E-04	1.1E-03	2.2E-03
2oo3	**0%**	4.8E-02	5.0E-02	5.3E-02	>1E-01	>1E-01	>1E-01	>1E-01	>1E-01	>1E-01
	60%	8.3E-03	1.1E-02	1.4E-02	3.2E-02	3.5E-02	4.0E-02	>1E-01	>1E-01	>1E-01
	90%	6.9E-04	1.5E-03	2.6E-03	2.3E-03	3.9E-03	5.9E-03	4.9E-02	5.4E-02	6.0E-02
	99%	2.7E-05	1.2E-04	2.3E-04	6.4E-05	2.4E-04	4.6E-04	7.1E-04	1.6E-03	2.6E-03

NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.

B.2.3 Examples for low demand mode of operation

B.2.3.1 Example 1

Consider a safety function requiring a SIL2 system. Suppose that the initial assessment for the system architecture, based on previous practice, is for one group of three analogue pressure sensors, voting 2oo3. The logic system is a redundant 1oo2D configured PES driving a single shutdown valve plus a single vent valve. This is shown in figure B.3. For the initial assessment, a proof test period of 1 year is assumed.

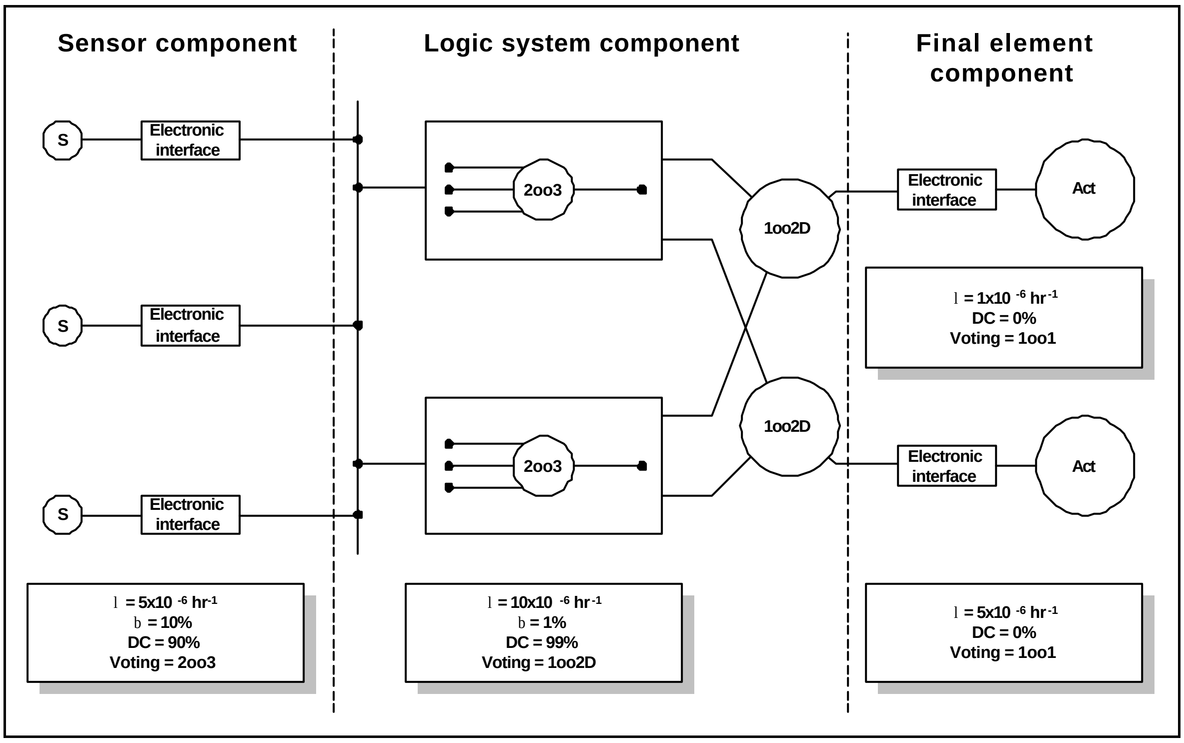


Figure B.3 — Architecture for example 1

Table B.6 — Average probability of failures on demand for the sensor elements in example 1 (proof test interval of a year and a mean time to restoration of 8 hours)

Architecture	DC	l = 5.0E-06		
		b = 1%	b = 5%	b = 10%
2oo3	0%	6.8E-04	1.5E-03	2.5E-03
	60%	1.6E-04	5.1E-04	9.4E-04
	90%	2.7E-05	1.2E-04	2.3E-04
	99%	2.5E-06	1.2E-05	2.4E-05
NOTE This table is abstracted from table B.3.				

Table B.7 — Average probability of failure on demand for the logic solver element in example 1 (proof test interval of a year and a mean time to restoration of 8 hours)

Architecture	**DC**	**l = 1.0E-05**		
		b = 1%	**b = 5%**	**b = 10%**
1oo2D	**0%**	1.1E-03	2.7E-03	4.8E-03
	60%	2.0E-04	9.0E-04	1.8E-03
	90%	4.5E-05	2.2E-04	4.4E-04
	99%	4.8E-06	2.4E-05	4.8E-05
NOTE This table is abstracted from table B.3.				

Table B.8 — Average probability of failure on demand for the final element in example 1 (proof test interval of a year and a mean time to restoration of 8 hours)

Architecture	**DC**	**l = 1.0E-06**			**l = 5.0E-06**		
		b = 1%	**b = 5%**	**b = 10%**	**b = 1%**	**b = 5%**	**b = 10%**
1oo1	**0%**	2.2E-03			1.1E-02		
	60%	8.8E-04			4.4E-03		
	90%	2.2E-04			1.1E-03		
	99%	2.6E-05			1.3E-04		
NOTE This table is abstracted from table B.3.							

From tables B.6 to B.8 we get the following values.

For the sensing element,

$PFD_{SE} = 2.3 \times 10^{-4}$

For the logic solving element,

$PFD_{LS} = 4.8 \times 10^{-6}$

For the final element,

$PFD_{FE} = 1.1 \times 10^{-2} + 2.2 \times 10^{-3}$

$= 1.3 \times 10^{-2}$

Therefore, for the safety function,

$PFD_{AVG} = 2.3 \times 10^{-4} + 4.8 \times 10^{-6} + 1.3 \times 10^{-2}$

$= 1.3 \times 10^{-2}$

º **safety integrity level 1**

To improve the system to meet safety integrity level 2, we do one of the following:

a) change the proof test interval to 6 months

PFD_{SE}	=	1.1×10^{-4}
PFD_{LS}	=	2.6×10^{-6}
PFD_{FE}	=	$5.5 \times 10^{-3} + 1.1 \times 10^{-3}$
	=	6.6×10^{-3}
PFD_{AVG}	=	6.7×10^{-3}
	º	**safety integrity level 2**

b) change the 1oo1 shutdown valve (which is the output device with the lower reliability) to 1oo2 (assuming $\beta = 5\%$) and change the proof test interval to 2 years.

PFD_{SE}	=	4.6×10^{-4}
PFD_{LS}	=	9.2×10^{-6}
PFD_{FE}	=	$2.7 \times 10^{-3} + 4.4 \times 10^{-3}$
	=	7.1×10^{-3}
PFD_{AVG}	=	7.6×10^{-3}
	º	**safety integrity level 2**

B.2.4 Architectures for low demand mode of operation

B.2.4.1 1oo1

This architecture consists of a single element, where any dangerous failure will prevent a valid alarm signal from being correctly processed.

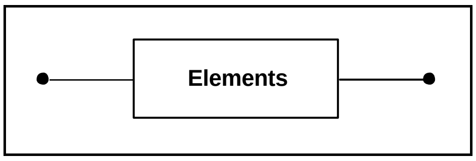


Figure B.4 — 1oo1 physical block diagram

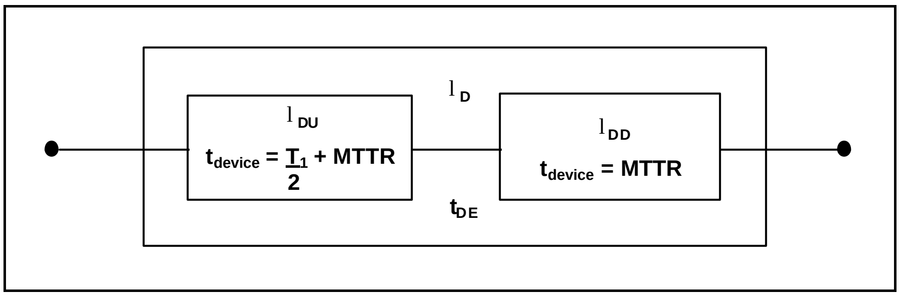


Figure B.5 — 1oo1 reliability block diagram

Figures B.4 and B.5 contain the relevant block diagrams. The probability of failure is given by:

$$l_D = l_{DU} + l_{DD} = \frac{l}{2}$$

When two elements are in series with different down times, in order to be able to calculate the system unavailability for different configurations (including multiple channel architectures – see B.2.4.2, B.2.4.3 and B.2.4.5) it is necessary to calculate a single equivalent down time which applies to the combined series elements. This is done by calculating the contribution from each element to the effective down time (t_{device}), in direct proportion to each individual component's contribution to the probability of failure of the channel:

$$t_{DE} = \frac{l_{DU}}{l_D}\left(\frac{T_1}{2} + MTTR\right) + \frac{l_{DD}}{l_D} MTTR$$

For every architecture the probability of detected dangerous failure and the probability of undetected dangerous failure are given by:

$$l_{DU} = \frac{l}{2}(1 - DC) \; ; \qquad l_{DD} = \frac{l}{2} DC$$

Hence, for a 1oo1 architecture, the average probability of failure is:

$$PFD_{AVG} = \left(l_{DU} + l_{DD}\right) t_{DE}$$

B.2.4.2 1oo2

This architecture consists of two elements connected in parallel, such that either element can command a shutdown output. Thus there would have to be a dangerous failure in both elements before a valid alarm signal would not be correctly processed. It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

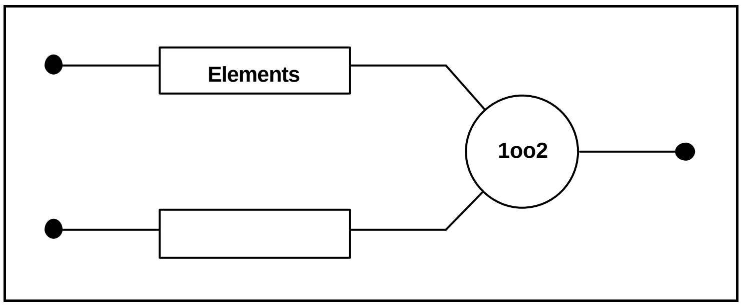


Figure B.6 — 1oo2 physical block diagram

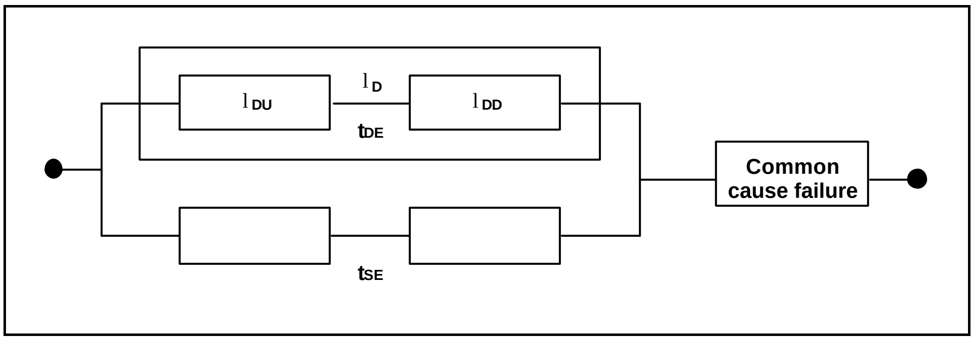


Figure B.7 — 1oo2 reliability block diagram

Figures B.6 and B.7 contain the relevant block diagrams. The value of t_{DE} is as given in B.2.4.1, but now it is necessary to also calculate t_{SE}, which is given by:

$$t_{SE} = \frac{l_{DU}}{l_D}\left(\frac{T_1}{3} + MTTR\right) + \frac{l_{DD}}{l_D} MTTR$$

The average probability of failure for the architecture is:

$$PFD_{AVG} = 2\left(\left(1-b\right) l_{DD} + \left(1-2b\right) l_{DU}\right)^2 t_{DE} t_{SE} + b\, l_{DD} MTTR + 2b\, l_{DU}\left(\frac{T_1}{2} + MTTR\right)$$

B.2.4.3 2oo2

This architecture consists of two elements connected in parallel, such that both elements need to demand a shutdown before a shutdown can take place. It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

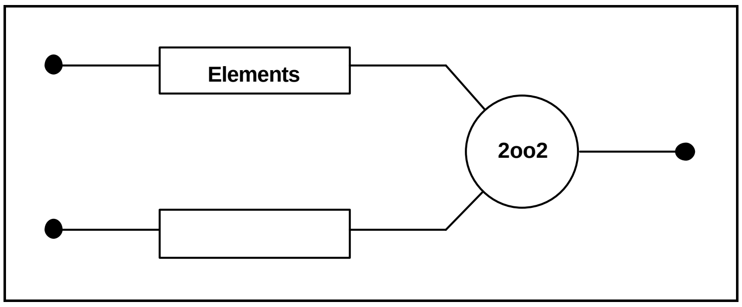


Figure B.8 — 2oo2 physical block diagram

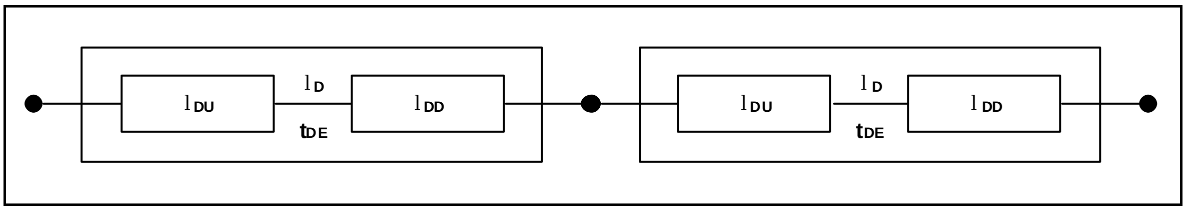


Figure B.9 — 2oo2 reliability block diagram

Figures B.8 and B.9 contain the relevant block diagrams. The value of t_{DE} is as given in B.2.4.1, and the average probability of failure for the architecture is:

$$PFD_{AVG} = 2\, l_D t_{DE}$$

B.2.4.4 1oo2D

This architecture consists of two elements connected in parallel. During normal operation, both elements need to demand a shutdown before a shutdown can take place. In addition, if the diagnostic tests in either element detect a fault then the output voting is adapted so that the system output state then follows that given by the other element. If the diagnostic tests find faults in both elements or a discrepancy that cannot be allocated to either element, then the output will go to the shutdown state. In order to detect a discrepancy between the elements, either element must be able to determine the state of the other element via a means independent of the other element.

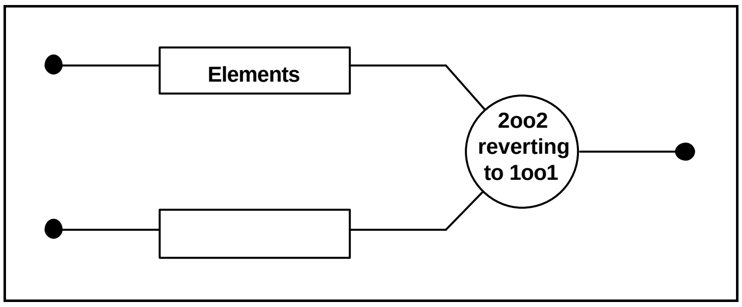


Figure B.10 — 1oo2D physical block diagram

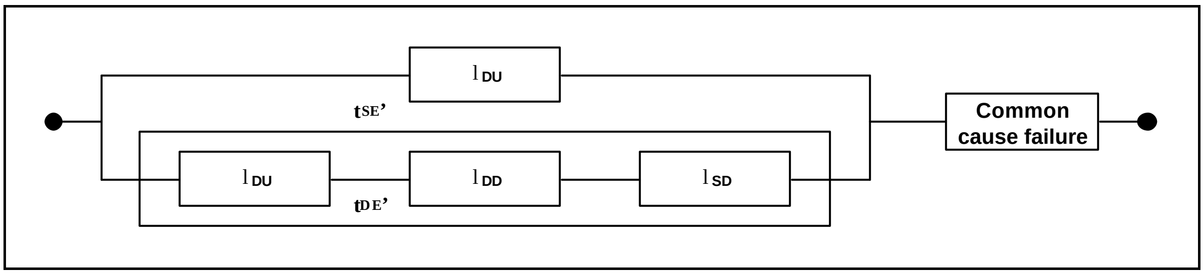


Figure B.11 — 1oo2D reliability block diagram

Figures B.10 and B.11 contain the relevant block diagrams. The values of the equivalent mean down times differ from those given for the other architectures in B.2.4 and hence are labelled t_{DE}' and t_{SE}'. Their values are given by:

$$t_{DE}' = \frac{l_{DU}\left(\frac{T_1}{2} + MTTR\right) + \left(l_{DD} + l_{SD}\right)MTTR}{l_{DU} + l_{DD} + l_{SD}}$$

$$t_{SE}' = \frac{l_{DU}\left(\frac{T_1}{3} + MTTR\right) + \left(l_{DD} + l_{SD}\right)MTTR}{l_{DU} + l_{DD} + l_{SD}}$$

The average probability of failure for the architecture is:

$$PFD_{AVG} = 2(1-2b)\,l_{DU}\left((1-2b)\,l_{DU} + (1-b)\,l_{DD} + l_{SD}\right)t_{DE}'\,t_{SE}' + b\,l_{DD}MTTR + 2\,b\,l_{DU}\left(\frac{T_1}{2} + MTTR\right)$$

B.2.4.5 2oo3

This architecture consists of three elements connected in parallel with a majority voting arrangement for the output signals, such that the output state is not changed if only one channel gives a different result which disagrees with the other two channels.

It is assumed that any diagnostic testing would only report the faults found and would not change any output states or change the output voting.

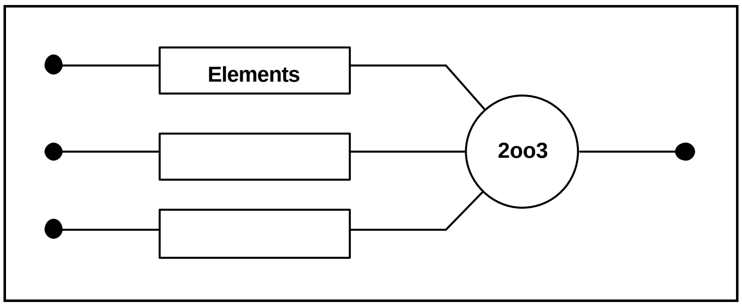


Figure B.12 — 2oo3 physical block diagram

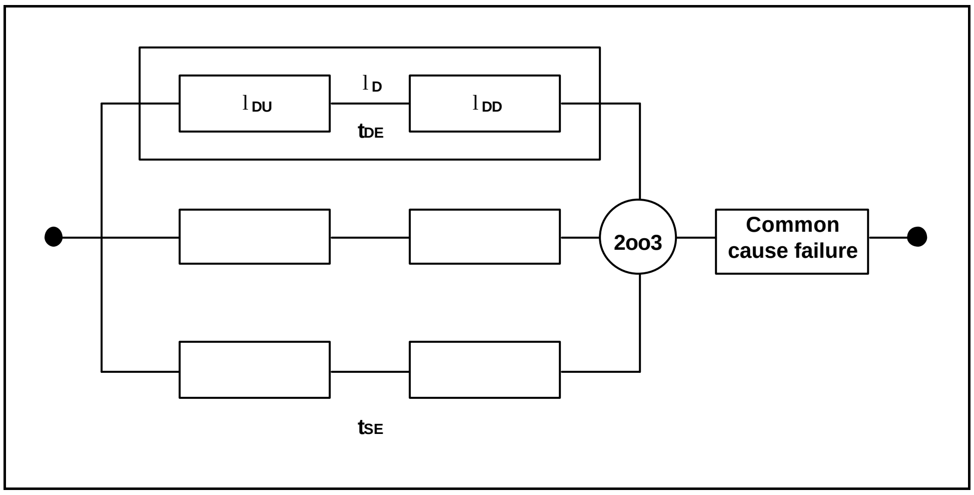


Figure B.13 — 2oo3 reliability block diagram

Figures B.12 and B.13 contain the relevant block diagrams. The value of t_{DE} is as given in B.2.4.1 and the value of t_{SE} is as given in B.2.4.2. The average probability of failure for the architecture is:

$$PFD_{AVG} = 6\left(\left(1-b\right) l_{DD} + \left(1-2b\right) l_{DU}\right)^2 t_{DE} t_{SE} + b\, l_{DD} MTTR + 2b\, l_{DU} \left(\frac{T_1}{2} + MTTR\right)$$

B.2.5 Effects of a non-perfect proof test

Faults in the safety system not detected by either self test or proof test will only be found by an incidence of demand that requires the safety function affected by the fault. Thus for these totally undetected faults, the effective down time will be governed by the expected demand rate on the safety system.

An example of this is given below for a 1oo2 system. T_2 in the equation is the time between demands on the system:

$$t_{DE} = \frac{l_{DU}}{2l_D}\left(\frac{T_1}{2} + MTTR\right) + \frac{l_{DU}}{2l_D}\left(\frac{T_2}{2} + MTTR\right) + \frac{l_{DD}}{l_D} MTTR$$

$$t_{SE} = \frac{l_{DU}}{2l_D}\left(\frac{T_1}{3} + MTTR\right) + \frac{l_{DU}}{2l_D}\left(\frac{T_2}{3} + MTTR\right) + \frac{l_{DD}}{l_D} MTTR$$

$$PFD = 2\left((1-b)\,l_{DD} + (1-2b)\,l_{DU}\right)^2 t_{DE}t_{SE} + b\,l_{DD}MTTR + 2b\,\frac{l_{DU}}{2}\left(\frac{T_1}{2} + MTTR\right) + 2b\,\frac{l_{DU}}{2}\left(\frac{T_2}{2} + MTTR\right)$$

Table B.9 below gives the numeric results of a 1oo2 system with a 100% one year proof test compared against a 50% proof test where the demand period T_2 is assumed to be 10 years. This example has been calculated assuming a failure rate of 10 x 10^{-6} per hour and a β-factor of 5%.

Table B.9 — Example for a non-perfect proof test

Architecture	DC	l = 1.0E-05	
		100% proof test	50% proof test (T_2 = 10 years)
		b = 5%	b = 5%
1oo2	0%	2.7E-03	6.6E-02
	60%	9.7E-04	2.6E-02
	90%	2.3E-04	6.6E-03
	99%	2.4E-05	7.0E-04

B.3 Probability of failure per hour (for high demand or continuous mode of operation)

B.3.1 Procedure for calculations

The method and procedure for calculating the probability of failure for an E/E/PE safety-related system operating in high demand or continuous mode of operation are identical with that for calculating for a low demand mode of operation, except that average probability of failure on demand (PFD_{AVG}) is replaced with probability of a dangerous failure per hour (λ_{AVG}).

The overall probability of a dangerous failure for the E/E/PE safety-related system, λ_{AVG}, is determined by calculating λ for all the subsystems which provide protection against a hazardous event and adding together these individual values. This is expressed by the following:

$$l_{AVG} = \sum l_{SE} + \sum l_{LS} + \sum l_{FE}$$

where,

— λ_{AVG} is the probability of failure per hour of the E/E/PE safety-related system;

— λ_{SE} is the probability of failure per hour of a sensor or input interface element;

— λ_{LS} is the probability of failure per hour of a logic system element; and

— λ_{FE} is the probability of failure per hour of an output interface element or final element.

B.3.2 Detailed tables for high demand or continuous mode of operation

Table B.10 — Probability of failure per hour (in high demand or continuous mode of operation) for a proof test interval of 1 month and a mean time to restoration of 8 hours

Architecture	**DC**	l **= 1.0E-07**			l **= 5.0E-07**			l **= 1.0E-06**		
		b **=1%**	b **=5%**	b **=10%**	b **=1%**	b **=5%**	b **=10%**	b **=1%**	b **=5%**	b **=10%**
1oo1 (see note)	**0%**	5.0E-08			2.5E-07			5.0E-07		
	60%	2.0E-08			1.0E-07			2.0E-07		
	90%	5.0E-09			2.5E-08			5.0E-08		
	99%	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0%**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.5E-09	1.8E-08	3.5E-08	7.1E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.5E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo2 (see note)	**0%**	1.0E-07			5.0E-07			1.0E-06		
	60%	4.0E-08			2.0E-07			4.0E-07		
	90%	1.0E-08			5.0E-08			1.0E-07		
	99%	1.0E-09			5.0E-09			1.0E-08		
1oo2D	**0%**	1.0E-09	5.0E-09	1.0E-08	5.0E-09	2.5E-08	5.0E-08	1.0E-08	5.0E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.5E-09	1.8E-08	3.5E-08	7.0E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.5E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo3	**0%**	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.6E-09	1.8E-08	3.5E-08	7.2E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.6E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.10 *(concluded)*

Architecture	**DC**	l **= 5.0E-06**			l **= 1.0E-05**			l **= 5.0E-05**		
		b **=1%**	b **=5%**	b **=10%**	b **=1%**	b **=5%**	b **=10%**	b **=1%**	b **=5%**	b **=10%**
1oo1 (see note)	**0%**	2.5E-06			5.0E-06			>1E-05		
	60%	1.0E-06			2.0E-06			1.0E-05		
	90%	2.5E-07			5.0E-07			2.5E-06		
	99%	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0%**	5.4E-08	2.5E-07	5.0E-07	1.2E-07	5.2E-07	1.0E-06	9.5E-07	2.9E-06	5.3E-06
	60%	3.7E-08	1.8E-07	3.5E-07	7.7E-08	3.6E-07	7.1E-07	5.4E-07	1.9E-06	3.6E-06
	90%	2.8E-08	1.4E-07	2.8E-07	5.7E-08	2.8E-07	5.5E-07	3.3E-07	1.4E-06	2.8E-06
	99%	2.5E-08	1.3E-07	2.5E-07	5.1E-08	2.5E-07	5.1E-07	2.7E-07	1.3E-06	2.5E-06
2oo2 (see note)	**0%**	5.0E-06			1.0E-05			>1E-05		
	60%	2.0E-06			4.0E-06			>1E-05		
	90%	5.0E-07			1.0E-06			5.0E-06		
	99%	5.0E-08			1.0E-07			5.0E-07		
1oo2D	**0%**	5.4E-08	2.5E-07	5.0E-07	1.2E-07	5.2E-07	1.0E-06	9.5E-07	2.9E-06	5.3E-06
	60%	3.6E-08	1.8E-07	3.5E-07	7.3E-08	3.5E-07	7.0E-07	4.3E-07	1.8E-06	3.6E-06
	90%	2.8E-08	1.4E-07	2.8E-07	5.5E-08	2.8E-07	5.5E-07	2.8E-07	1.4E-06	2.8E-06
	99%	2.5E-08	1.3E-07	2.5E-07	5.1E-08	2.5E-07	5.1E-07	2.5E-07	1.3E-06	2.5E-06
2oo3	**0%**	6.3E-08	2.6E-07	5.1E-07	1.5E-07	5.5E-07	1.0E-06	1.8E-06	3.6E-06	5.9E-06
	60%	4.1E-08	1.8E-07	3.5E-07	9.2E-08	3.7E-07	7.2E-07	9.1E-07	2.2E-06	3.9E-06
	90%	2.9E-08	1.4E-07	2.8E-07	6.2E-08	2.8E-07	5.6E-07	4.4E-07	1.5E-06	2.9E-06
	99%	2.6E-08	1.3E-07	2.5E-07	5.2E-08	2.5E-07	5.1E-07	3.0E-07	1.3E-06	2.6E-06
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.11 — Probability of failure per hour (in high demand or continuous mode of operation) for a proof test interval of 3 months and a mean time to restoration of 8 hours

Architecture	DC	l = 1.0E-07			l = 5.0E-07			l = 1.0E-06		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	0%	5.0E-08			2.5E-07			5.0E-07		
	60%	2.0E-08			1.0E-07			2.0E-07		
	90%	5.0E-09			2.5E-08			5.0E-08		
	99%	5.0E-10			2.5E-09			5.0E-09		
1oo2	0%	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.6E-09	1.8E-08	3.5E-08	7.2E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.6E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo2 (see note)	0%	1.0E-07			5.0E-07			1.0E-06		
	60%	4.0E-08			2.0E-07			4.0E-07		
	90%	1.0E-08			5.0E-08			1.0E-07		
	99%	1.0E-09			5.0E-09			1.0E-08		
1oo2D	0%	1.0E-09	5.0E-09	1.0E-08	5.1E-09	2.5E-08	5.0E-08	1.1E-08	5.0E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.5E-09	1.8E-08	3.5E-08	7.1E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.5E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo3	0%	1.0E-09	5.0E-09	1.0E-08	5.4E-09	2.5E-08	5.0E-08	1.2E-08	5.1E-08	1.0E-07
	60%	7.1E-10	3.5E-09	7.0E-09	3.7E-09	1.8E-08	3.5E-08	7.7E-09	3.6E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.7E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.11 *(concluded)*

Architecture	DC	l = 5.0E-06			l = 1.0E-05			l = 5.0E-05		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	0%	2.5E-06			5.0E-06			>1E-05		
	60%	1.0E-06			2.0E-06			1.0E-05		
	90%	2.5E-07			5.0E-07			2.5E-06		
	99%	2.5E-08			5.0E-08			2.5E-07		
1oo2	0%	6.3E-08	2.6E-07	5.1E-07	1.5E-07	5.4E-07	1.0E-06	1.8E-06	3.6E-06	5.9E-06
	60%	4.0E-08	1.8E-07	3.5E-07	9.2E-08	3.7E-07	7.2E-07	8.9E-07	2.2E-06	3.9E-06
	90%	2.9E-08	1.4E-07	2.8E-07	6.1E-08	2.8E-07	5.5E-07	4.2E-07	1.5E-06	2.9E-06
	99%	2.5E-08	1.3E-07	2.5E-07	5.1E-08	2.5E-07	5.1E-07	2.8E-07	1.3E-06	2.5E-06
2oo2 (see note)	0%	5.0E-06			1.0E-05			>1E-05		
	60%	2.0E-06			4.0E-06			>1E-05		
	90%	5.0E-07			1.0E-06			5.0E-06		
	99%	5.0E-08			1.0E-07			5.0E-07		
1oo2D	0%	6.3E-08	2.6E-07	5.1E-07	1.5E-07	5.4E-07	1.0E-06	1.8E-06	3.6E-06	5.9E-06
	60%	3.7E-08	1.8E-07	3.5E-07	7.9E-08	3.6E-07	7.1E-07	5.7E-07	1.9E-06	3.7E-06
	90%	2.8E-08	1.4E-07	2.8E-07	5.6E-08	2.8E-07	5.5E-07	2.9E-07	1.4E-06	2.8E-06
	99%	2.5E-08	1.3E-07	2.5E-07	5.1E-08	2.5E-07	5.1E-07	2.5E-07	1.3E-06	2.5E-06
2oo3	0%	9.0E-08	2.8E-07	5.3E-07	2.6E-07	6.3E-07	1.1E-06	4.5E-06	5.9E-06	7.6E-06
	60%	5.1E-08	1.9E-07	3.6E-07	1.4E-07	4.1E-07	7.5E-07	2.0E-06	3.2E-06	4.7E-06
	90%	3.2E-08	1.4E-07	2.8E-07	7.2E-08	2.9E-07	5.6E-07	7.1E-07	1.8E-06	3.1E-06
	99%	2.6E-08	1.3E-07	2.5E-07	5.3E-08	2.6E-07	5.1E-07	3.2E-07	1.3E-06	2.6E-06
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.12 — Probability of failure per hour (in high demand or continuous mode of operation) for a proof test interval of 6 months and a mean time to restoration of 8 hours

Architecture	DC	l = 1.0E-07			l = 5.0E-07			l = 1.0E-06		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	5.0E-08			2.5E-07			5.0E-07		
	60%	2.0E-08			1.0E-07			2.0E-07		
	90%	5.0E-09			2.5E-08			5.0E-08		
	99%	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0%**	1.0E-09	5.0E-09	1.0E-08	5.3E-09	2.5E-08	5.0E-08	1.1E-08	5.1E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.6E-09	1.8E-08	3.5E-08	7.4E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.6E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo2 (see note)	**0%**	1.0E-07			5.0E-07			1.0E-06		
	60%	4.0E-08			2.0E-07			4.0E-07		
	90%	1.0E-08			5.0E-08			1.0E-07		
	99%	1.0E-09			5.0E-09			1.0E-08		
1oo2D	**0%**	1.0E-09	5.0E-09	1.0E-08	5.3E-09	2.5E-08	5.0E-08	1.1E-08	5.1E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.5E-09	1.8E-08	3.5E-08	7.2E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.5E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo3	**0%**	1.0E-09	5.0E-09	1.0E-08	5.8E-09	2.6E-08	5.1E-08	1.3E-08	5.3E-08	1.0E-07
	60%	7.1E-10	3.5E-09	7.0E-09	3.8E-09	1.8E-08	3.5E-08	8.3E-09	3.6E-08	7.1E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.8E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.12 *(concluded)*

Architecture	DC	l = 5.0E-06			l = 1.0E-05			l = 5.0E-05		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	2.5E-06			5.0E-06			>1E-05		
	60%	1.0E-06			2.0E-06			1.0E-05		
	90%	2.5E-07			5.0E-07			2.5E-06		
	99%	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0%**	7.6E-08	2.7E-07	5.2E-07	2.1E-07	5.9E-07	1.1E-06	3.1E-06	4.7E-06	6.8E-06
	60%	4.6E-08	1.8E-07	3.6E-07	1.1E-07	3.9E-07	7.3E-07	1.4E-06	2.7E-06	4.3E-06
	90%	3.0E-08	1.4E-07	2.8E-07	6.6E-08	2.9E-07	5.6E-07	5.5E-07	1.6E-06	3.0E-06
	99%	2.6E-08	1.3E-07	2.5E-07	5.2E-08	2.5E-07	5.1E-07	2.9E-07	1.3E-06	2.6E-06
2oo2 (see note)	**0%**	5.0E-06			1.0E-05			>1E-05		
	60%	2.0E-06			4.0E-06			>1E-05		
	90%	5.0E-07			1.0E-06			5.0E-06		
	99%	5.0E-08			1.0E-07			5.0E-07		
1oo2D	**0%**	7.6E-08	2.7E-07	5.2E-07	2.1E-07	5.9E-07	1.1E-06	3.1E-06	4.7E-06	6.8E-06
	60%	3.9E-08	1.8E-07	3.5E-07	8.7E-08	3.7E-07	7.1E-07	7.8E-07	2.1E-06	3.8E-06
	90%	2.8E-08	1.4E-07	2.8E-07	5.6E-08	2.8E-07	5.5E-07	3.0E-07	1.4E-06	2.8E-06
	99%	2.5E-08	1.3E-07	2.5E-07	5.1E-08	2.5E-07	5.1E-07	2.5E-07	1.3E-06	2.5E-06
2oo3	**0%**	1.3E-07	3.2E-07	5.5E-07	4.2E-07	7.7E-07	1.2E-06	8.4E-06	9.2E-06	1.0E-05
	60%	6.7E-08	2.0E-07	3.7E-07	2.0E-07	4.6E-07	8.0E-07	3.6E-06	4.6E-06	6.0E-06
	90%	3.6E-08	1.5E-07	2.8E-07	8.8E-08	3.1E-07	5.8E-07	1.1E-06	2.1E-06	3.4E-06
	99%	2.6E-08	1.3E-07	2.5E-07	5.5E-08	2.6E-07	5.1E-07	3.6E-07	1.4E-06	2.6E-06
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.13 — Probability of failure per hour (in high demand or continuous mode of operation) for a proof test interval of 1 year and a mean time to restoration of 8 hours

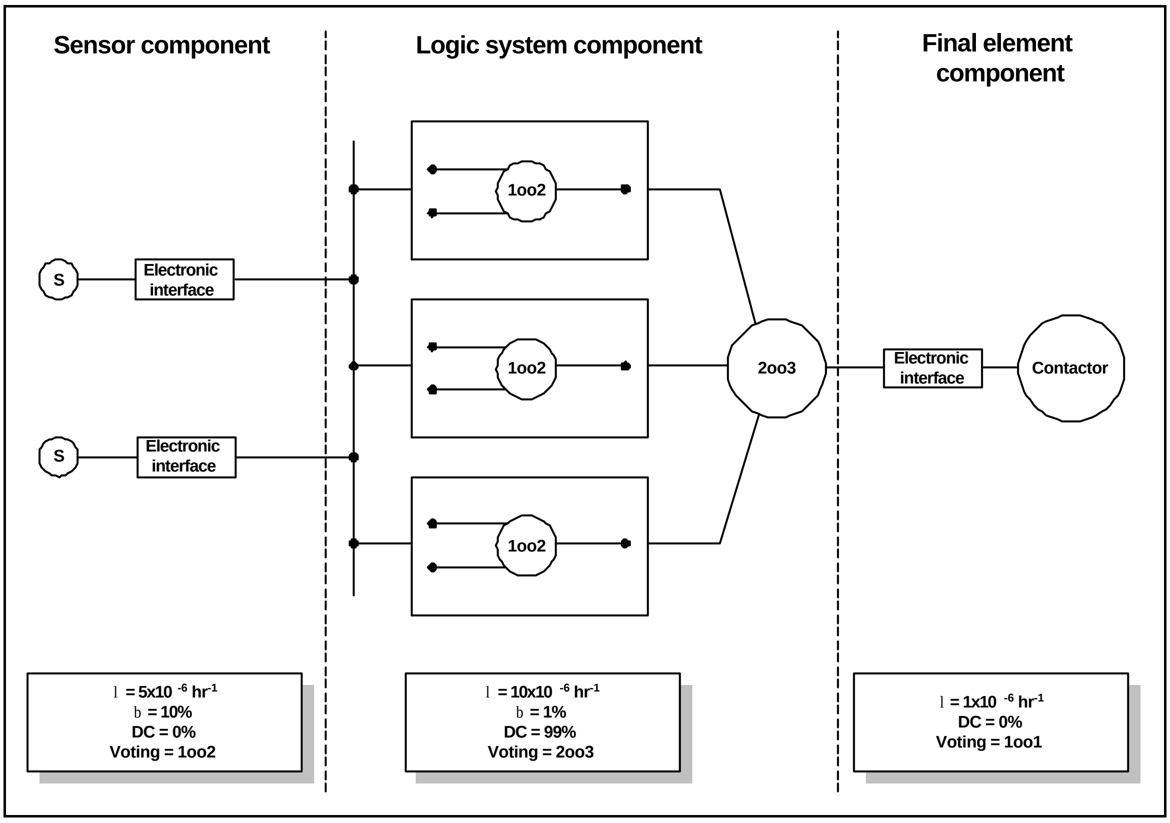
Architectur e	DC	l = 1.0E-07			l = 5.0E-07			l = 1.0E-06		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	5.0E-08			2.5E-07			5.0E-07		
	60%	2.0E-08			1.0E-07			2.0E-07		
	90%	5.0E-09			2.5E-08			5.0E-08		
	99%	5.0E-10			2.5E-09			5.0E-09		
1oo2	**0%**	1.0E-09	5.0E-09	1.0E-08	5.5E-09	2.5E-08	5.0E-08	1.2E-08	5.2E-08	1.0E-07
	60%	7.1E-10	3.5E-09	7.0E-09	3.7E-09	1.8E-08	3.5E-08	7.9E-09	3.6E-08	7.1E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.7E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo2 (see note)	**0%**	1.0E-07			5.0E-07			1.0E-06		
	60%	4.0E-08			2.0E-07			4.0E-07		
	90%	1.0E-08			5.0E-08			1.0E-07		
	99%	1.0E-09			5.0E-09			1.0E-08		
1oo2D	**0%**	1.0E-09	5.0E-09	1.0E-08	5.5E-09	2.5E-08	5.0E-08	1.2E-08	5.2E-08	1.0E-07
	60%	7.0E-10	3.5E-09	7.0E-09	3.6E-09	1.8E-08	3.5E-08	7.3E-09	3.5E-08	7.0E-08
	90%	5.5E-10	2.8E-09	5.5E-09	2.8E-09	1.4E-08	2.8E-08	5.5E-09	2.8E-08	5.5E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
2oo3	**0%**	1.1E-09	5.1E-09	1.0E-08	6.6E-09	2.6E-08	5.1E-08	1.6E-08	5.5E-08	1.0E-07
	60%	7.3E-10	3.5E-09	7.0E-09	4.1E-09	1.8E-08	3.5E-08	9.6E-09	3.7E-08	7.2E-08
	90%	5.6E-10	2.8E-09	5.5E-09	2.9E-09	1.4E-08	2.8E-08	6.2E-09	2.8E-08	5.6E-08
	99%	5.1E-10	2.5E-09	5.1E-09	2.5E-09	1.3E-08	2.5E-08	5.1E-09	2.5E-08	5.1E-08
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

Table B.13 *(concluded)*

Architectur e	DC	l = 5.0E-06			l = 1.0E-05			l = 5.0E-05		
		b =1%	b =5%	b =10%	b =1%	b =5%	b =10%	b =1%	b =5%	b =10%
1oo1 (see note)	**0%**	2.5E-06			5.0E-06			>1E-05		
	60%	1.0E-06			2.0E-06			1.0E-05		
	90%	2.5E-07			5.0E-07			2.5E-06		
	99%	2.5E-08			5.0E-08			2.5E-07		
1oo2	**0%**	1.0E-07	2.9E-07	5.4E-07	3.1E-07	6.8E-07	1.1E-06	5.8E-06	6.9E-06	8.5E-06
	60%	5.6E-08	1.9E-07	3.7E-07	1.6E-07	4.3E-07	7.7E-07	2.5E-06	3.7E-06	5.1E-06
	90%	3.3E-08	1.4E-07	2.8E-07	7.7E-08	2.9E-07	5.7E-07	8.2E-07	1.9E-06	3.2E-06
	99%	2.6E-08	1.3E-07	2.5E-07	5.3E-08	2.5E-07	5.1E-07	3.2E-07	1.3E-06	2.6E-06
2oo2 (see note)	**0%**	5.0E-06			1.0E-05			>1E-05		
	60%	2.0E-06			4.0E-08			>1E-05		
	90%	5.0E-07			1.0E-06			5.0E-06		
	99%	5.0E-08			1.0E-07			5.0E-07		
1oo2D	**0%**	1.0E-07	2.9E-07	5.4E-07	3.1E-07	6.8E-07	1.1E-06	5.8E-06	6.9E-06	8.5E-06
	60%	4.4E-08	1.8E-07	3.6E-07	1.0E-07	3.8E-07	7.3E-07	1.2E-06	2.5E-06	4.1E-06
	90%	2.8E-08	1.4E-07	2.8E-07	5.7E-08	2.8E-07	5.5E-07	3.3E-07	1.4E-06	2.8E-06
	99%	2.5E-08	1.3E-07	2.5E-07	5.1E-08	2.5E-07	5.1E-07	2.5E-07	1.3E-06	2.5E-06
2oo3	**0%**	2.1E-07	3.8E-07	6.1E-07	7.3E-07	1.0E-06	1.4E-06	>1E-05	>1E-05	>1E-05
	60%	9.9E-08	2.3E-07	4.0E-07	3.3E-07	5.8E-07	9.0E-07	6.8E-06	7.5E-06	8.4E-06
	90%	4.4E-08	1.5E-07	2.9E-07	1.2E-07	3.3E-07	6.0E-07	1.9E-06	2.9E-06	4.1E-06
	99%	2.7E-08	1.3E-07	2.5E-07	5.8E-08	2.6E-07	5.1E-07	4.4E-07	1.4E-06	2.7E-06
NOTE For 1oo1 and 2oo2 architectures, the value of β does not affect the average probability of failure.										

B.3.3 Example for high demand or continuous mode of operation

Consider a safety function requiring a SIL2 system. Suppose that the initial assessment for the system architecture, based on previous practice, is for one group of two sensors, voting 1oo2. The logic system is a redundant 2oo3 configured PES driving a single shutdown contactor. This is shown in the figure B.14. For the initial assessment, a proof test period of 6 months is assumed.



Sensor component
Logic system component
Final element component
S
Electronic interface
S
Electronic interface
1oo2
1oo2
1oo2
2oo3
Electronic interface
Contactor
l = 5x10 $^{-6}$ hr^{-1} b = 10% DC = 0% Voting = 1oo2
l = 10x10 $^{-6}$ hr^{-1} b = 1% DC = 99% Voting = 2oo3
l = 1x10 $^{-6}$ hr^{-1} DC = 0% Voting = 1oo1

Figure B.14 — Architecture for high demand or continuous mode example

Table B.14 —Probability of failure per hour for the example sensor element (proof test interval of 6 months and a mean time to restoration of 8 hours)

Architecture	DC	l = 5.0E-06		
		b = 1%	b = 5%	b = 10%
1oo2	0%	7.6E-08	2.7E-07	5.2E-07
	60%	4.6E-08	1.8E-07	3.6E-07
	90%	3.0E-08	1.4E-07	2.8E-07
	99%	2.6E-08	1.3E-07	2.5E-07
NOTE This table is abstracted from table B.12.				

Table B.15 — Probability of failure per hour for the example logic solver element (proof test interval of 6 months and a mean time to restoration of 8 hours)

Architecture	DC	l = 1.0E-05		
		b = 1%	b = 5%	b = 10%
2oo3	0%	4.2E-07	7.7E-07	1.2E-06
	60%	2.0E-07	4.6E-07	8.0E-07
	90%	8.8E-08	3.1E-07	5.8E-07
	99%	5.5E-08	2.6E-07	5.1E-07
NOTE This table is abstracted from table B.12.				

Table B.16 — Probability of failure per hour for the example final element (proof test interval of 6 months and a mean time to restoration of 8 hours)

Architecture	DC	l = 1.0E-06		
		b = 1%	b = 5%	b = 10%
1oo1	0%	5.0E-07		
	60%	2.0E-07		
	90%	5.0E-08		
	99%	5.0E-09		
NOTE This table is abstracted from table B.12.				

From tables B.14 to B.16 we get the following values.

For the sensing element,

$\lambda_{SE} = 5.2 \times 10^{-7}$ / hr

For the logic solving element,

$\lambda_{LS} = 5.5 \times 10^{-8}$ / hr

For the final element,

$\lambda_{FE} = 5.0 \times 10^{-7}$ / hr

Therefore, for the safety function,

$\lambda_{AVG} = 5.2 \times 10^{-7} + 5.5 \times 10^{-8} + 5.0 \times 10^{-7}$

$= 1.1 \times 10^{-6}$ / hr

º **safety integrity level 1**

To improve the system to meet safety integrity level 2, we do one of the following:

a) change the input sensor type and mounting to improve the defences against common cause failure, thus improving β from 10% to 5%;

λ_{SE}	=	$2.7\ x10^{-7}$ / hr
λ_{LS}	=	5.5×10^{-8} / hr
λ_{FE}	=	5.0×10^{-7} / hr
λ_{AVG}	=	8.3×10^{-7} / hr
	º	**safety integrity level 2**

b) change the single output device to two devices in 1oo2 (β = 5%).

λ_{SE}	=	5.2×10^{-7}
λ_{LS}	=	5.5×10^{-8} / hr
λ_{FE}	=	5.1×10^{-8} / hr
λ_{AVG}	=	6.3×10^{-7} / hr
	º	**safety integrity level 2**

B.3.4 Architectures for high demand or continuous mode of operation

B.3.4.1 1oo1

Figures B.4 and B.5 show the relevant block diagrams.

$$l_D = l_{DU} + l_{DD} = \frac{l}{2}$$

When you have two elements in series with different down times, in order to be able to calculate the system unavailability for different configurations (including multiple channel architectures – see B.3.4.2, B.3.4.3 and B.3.4.5), it is necessary to calculate single effective down time which is appropriate to the combined series elements. This is done by calculating the contribution from each element to the effective down time in direct proportion to each individual component's contribution to the probability of failure of the channel.

$$t_{DE} = \frac{l_{DU}}{l_D}\left(\frac{T_1}{2} + MTTR\right) + \frac{l_{DD}}{l_D} MTTR$$

$$l_{DU} = \frac{l}{2}(1 - DC); \qquad l_{DD} = \frac{l}{2} DC$$

If we assume that the safety system will put the EUC into a safe state on detection of any failure, we get:

$$l_{AVG} = l_{DU}$$

B.3.4.2 1oo2

Figures B.6 and B.7 show the relevant block diagrams.

$$l_{AVG} = 2\left(\left(1-b\right)l_{DD} + \left(1-2b\right)l_{DU}\right)^{2} t_{DE} + b\,l_{DD} + 2b\,l_{DU}$$

B.3.4.3 2oo2

Figures B.8 and B.9 show the relevant block diagrams. If we assume that each channel will be put into a safe state on detection of any fault, we get:

$$l_{AVG} = 2\,l_{DU}$$

B.3.4.4 1oo2D

Figures B.10 and B.11 show the relevant block diagrams.

$$t_{D} = \frac{l_{DU}\left(\frac{T_1}{2} + MTTR\right) + \left(l_{DD} + l_{SD}\right)MTTR}{l_{DU} + l_{DD} + l_{SD}}$$

$$l_{AVG} = 2\left(1-2b\right)l_{DU}\left(\left(1-2b\right)l_{DU} + \left(1-b\right)l_{DD} + l_{SD}\right)t_{D} + b\,l_{DD} + 2b\,l_{DU}$$

B.3.4.5 2oo3

Figures B.12 and B.13 show the relevant block diagrams.

$$l_{AVG} = 6\left(\left(1-b\right)l_{DD} + \left(1-2b\right)l_{DU}\right)^{2} t_{DE} + b\,l_{DD} + 2b\,l_{DU}$$

B.4 References

The following references give further details on evaluating probabilities of failure:

a) IEC 61078, *Analysis techniques for dependability – Reliability block diagram method*

b) IEC 61165, *Application of Markov techniques*

c) BS 5760*, Reliability of system equipment and components – Part 2: Guide to assessment of reliability*

d) D J Smith, *Reliability, maintainability and risk – Practical methods for engineers*, Butterworth-Heinemann, 5th edition, 1997, ISBN 0-7506-3752-8

e) R Billington and R N Allan, *Reliability evaluation of engineering systems*, Plenum, 1992, ISBN 0-306-44063-6

f) W M Goble, *Evaluating control system reliability – Techniques and applications*, Instrument Society of America, 1992, ISBN 1-55617-128-5

Annex C
(informative)

Calculation of diagnostic coverage: worked example

A method for calculating diagnostic coverage is given in 7.4.4.6 and 7.4.4.7 of part 2. This annex briefly describes the use of this method to calculate the diagnostic coverage of an E/E/PE safety-related system. It is assumed that all of the information specified in part 2 is available and has been used where required in obtaining the values shown in table C.1. Table C.2 gives limitations on diagnostic coverage that can be claimed for certain E/E/PE safety-related system components or subsystems. The values in table C.2 are based on engineering judgement.

To understand all the values in table C.1, a detailed hardware schematic would be required, from which the effect of all failure modes could be determined. These values are only examples, for instance some components in table C.1 assume no diagnostic coverage because it is practically impossible to detect all failure modes of all components.

Table C.1 has been derived as follows.

a) A failure mode and effect analysis has been carried out to determine the effect of each failure mode for every component on the behaviour of the system without diagnostic tests. The fractions of the overall failure rate associated with each failure mode are shown for each component, divided between safe (S) and dangerous (D) failures. The division between safe and dangerous failures may be deterministic for simple components but will otherwise be based on engineering judgement. For complex components, where a detailed analysis of each failure mode is not possible, a division of failures into 50% safe, 50% dangerous is generally accepted. For this table, the failure modes given in reference a) have been used, although other divisions between failure modes are possible and may be preferable.

b) The diagnostic coverage for each specific diagnostic test on each component is given (in the column labelled “DC_{comp}”). Specific diagnostic coverage’s are given for the detection of both safe and dangerous failures. Although open-circuit or short-circuit failures for simple components (for example resistors, capacitors and transistors) are shown to be detected with a specific diagnostic coverage of 100%, the use of table C.2 has limited the diagnostic coverage with respect to item U16, a complex type B component, to 90%.

c) The column labelled “λ” gives the probability, in the absence of diagnostic tests, of safe and dangerous failures for each component.

d) We can consider a detected dangerous failure to be effectively a safe failure, so we now find the division between effectively safe failures (ie either detected safe, undetected safe or detected dangerous failures) and undetected dangerous failures. The probability of effectively safe failure is found by multiplying the probability of dangerous failure by the specific diagnostic coverage for dangerous failure and adding the result to the probability of safe failure. Likewise, the probability of undetected dangerous failure is found by subtracting the specific diagnostic coverage for dangerous failure from one and multiplying the result by the probability of dangerous failure. The results are given in the column labelled “λ_{effect}”.

e) The last column labelled “λ_{det}” gives the probabilities of detected safe failures and detected dangerous failures found by multiplying the specific diagnostic coverage by the probability of failure.

f) The table yields the following results:

total probability of safe failure (including detected dangerous failures)	=	9.9×10^{-7}		
total probability of undetected dangerous failure	=	5.1×10^{-8}		
total probability of failure	=	1.0×10^{-6}		
total probability of undetected safe failure	=	2.7×10^{-8}		
diagnostic coverage for safe failures	=	3.38 / 3.65	=	93%
diagnostic coverage for dangerous failures (normally termed simply “diagnostic coverage”)	=	6.21 / 6.72	=	92%

g) The division of the probability of failure without diagnostic tests is 35% safe failures and 65% dangerous failures.

Table C.1 — Example calculations for diagnostic coverage

Item	**No**	**Type**	**Division of safe and dangerous failures for each failure mode**								**Division of safe and dangerous failures for diagnostic coverage’s and calculated probabilities of failure**							
			OC		**SC**		**Drift**		**Function**		**DC$_{comp}$**		**l (x 10^{-9})**		**l$_{effect}$ (x 10^{-9})**		**l$_{det}$ (x 10^{-9})**	
			S	**D**	**S**	**D**	**S**	**D**	**S**	**D**	**S**	**D**	**S**	**D**	**S**	**D**	**S**	**D**
Print	1	Print	0.5	0.5	0.5	0.5	0	0	0	0	0.99	0.99	11.0	11.0	21.9	0.1	10.9	10.9
CN1	1	Con96pin	0.5	0.5	0.5	0.5					0.99	0.99	11.5	11.5	22.9	0.1	11.4	11.4
C1	1	100nF	1	0	1	0	0	0	0	0	1	0	3.2	0.0	3.2	0.0	3.2	0.0
C2	1	10uF	0	0	1	0	0	0	0	0	1	0	0.8	0.0	0.8	0.0	0.8	0.0
R4	1	1M	0.5	0.5	0.5	0.5					1	1	1.7	1.7	3.3	0.0	1.7	1.7
R6	1	100k									0	0	0.0	0.0	0.0	0.0	0.0	0.0
OSC1	1	OSC24Mhz	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	1	1	16.0	16.0	32.0	0.0	16.0	16.0
U8	1	74HCT85	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.99	0.99	22.8	22.8	45.4	0.2	22.6	22.6
U16	1	MC68000-12	0	1	0	1	0.5	0.5	0.5	0.5	0.90	0.90	260.4	483.6	695.6	48.4	234.4	435.2
U26	1	74HCT74	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.99	0.99	22.8	22.8	45.4	0.2	22.6	22.6
U27	1	74F74	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.99	0.99	14.4	14.4	28.7	0.1	14.3	14.3
U28	1	PAL16L8A	0	1	0	1	0	1	0	1	0.98	0.98	0.0	88.0	86.2	1.8	0.0	86.2
T1	1	BC817	0	0	0	0.67	0	0.5	0	0	1	1	0.0	0.2	0.4	0.0	0.0	0.2
Total													365	672	986	50.9	338	621

NOTE None of the failure modes of item R6 are detected, but a failure does not affect either safety or availability.

Key

S	Safe failure
D	Dangerous failure
OC	Open circuit
SC	Short circuit
Drift	Change of value
Function	Functional failures
DC$_{comp}$	Specific diagnostic coverage for the component
λ	Probability of failure for the component, according to an agreed failure rate database (see 7.4.4.7 d) of part 2)
λ_{effect}	“Effective” probability of failure (by considering detected dangerous failures to be safe)
λ_{det}	Probability of detected failure

Table C.2 — Diagnostic coverage and effectiveness for different subsystems

Component	Low diagnostic coverage	Medium diagnostic coverage	High diagnostic coverage
CPU (see note 3)	total less than 70%	total less than 90%	
register, internal RAM	50 - 70%	85 - 90%	99 - 99.99%
coding and execution including flag register (see note 3)	50 - 60%	75 - 95%	-
address calculation (see note 3)	50 - 70%	85 - 98%	-
program counter, stack pointer	40 - 60%	60 - 90%	85 - 98%
Bus			
memory management unit	50%	70%	90 - 99%
bus-arbitration	50%	70%	90 - 99%
Interrupt handling	40 - 60%	60 - 90%	85 - 98%
Clock (quartz) (see note 4)	50%	-	95 - 99%
Program flow monitoring			
temporal (see note 3)	40 - 60%	60 - 80%	-
logical (see note 3)	40 - 60%	60 - 90%	-
temporal and logical (see note 5)	-	65 - 90%	90 - 98%
Invariable memory	50 - 70%	99%	99.99%
Variable memory	50 - 70%	85 - 90%	99 - 99.99%
Discrete hardware			
digital I/O	70%	90%	99%
analogue I/O	50 - 60%	70 - 85%	99%
power supply	50 - 60%	70 - 85%	99%
Communication and mass storage	90%	99.9%	99.99%
Electromechanical devices	90%	99%	99.9%
Sensors	50 - 70%	70 - 85%	99%
Final elements	50 - 70%	70 - 85%	99%

NOTE 1 This table should be read in conjunction with table A.1 of part 2 which provides the failure modes to be considered.

NOTE 2 When a range is given for diagnostic coverage, the upper interval boundaries may be set only for narrowly tolerated monitoring means, or for test measures that stress the function to be tested in a highly dynamic manner.

NOTE 3 For techniques where there is no high diagnostic coverage figure, at present no measures and techniques of high effectiveness are known.

NOTE 4 At present no measures and techniques of medium effectiveness are known for quartz clocks.

NOTE 5 The minimum diagnostic coverage for a combination of temporal and logical program flow monitoring is medium.

Useful references include the following:

a) Reliability Analysis Center (RAC), Failure Mode/Mechanism Distributions, 1991
Department of Defense, United States of America
PO Box 4700
201 Mill Street, Rome, NY 13440-8200
Organization report number: FMD-91
NSN 7540-01-280-5500

b) Qualität und Zuverlassigkeit technischer Systeme, Theorie, Praxis, Management, Dritte Auflage,1991
Allessandro Birolini
Springer-Verlag, Berlin Heidelberg New York
ISBN 3-540-54067-9 3 Aufl.
ISBN 0-387-54067-9 3 ed.

c) MIL-HDBK-217F, Military Handbook Reliability prediction of electronic equipment, 2 December 1991
Department of Defense, United States of America
Washington DC 20301

Annex D
(informative)

A methodology for quantifying the effect of hardware-related common cause failures in multi-channel programmable electronic systems

D.1 General

This standard incorporates a number of measures which deal with systematic failures. However, no matter how well these measures are applied, there will be a residual probability of systematic failures occurring. Although this will not have a significant effect on the reliability calculations for single-channel systems, the potential for failures which may affect more than one channel in a multi-channel system, ie common cause failures, will result in substantial errors when reliability calculations are applied to multi-channel systems.

This informative annex describes a methodology which will allow common cause failures to be taken into account in the safety assessment of a multi-channel PES. The use of the methodology will give a more accurate estimation of the integrity of such a system than if the potential for common cause failures were ignored. Alternative methodologies may be preferred in some cases, for example, where a more accurate β-factor can be proven as a result of the availability of data on common cause failures.

D.2 Brief overview

The failures of a system are considered to arise from two causes:

— random hardware failures; and

— systematic failures.

The former are assumed to occur randomly in time for any component and will result in a failure of a channel within a system of which the component forms part. There is a finite probability that independent random hardware failures could occur in all channels of a multi-channel system such that all of the channels were simultaneously in a failed state. Because random hardware failures are assumed to occur randomly with time, the probability of such failures concurrently affecting parallel channels is low compared to the probability of a single channel failing. This probability can be calculated using well-established techniques.

However, some failures, ie common cause failures which result from a single cause, may affect more than one channel. These may result from a systematic fault (for example, a design or specification mistake) or an external stress leading to an early random hardware failure (for example, an excessive temperature resulting from the random hardware failure of a common cooling fan, which accelerates the life of the components or takes them outside their specified operating environment) or, possibly, a combination of both. Because common cause failures are likely to affect more than one channel in a multi-channel system, the probability of common cause failure is likely to be the dominant factor in determining the overall probability of failure of a multi-channel system and this **must** be taken into account if a realistic estimate of the safety integrity level of the combined system is to be obtained.

Although common cause failures result from a single cause, they will not all manifest themselves simultaneously in all channels. For example, if a cooling fan fails, all of the channels of a multi-channel PE system could fail, leading to a common cause failure. However, it is unlikely that all of the channels will warm at the same rate or have the same critical temperature. Therefore, the failures will occur at different times in the different channels.

The architecture of programmable systems allows them to carry out internal diagnostic testing functions during their on-line operation. These can be employed in a number of ways, for example:

— A single channel PES can continuously be checking its internal operation together with the functionality of the input and output devices. If designed from the outset, a test coverage in the region of 99% is achievable (see reference a) in D.8). If 99% of internal faults are revealed before they can result in a failure, the probability of single-channel faults which can ultimately contribute to common cause failures will significantly be reduced.

— In addition to internal testing, each channel in a PES can monitor the outputs of other channels in a multi-channel PES (or each PE device can monitor another PE device in a multi-PE system). Therefore, if a failure occurs in one channel, this can be detected and a safe shut-down initiated by the one or more remaining channels that have not failed and are executing the cross-monitoring test. (It should be noted that cross-monitoring is effective only when the state of the control system is continuously changing, for example the interlock of a frequently used guard in a cyclic machine, or when brief changes can be introduced without affecting the controlled function.) This cross-monitoring can be carried out at a high rate, so that, when a non-simultaneous common cause failure is about to occur, it is likely that a cross-monitoring test will detect the failure of the first channel to fail and will be able to put the system into a safe state before a second channel is affected.

In the case of the cooling fan example, the rate of temperature rise and the susceptibility of each channel will be slightly different, resulting in the second channel failing possibly several tens of minutes after the first. This will allow the diagnostic testing to initiate a safe shutdown before the second channel succumbs to the common cause fault.

As a result of the above:

— PE-based systems have the potential to incorporate defences against common cause failures and, therefore, be less susceptible to them when compared to other technologies.

— A different β-factor may be applicable to PE-based systems when compared to other technologies. Therefore, β-factor estimates based on historic data are likely to be invalid. (None of the existing investigated models used for estimating the probability of common cause failure allow for the effect of automatic cross-monitoring.)

— Because common cause failures that are distributed in time may be revealed by the diagnostic tests before they affect all channels, such failures may not be recognized or reported as being common cause failures.

There are 3 avenues that can be taken to reduce the probability of potentially dangerous common cause failures.

a) Reduce the number of random hardware and systematic failures overall. (This will reduce the areas of the ellipses in figure D.1 leading to a reduction in the area of overlap.)

b) Maximize the independence of the channels. (This will reduce the amount of overlap between the ellipses in figure D.1 whilst maintaining their area.)

c) Reveal non-simultaneous common cause failures while only one, and before a second, channel has been affected, ie use diagnostic tests.

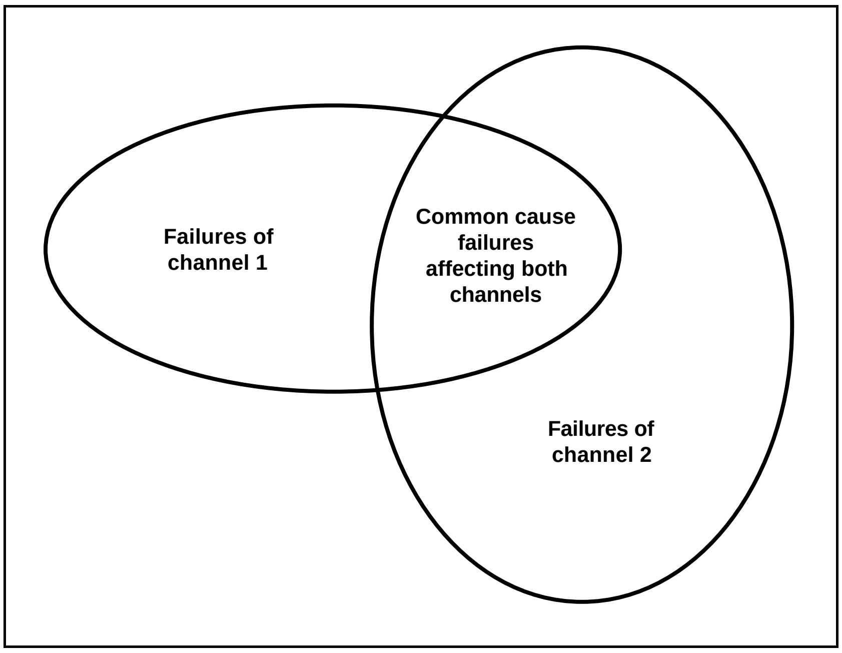


Figure D.1 — Relationship of common cause failures to the failures of individual channels

This standard is based on these 3 avenues and requires a threefold approach.

a) Apply the techniques specified in this standard to reduce the overall probability of systematic failure to a level commensurate with the probability of random hardware failure.

b) Quantify those factors that can be quantified, ie take into account the probability of random hardware failure, as specified in part 2.

c) Derive, by what is considered at the present time to be the best practicable means, a factor relating the probability of common cause failure of the hardware to the probability of random hardware failure. The methodology described in this annex relates to the derivation of this factor.

Most methodologies for estimating the probability of common cause failures attempt to make their predictions from the probability of random hardware failure. Clearly, the justification for any direct relationship between these probabilities is tenuous, nevertheless, such a correlation has been found in practice and probably results from second-order effects. For example, the higher the probability of random hardware failure of a system:

— the higher will be the amount of maintenance required by the system. The probability of a systematic fault being introduced during maintenance will depend on the number of times maintenance is carried out, and this will also affect the rate of human errors leading to common cause failures. This will lead to a relationship between the probability of random hardware failure and the probability of common cause failure. For example:

  — a repair, followed by testing and, possibly, recalibration will be required each time a random hardware failure occurs;

  — for a given safety integrity level, a system with a higher probability of random hardware failure will require proof tests to be carried out more frequently and with greater depth/complexity, leading to additional human interference.

— the more complex will be the system. The probability of random hardware failure will depend on the number of components, and, hence, the complexity of a system. A complex system will be less easily understood, so will be more prone to the introduction of systematic faults. In addition, the complexity will make it difficult to detect the faults, by either analysis or test, and could lead to parts of the logic of a system not being exercised except in infrequent circumstances. Again, this will lead to a relationship between the probability of random hardware failure and the probability of common cause failure.

Despite the limitations of the current models, it is believed that they represent the best way forward at the present time for providing an estimate of the probability of common cause failure of a multi-channel system. The methodology described in this annex uses an approach similar to the well-established β-factor model as the third part of the threefold approach already described.

The following two difficulties are faced when using the β-factor model on a PES.

— What value should be chosen for β? Many sources (for example reference a) in D.8) suggest ranges within which the value of β is likely to occur but no actual value is given, leaving the user to make a subjective choice. To overcome this problem, the methodology in this annex is based on the system originally described in reference b) of D.8 and recently refined in reference c) of D.8.

— The β-factor model does not take into account the sophisticated diagnostic testing capabilities of modern PESs, which can be used to detect a non-simultaneous common cause failure before it has had sufficient time to manifest itself fully. To overcome this deficiency, the approach described in references b) and c) of D.8 has been modified to reflect the effect of diagnostic tests in the estimation of the likely value of β.

The diagnostic testing functions running within a PES are continuously comparing the operation of the PES with predefined states. These states can be predefined in software or in hardware (for example, by a watchdog timer). Looked on in this way, the diagnostic testing functions may be thought of as an additional, and partially diverse, channel running in parallel with the PES.

Cross-monitoring between channels also can be carried out. For many years, this technique has been used in dual-channel interlocking systems based solely on relays. However, with relay technology, it is usually possible to carry out the cross-checks only when the channels change state, making such tests inappropriate for revealing non-simultaneous common cause failures where systems remain in the same (for example, ON) state for long periods. With PES technology, cross-monitoring may be carried out with a high repetition frequency.

D.3 Scope of the methodology

The scope of the methodology is limited to common cause failures within hardware. The reasons for this include:

— The β-factor model relates the probability of common cause failure to the probability of random hardware failure. The probability of common cause failures which involve the system as a whole will depend on the complexity of the system (possibly dominated by the user software) and not on the hardware alone. Clearly, any calculations based on the probability of random hardware failure cannot take into account the complexity of the software.

— Reporting of common cause failures will generally be limited to hardware failures - the area of most concern to the manufacturers of the hardware.

— It is not considered practicable to model systematic failures (for example software failures).

— The measures specified in part 3 are intended to reduce the probability of software-related common cause failure to an acceptable level for the target safety integrity level.

Therefore, the estimate of the probability of common cause failure derived by this methodology relates to only those failures associated with the hardware. It should NOT be assumed that the methodology can be used to obtain an overall failure rate which takes the probability of software-related failure into account.

D.4 Points taken into account in the methodology

Because the sensors, logic system and actuators will be subject to, for example, different environmental conditions and diagnostic tests with varying levels of capability, the methodology must be applied to each of these subsystems separately. For example, the logic systems are more likely to be in a controlled environment, whereas the sensors may be mounted outside on pipework that is exposed to the elements.

Programmable electronic channels have the potential for carrying out sophisticated diagnostic testing functions. These can:

— have a high diagnostic coverage within the channels;

— monitor additional redundancy channels;

— have a high repetition rate; and

— in an increasing number of cases, also monitor sensors and/or actuators.

A large fraction of common cause failures do not occur concurrently in all of the affected channels. Therefore, if the repetition frequency of the diagnostic tests is sufficiently high, a large fraction of common cause failures can be revealed and, hence, avoided before they affect all available channels.

Not all features of a multi-channel system, that will have a bearing on its immunity to common cause failures, will be affected by diagnostic tests. However, those features relating to diversity or independence will be made more effective. Any feature which is likely to increase the time between channel failures in a non-simultaneous common cause failure (or reduce the fraction of simultaneous common cause failures) will increase the probability of the diagnostic tests detecting the failure and putting the plant into a safe state. Therefore, the features relating to immunity to common cause failures have been divided into those whose effect is thought would be increased by the use of diagnostic tests and those whose effect would not. This leads to the two columns, X and Y respectively, in table D.1.

Although, for a 3 channel system, the probability of common cause failures which affect all 3 channels is likely to be slightly lower than the probability of failures which affect 2 channels, it is assumed, in order to simplify the methodology, that the probability is independent of the number of affected channels, ie it is assumed that if a common cause failure occurs it will affect all channels.

There is no known data on hardware-related common cause failures available for the calibration of the methodology. Therefore, the tables in this annex are based on engineering judgement.

Diagnostic test routines are sometimes not regarded as having a direct safety role so may not receive the same level of quality assurance as the routines providing the main control functions. The methodology was developed on the presumption that the diagnostic tests have an integrity commensurate with the target safety integrity level. Therefore, any software-based diagnostic test routines should be developed using techniques appropriate to the target safety integrity level.

D.5 Using b to calculate the probability of failure in an E/E/PE safety-related system due to common cause failures

Consider the effect of common cause failures on a multi-channel system with diagnostic tests running within each of its channels.

Using the β-factor model, the probability of dangerous common cause failures is

$$l_D b$$

where λ_D is the probability of dangerous random hardware failures for each individual channel and β is the β-factor in the absence of diagnostic tests, ie the fraction of single-channel failures that will affect all channels.

We shall assume that common cause failures affect all channels, and that the span of time between the first channel and all channels being affected is small compared to the time interval between common cause failures.

Suppose that there are diagnostic tests running in each channel and a fraction of the failures will be detected and revealed by the diagnostic tests. We can divide the failures into two: those that lie outside the coverage of the diagnostic tests (and so can never be detected) and those that lie within the coverage (so would eventually be detected by the diagnostic tests).

The overall probability of failure due to dangerous common cause failures is then given by

$$l_{DU} b + l_{DD} b_D$$

where,

— λ_{DU} is the probability of a undetected failure of a single channel, ie the probability of failures which lie outside the coverage of the diagnostic tests – clearly, any reduction in the β-factor resulting from the repetition rate of the diagnostic tests cannot affect this fraction of the failures;

— λ_{DD} is the probability of detected failure of a single channel, ie the probability of failures of a single channel that lie within the coverage of the diagnostic tests – here, if the repetition rate of the diagnostic tests is high, a fraction of the failures will be revealed leading to a reduction in the value of β, ie β_D;

— β is obtained from table D.4, using a score, $S = X + Y$ (see D.6);

— β_D is obtained from table D.4, using a score, $S_D = X(Z+1) + Y$.

D.6 Using the tables to estimate b

β should be calculated for the sensors, the logic system and the actuators separately.

In order to minimize the probability of occurrence of common cause failures, one must first establish which measures will lead to an efficient defence against their occurrence. The implementation of the appropriate measures in the system will lead to a reduction in the value of β used in estimating the probability of failure due to common cause failures.

In order to take the contribution of each of these measures into account, the methodology uses tables which list the measures and contain an associated value representing the contribution of each. Because sensors and actuators must be treated differently to the programmable electronics, separate columns are used in table D.1 for scoring the programmable electronics and the sensors or actuators.

Extensive diagnostic tests may be incorporated into programmable electronic systems which allow the detection of non-simultaneous common cause failures. To allow for these to be taken into account in the estimation of β, two sets of values are incorporated into the tables. One of these sets of values, in the column labelled Y, corresponds to those measures whose contribution will not be improved by the use of diagnostic tests. The other, in the column labelled X, is thought to lead to an improvement when diagnostic tests are in operation.

The user of table D.1 must ascertain which measures apply to the system in question, and sum the corresponding values shown in each of columns X_{LS} and Y_{LS} for the logic system, or X_{SA} and Y_{SA} for the sensors or actuators, the sums being referred to as X and Y, respectively.

Tables D.2 and D.3 may be used to determine a factor from the frequency and coverage of the diagnostic tests, Z, taking into account the important note in table D.2. The score S is then calculated using the following equations, as appropriate (see previous section):

— $S = X + Y$ to obtain the value of β (for undetected failures); and

— $S_D = X(Z+1) + Y$ to obtain the value of β_D (for detected failures).

Here S or S_D is a score which is used in table D.4 to determine the appropriate β-factor.

Table D.1 — Scoring programmable electronics or sensors/actuators

Item	X_{LS}	Y_{LS}	X_{SA}	Y_{SA}
Separation/segregation				
Are all signal cables for the channels routed separately at all positions?	1.5	1.5	1.0	2.0
Are the logic system channels on separate printed-circuit boards?	3.0	1.0		
Are the logic system channels in separate cabinets?	2.5	0.5		
If the sensors/actuators have dedicated control electronics, is the electronics for each channel on separate printed-circuit boards?			2.5	1.5
If the sensors/actuators have dedicated control electronics, is the electronics for each channel indoors and in separate cabinets?			2.5	0.5
Diversity/Redundancy				
Do the channels employ different electrical technologies? for example, one electronic or programmable electronic and the other relay	7.0			
Do the channels employ different electronic technologies? for example, one electronic, the other programmable electronic	5.0			
Do the devices employ different physical principles for the sensing elements for example, pressure and temperature, vane anemometer and Doppler transducer, etc.			7.5	
Do the devices employ different electrical principles/designs for example, digital and analogue, different manufacturer (not re-badged) or different technology			5.5	
Do the channels employ enhanced redundancy with MooN architecture, where N > M + 2	2.0	0.5	2.0	0.5
Do the channels employ enhanced redundancy with MooN architecture, where N = M + 2	1.0	0.5	1.0	0.5
Is low diversity used, for example hardware diagnostic tests using same technology	2.0	1.0		
Is medium diversity used, for example hardware diagnostic tests using different technology	3.0	1.5		
Were the channels designed by different designers with no communication between them during the design activities?	1.0	1.0		
Are separate test methods and people used for each channel during commissioning?	1.0	0.5	1.0	1.0
Is maintenance on each channel carried out by different people at different times?	2.5		2.5	
Complexity/design/application/maturity/experience				
Does cross-connection between channels preclude the exchange of any information other than that used for diagnostic testing or voting purposes?	0.5	0.5	0.5	0.5
Is the design based on techniques used in equipment that has been used successfully in the field for > 5 years?	0.5	1.0	1.0	1.0
Is there more than 5 years experience with the same hardware used in similar environments?	1.0	1.5	1.5	1.5
Is the system simple, for example no more than 10 inputs or outputs per channel?		1.0		
Are inputs and outputs protected from potential levels of over-voltage and over-current?	1.5	0.5	1.5	0.5
Are all devices/components conservatively rated? (for example, by a factor of 2 or more)	2.0		2.0	
Assessment/analysis and feedback of data				
Have the results of the failure modes and effects analysis or fault tree analysis been examined to establish sources of common cause failure and have predetermined sources of common cause failure been eliminated by design?		3.0		3.0
Were common cause failures considered in design reviews with the results fed back into the design? (Documentary evidence of the design review activity is required.)		3.0		3.0
Are all field failures fully analysed with feedback into the design? (Documentary evidence of the procedure is required.)	0.5	3.5	0.5	3.5
Procedures/human interface				
Is there a written system of work which will ensure that all component failures (or degradations) are detected, the root causes established and other similar items are inspected for similar potential causes of failure?		1.5	0.5	1.5

Table D.1 *(concluded)*

Item	X_{LS}	Y_{LS}	X_{SA}	Y_{SA}
Procedures/human interface (continued)				
Are procedures in place to ensure that: maintenance (including adjustment or calibration) of any part of the independent channels is staggered, and, in addition to the manual checks carried out following maintenance, the diagnostic tests are allowed to run satisfactorily between the completion of maintenance on one channel and the start of maintenance on another?	1.5	0.5	2.0	1.0
Do the documented maintenance procedures specify that all parts of redundant systems (for example, cables, etc.), intended to be independent of each other, must not be relocated?	0.5	0.5	0.5	0.50
Is all maintenance of printed-circuit boards, etc. carried out off-site at a qualified repair centre and have all the repaired items gone through a full pre-installation testing?	0.5	1.0	0.5	1.5
Do the system have low diagnostic coverage (60% to 90%) and report failures to the level of a field-replaceable module?	0.5			
Do the system have medium diagnostics coverage (90% to 99%) and report failures to the level of a field-replaceable module?	1.5	1.0		
Do the system have high diagnostics coverage (>99%) and report failures to the level of a field-replaceable module?	2.5	1.5		
Do the system diagnostic tests report failures to the level of a field-replaceable module?			1.0	1.0
Competence/training/safety culture				
Have designers been trained (with training documentation) to understand the causes and consequences of common cause failures	2.0	3.0	2.0	3.0
Have maintainers been trained (with training documentation) to understand the causes and consequences of common cause failures	0.5	4.5	0.5	4.5
Environmental control				
Is personnel access limited (for example locked cabinets, inaccessible position)?	0.5	2.5	0.5	2.5
Will the system be operating within the range of temperature, humidity, corrosion, dust, vibration, etc., over which it has been tested, without the use of external environmental control?	3.0	1.0	3.0	1.0
Are all signal and power cables separate at all positions?	2.0	1.0	2.0	1.0
Environmental testing				
Has a system been tested for immunity to all relevant environmental influences (for example EMC, temperature, vibration, shock, humidity) to an appropriate level as specified in recognized standards?	10.0	10.0	10.0	10.0
NOTE A number of the items relate to the operation of the system, which may be difficult to predict at design time. In these cases, the designers should make reasonable assumptions and subsequently ensure that the eventual user of the system is made aware of, for example, the procedures that must be put in place in order to achieve the designed level of safety integrity. This could be by including the necessary information in the accompanying documentation.				

Table D.2 — Diagnostic coverage: programmable electronics

Diagnostic coverage	**Value of Z to be used when the diagnostic tests are repeated at intervals of:**		
	Less than 1 minute	**Between 1 and 5 minutes**	**Greater than 5 minutes**
$\geq$ 99%	2.0	1.0	0
$\geq$ 90%	1.5	0.5	0
$\geq$ 60%	1.0	0	0

NOTE Diagnostic tests are effective against common cause failures only if an automatic shut-down, initiated by the diagnostic tests on the detection of a fault, can be assured before a non-simultaneous common cause failure would prevent a safe shut-down being implemented. If a safe shut-down is not initiated after a first fault then the following requirements will apply:

— the diagnostic tests shall determine the locality of the fault and be capable of localizing the fault;

— the diagnostic tests shall continue to be capable of placing the EUC in the safe state after the detection of any subsequent faults.

Only if the above criteria apply, can a non-zero value be used for Z.

Table D.3 — Diagnostic coverage: sensors or actuators

Diagnostic coverage	Value of Z to be used when the diagnostic tests are repeated at intervals of:			
	Less than 2 hours	Between 2 hours and 2 days	Between 2 days and 1 week	Greater than 1 week
≥ 99%	2.0	1.5	1.0	0
≥ 90%	1.5	1.0	0.5	0
≥ 60%	1.0	0.5	0	0

NOTE 1 In order for the methodology to be most effective, account must be taken uniformly across the list of the categories in table D.1. Therefore, it is strongly recommended that the total score in the X and Y columns for each category should not be less than the total score in the X and Y columns divided by 20. For example, if the total score (X + Y) is 80, none of the categories (for example, procedures/human interface) should have a total score (X + Y) of less than 4.

NOTE 2 When using table D.1, take account of the scores for all items that apply - the scoring has been designed to allow for items which are not mutually exclusive. For example, a system with logic system channels in separate racks is entitled to both the score for "Are the logic system channels in separate cabinets" and that for "Are the logic system channels on separate printed-circuit boards".

NOTE 3 If sensors or actuators are PE-based, they should be treated as part of the logic system if they are enclosed within the same building (or vehicle) as the device that constitutes the major part of the logic system, and as sensors or actuators if they are not so enclosed.

NOTE 4 For a non-zero value of Z to be used, it must be ensured that the equipment under control is put into a safe state before a non-simultaneous common cause failure can affect all the channels. The time taken to assure this safe state must be less than the claimed diagnostic test interval. A non-zero value for Z can be used only if:

— the system initiates an automatic shut-down on detection of a fault; or

— the system continues to run on a reduced number of channels and, following a fault, automatic shut-down will be assured immediately that the number of operating channels, that are capable of ensuring a safe shut-down, reduces to one[1]; or

— a formal system of work is in place to ensure that the cause of any revealed fault will be fully investigated within the claimed diagnostic test interval and:

  — if the fault has the potential for leading to a common cause failure, the plant will be immediately shut-down, or

  — the faulty channel will be repaired within the claimed diagnostic test interval.

NOTE 5 In the process industries, it is unlikely to be feasible to shut down the EUC when a fault is detected within the diagnostic test interval as described in table D.2. This methodology should not be interpreted as a requirement for process plants to be shut down when such faults are detected. However, if a shut down is not implemented, no reduction in the β-factor can be gained by the use of diagnostic tests for the programmable electronics. In some industries, a shut down may be feasible within the described time. In these cases, a non-zero value of Z may be used.

NOTE 6 Where diagnostic tests are carried out in a modular way, the repetition time used in tables D.2 or D.3 is the time between the successive completions of the full set of diagnostic testing modules. The diagnostic coverage is the total coverage provided by all of the modules.

[1] The operation of the system on the identification of a fault must be taken into account. For example, a simple 2oo3 system must be shut down (or be repaired) within the times quoted in tables D.2 or D.3, following the identification of a single failure. If this is not done, a failure of a second channel could result in the two failed channels outvoting the remaining (good) channel. A system which automatically reconfigures itself to 1oo2 voting when one channel fails, and which will automatically shut down on the occurrence of a second failure, has an increased probability of revealing the fault in the second channel and so a non-zero value for Z may be claimed.

Table D.4 — Calculation of b or b_D

Score (S or S_D)	**Corresponding value of b or b_D for the:**	
	Logic system	**Sensors or actuators**
120 or above	0.5%	1%
70 to 120	1%	2%
45 to 70	2%	5%
Less than 45	5%	10%

NOTE 1 The maximum levels of β_D shown in this table are lower than would normally be used, reflecting the use of the techniques specified elsewhere in this standard for the reduction in the probability of systematic failures as a whole, and of common cause failures as a result of this.

NOTE 2 Values of β_D lower than 0.5% for the logic system and 1% for the sensors would be difficult to justify.

D.7 Examples of the use of the methodology

In order to demonstrate the effect of using the methodology, some simple examples have been worked through in table D.5 for the **programmable electronics**.

For categories not relating to diagnostic tests nor diversity, half the maximum score for a category was used; this being the same for each of the examples. It should be noted that when table D.1 is used in practice, no such subdivision of scores is allowed.

Table D.5 — Example values for programmable electronics

Category		**Diverse system with good diagnostic testing**	**Diverse system with poor diagnostic testing**	**Redundancy system with good diagnostic testing**	**Redundancy system with poor diagnostic testing**
Separation/segregation	X	3.50	3.50	3.50	3.50
	Y	1.50	1.50	1.50	1.50
Diversity/redundancy	X	14.50	14.50	2.00	2.00
	Y	3.00	3.00	1.00	1.00
Complexity/design/.....	X	2.75	2.75	2.75	2.75
	Y	2.25	2.25	2.25	2.25
Assessment/analysis....	X	0.25	0.25	0.25	0.25
	Y	4.75	4.75	4.75	4.75
Procedures/human interface	X	3.50	3.50	3.50	3.50
	Y	3.00	3.00	3.00	3.00
Competence/training/...	X	1.25	1.25	1.25	1.25
	Y	3.75	3.75	3.75	3.75
Environmental control	X	2.75	2.75	2.75	2.75
	Y	2.25	2.25	2.25	2.25
Environmental test	X	5.00	5.00	5.00	5.00
	Y	5.00	5.00	5.00	5.00
Diagnostic coverage	Z	2.00	0.00	2.00	0.00
Total X		33.5	33.5	21	21
Total Y		25.5	25.5	23.5	23.5
Score S		59	59	44.5	44.5
β		2%	2%	2%	2%
Score S_D		126	59	86.5	44.5
β_D		0.5%	2%	1%	5%

D.8 References

The following references provide useful information relating to common cause failures.

a) Programmable electronic systems in safety-related applications, Part 2: General technical guidelines, Health and Safety Executive, HMSO, ISBN 0 11 883906 3, 1987.

b) Assigning a numerical value to the beta factor common-cause evaluation, Humphreys, R. A., Proc. Reliability '87.

c) UPM3.1: A pragmatic approach to dependent failures assessment for standard systems, AEA Technology, Report SRDA-R-13, ISBN 085 356 4337, 1996.

Annex E
(informative)

Example application of software safety integrity tables of part 3

E.1 Introduction

This annex gives two worked examples in the application of the software safety integrity tables specified in annex A of part 3.

The first example is a safety integrity level 2 programmable electronic safety-related system required for a process within a chemical plant. The programmable electronic safety-related system will utilise ladder logic for the application program, and is an illustration of limited variability language application programming.

The second example is a shut-down application based on a high level language, of safety integrity level 3.

E.2 Example for safety integrity level 2

The application consists of several reactor vessels linked by intermediate storage vessels which are filled with inert gas at certain points in the reaction cycle to suppress ignition and explosions. The programmable electronic safety-related system functions include: receiving inputs from the sensors; energising and interlocking the valves, pumps and actuators; detecting dangerous situations and activating the alarm; interfacing to a distributed control system, as required by the safety requirements specification.

Assumptions:

— the programmable electronic safety-related system controller is a PLC;

— the hazard and risk analysis has established that a programmable electronic safety-related system is required, and that safety integrity level 2 is required in this application (by the application of parts 1 and 2);

— although the controller operates in real time, only a relatively slow response is needed;

— there are interfaces to a human operator and to a distributed control system;

— the source code of the system software and the design of the programmable electronics of the PLC is not available for examination, but has been qualified to safety integrity level 2;

— the application code is required to run on only a single type of PLC;

— the application programming language is ladder logic, produced using the PLC supplier's development system;

— the whole of the software development was reviewed by a person independent of the software team;

— a person independent of the software team witnessed and approved the validation testing;

— modifications (if needed) require authorisation by a person independent of the software team.

NOTE For definition of independent person, see 3.8.10 of part 4.

The following tables show how the annex tables of part 3 may be interpreted for this application.

NOTE In the reference columns of the following tables, the technique/measure subclauses are of part 7 and the tables are of part 3.

Table E.1 — Software safety requirements specification (see 7.2 of part 3)

To specify the requirements for software safety in terms of (1) the requirements for software safety functions and (2) the requirements for software safety integrity, for each E/E/PE safety-related system necessary to implement the required safety functions, and to achieve the safety integrity level specified for each safety function allocated to that E/E/PE safety-related system.

Technique/Measure	Ref	SIL2	Interpretation in this application
1 Computer-aided specification tools	B.2.4	R	Development tools supplied by the PLC manufacturer.
2a Semi-formal methods	table B.7	R	Cause-effect diagrams, sequence diagrams, function blocks. Typically used for PLC application software requirements specification.
2b Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	R	Not used for limited variability programming.

a) The software safety requirements specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.

b) The table reflects additional requirements for specifying the software safety requirements clearly and precisely.

c) Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Table E.2 — Software design and development: software architecture design (see 7.4.3 of part 3)

To specify and justify techniques and measures necessary during the software safety lifecycle phases to satisfy the specification of requirements for software safety at the required safety integrity level .
To identify, evaluate and detail the significance of all hardware/software interactions.
To specify appropriate software architecture integration tests to ensure that the software architecture satisfies the specification of requirements for software safety at the required safety integrity level.

Technique/Measure	Ref	SIL2	Interpretation in this application
1 Fault detection and diagnosis	C.3.1	R	Checking of data range, watch-dog timer, I/O, communication. Raise an alarm if errors (see 3a).
2 Error detecting and correcting codes	C.3.2	R	Embedded with user options - careful selection required.
3a Failure assertion programming	C.3.3	R	Dedicate some PLC program rungs to test certain essential safety conditions (see 1).
3b Safety bag techniques	C.3.4	R	Check legal I/O combinations in an independent hardware safety monitor .
3c Diverse programming	C.3.5	R	Required by the application (see 3b).
3d Recovery block	C.3.6	R	Embedded with user options - careful selection required.
3e Backward recovery	C.3.7	R	Embedded with user options - careful selection required.
3f Forward recovery	C.3.8	R	Embedded with user options - careful selection required.
3g Re-try fault recovery mechanisms	C.3.9	R	Used as required by the application (see 2 and 3b).
3h Memorising executed cases	C.3.10	R	Not used for limited variability programming.
4 Graceful degradation	C.3.11	R	Not used for limited variability programming.
5 Artificial intelligence - fault correction	C.3.12	NR	Not used for limited variability programming.
6 Dynamic reconfiguration	C.3.13	NR	Not used for limited variability programming.
7a Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	C.2.1	HR	Rarely used for limited variability programming.
7b Semi-formal methods	table B.7	R	Rarely used for limited variability programming.
7c Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	R	Rarely used for limited variability programming.
8 Computer-aided specification tools	B.2.4	R	Development tools supplied by the PLC manufacturer.

a) Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

b) The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in part 2 of this standard.

NOTE It is impractical to implement some of the above techniques in limited variability programming.

Table E.3 — Software design and development:
support tools and programming language (see 7.4.4 of part 3)

To select a suitable set of integrated tools including languages, compilers, configuration management tools (and when applicable, automatic testing tools) to supply the relevant services over the whole lifetime of the E/E/PE safety-related system.			
Technique/Measure	**Ref**	**SIL2**	**Interpretation in this application**
1 Suitable programming language	C.4.6	HR	Usually ladder, and often the proprietary variety of the PLC supplier.
2 Strongly typed programming language	C.4.1	HR	IEC 61131-3 structured text.
3 Language subset	C.4.2	---	Beware of complex "macro" instructions, interrupts which alter PLC scan cycle, etc.
4a Certificated tools	C.4.3	HR	Available from some PLC manufacturers.
4b Tools: increased confidence from use	C.4.4	HR	PLC supplier's development kit; in-house tools developed over several projects.
5a Certificated translator	C.4.3	HR	Available from some PLC manufacturers.
5b Translator: increased confidence from use	C.4.4	HR	Not used for limited variability programming.
6 Library of trusted/verified software modules and components	C.4.5	HR	Function blocks, part programs.
Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.			

Table E.4 — Software design and development:
detailed design (see 7.4.5 and 7.4.6 of part 3)
(This includes software system design, software module design and coding)

To design and implement software which satisfies the specified requirements for software safety at the required safety integrity level, and which is modular, testable, and has the capacity for safe modification.			
Technique/Measure	**Ref**	**SIL2**	**Interpretation in this application**
1a Structured methods including for example, JSD, MASCOT, SADT and Yourdon	C.2.1	HR	Applied earlier at the system level.
1b Semi-formal methods	table B.7	HR	Cause-effect diagrams, sequence diagrams, function blocks. Typical for limited variability programming.
1c Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	R	Not used for limited variability programming.
2 Computer-aided design tools	B.3.5	R	Development tools supplied by the PLC manufacturer.
3 Defensive programming	C.2.5	R	Included in the system software.
4 Modular approach	table B.9	HR	Group the ladder rungs; consistent with the modularity of the equipment under control.
5 Design and coding standards	table B.1	HR	In-house conventions for documentation and maintainability
6 Structured programming	C.2.7	HR	Similar to modularity in this context.
7 Use of trusted/verified software modules and components (if available)	C.4.5	HR	Used.
Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.			

**Table E.5 — Software design and development:
software module testing and integration (see 7.4.7 and 7.4.8 of part 3)**

To test each software module, as specified during software design, and with the specified software integration tests, to show that each software module performs its intended function and does not perform unintended functions, and that all software modules and software components/subsystems interact correctly.

Technique/Measure	Ref	SIL2	Interpretation in this application
1 Probabilistic testing	C.5.1	R	Not used for limited variability programming.
2 Dynamic analysis and testing	B.6.5 table B.2	HR	Used.
3 Data recording and analysis	C.5.2	HR	Records of test cases and results.
4 Functional and black box testing	B.5.1 B.5.2 table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning.
5 Performance testing	C.5.20 table B.6	R	Not used for limited variability programming.
6 Interface testing	C.5.3	R	Included in functional and black box testing.

a) Software module and integration testing are verification activities (see table A.9 of part 3).

b) A numbered technique/measure shall be selected according to the safety integrity level.

c) Appropriate techniques/measures shall be selected according to the safety integrity level.

**Table E.6 — Programmable electronics integration (hardware and software)
(see 7.5 of part 3)**

To combine the software and hardware in the safety-related programmable electronics to ensure their compatibility and to meet the requirements of the intended safety integrity level.

Technique/Measure	Ref	SIL2	Interpretation in this application
1 Functional and black box testing	B.5.1 B.5.2 table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning.
2 Performance testing	C.5.20 table B.6	R	When the PLC system is assembled for factory acceptance test .

a) Programmable electronics integration is a verification activity (see table A.9 of part 3).

b) A numbered technique/measure shall be selected according to the safety integrity level.

c) Appropriate techniques/measures shall be selected according to the safety integrity level.

To ensure that the integrated system complies with the specified requirements for software safety (see 7.2) at the intended safety integrity level.			
Technique/Measure	**Ref**	**SIL2**	**Interpretation in this application**
1 Probabilistic testing	C.5.1	R	Not used for limited variability programming.
2 Simulation/modelling	table B.5	R	Not used for limited variability programming.
3 Functional and black box testing	B.5.1 B.5.2 table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning.
a) A numbered technique/measure shall be selected according to the safety integrity level. b) Appropriate techniques/measures shall be selected according to the safety integrity level.			

Table E.8 — Modification (see 7.8 of part 3)

To make corrections enhancements or adaptations to the validated software, ensuring that the required software safety integrity level is sustained.			
Technique/Measure	**Ref**	**SIL2**	**Interpretation in this application**
1 Impact analysis	C.5.23	HR	Consider how the effect of the proposed changes is limited by the modularity of overall system. Done as required.
2 Reverify changed software module	C.5.23	HR	Repeat earlier tests. Done as required.
3 Reverify affected software modules	C.5.23	HR	Repeat earlier tests. Done as required.
4 Revalidate complete system	C.5.23	R	May be necessary if impact analysis shows that the modification is extensive. Done as required.
5 Software configuration management	C.5.24	HR	Baselines, records of changes, impact on other system requirements. Done as required.
6 Data recording and analysis	C.5.2	HR	Records of test cases and results. Done as required.
a) A numbered technique/measure shall be selected according to the safety integrity level. b) Appropriate techniques/measures shall be selected according to the safety integrity level.			

To the extent required by the safety integrity level, to test and evaluate the outputs from a given software safety lifecycle phase to ensure correctness and consistency with respect to the outputs and standards provided as input to that phase.

Technique/Measure	Ref	SIL2	Interpretation in this application
1 Formal proof	C.5.13	R	Not used for limited variability programming.
2 Probabilistic testing	C.5.1	R	Replaced by operating experience of existing parts.
3 Static analysis	B.6.4 table B.8	HR	Clerical cross-referencing of usage of variables, conditions, etc.
4 Dynamic analysis and testing	B.6.5 table B.2	HR	Automatic test harness to facilitate regression testing.
5 Software complexity metrics	C.5.14	R	Not used for limited variability programming.
Software module testing and integration	See table A.5 of part 3		
Programmable electronics integration testing	See table A.6 of part 3		
Software system testing (validation)	See table A.7 of part 3		

a) For convenience all verification activities have been drawn together under this table. However this does not place additional requirements for the dynamic testing element of verification in table A.5 of part 3 (software module testing and integration) and table A.6 of part 3 (programmable electronics integration) which are verification activities in themselves. Nor does this clause require verification testing in addition to software validation (table A.7 of part 3), which in this standard is the phase of demonstrating conformance to the safety requirements specification (end-end verification).

b) Verification crosses the boundaries of parts 1, 2 and 3 of this standard. Therefore the first verification of the safety-related system is against the earlier system level specifications.

c) In the early phases of the software safety lifecycle verification is static, for example inspection, review, formal proof. When code is produced dynamic testing becomes possible. It is the combination of both types of information that is required for verification. For example code verification of a software module by static means includes such techniques as software inspections, walkthroughs, static analysis, formal proof. Code verification by dynamic means includes functional testing, white box testing, statistical testing. It is the combination of both types of evidence that provides assurance that each software module satisfies its associated specification.

d) A numbered technique/measure shall be selected according to the safety integrity level.

e) Appropriate techniques/measures shall be selected according to the safety integrity level.

To investigate and arrive at a judgement on the functional safety achieved by the programmable electronic safety-related system.			
Technique/Measure	**Ref**	**SIL2**	**Interpretation in this application**
1 Checklists	B.2.5	R	Used.
2 Decision/truth tables	C.6.1	R	Used to a limited degree.
3 Software complexity metrics	C.5.14	R	Not used for limited variability programming.
4 Failure analysis	table B.4	R	Cause-consequence diagrams at system level, but otherwise, failure analysis is not used for limited variability programming.
5 Common cause failure analysis of diverse software (if diverse software is actually used)	C.6.3	R	Not used for limited variability programming.
6 Reliability block diagram	C.6.5	R	Not used for limited variability programming.
Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.			

E.3 Example for safety integrity level 3

The software system is relatively large in terms of safety systems: more than 30,000 lines of source code are developed specifically for the system; in addition to that the usual intrinsic functions are used, at least 2 divers operating systems and pre-existing code from earlier projects (proven in use); in total more than 100,000 lines of source code were there, if it were available as such.

The whole hardware (including sensors and actuators) is a dual channel system with its outputs to the final elements connected as a logical AND.

Assumptions:

— although fast response is not required a maximum response time is guaranteed;

— there are interfaces to sensors, actuators and annunciators to human operators;

— the source code of the operating systems, graphic routines and commercial mathematical routines is not available;

— the system will be subject to later changes;

— the specifically developed software uses one of the common procedural languages;

— it is partially object oriented;

— all parts, whose source code is not available are implemented diverse, the software components being taken from different suppliers, their object code is generated by diverse translators;

— the software runs on several commercially available processors that fulfil the requirements of part 2;

— all requirements for control and avoidance of hardware faults according to part 2 are fulfilled by the embedded system; and

— the software development was assessed by an independent organisation.

NOTE For definition of independent organisation, see 3.8.10 of part 4.

The following tables show how the annex tables of part 3 may be interpreted for this application.

NOTE In the reference columns of the following tables, the technique/measure subclauses are of part 7 and the tables are of part 3.

Table E.11 — Software safety requirements specification (see 7.2 of part 3)

To specify the requirements for software safety in terms of (1) the requirements for software safety functions and (2) the requirements for software safety integrity, for each E/E/PE safety-related system necessary to implement the required safety functions, and to achieve the safety integrity level specified for each safety function allocated to that E/E/PE safety-related system.

Technique/Measure	Ref	SIL3	Interpretation in this application
1 Computer-aided specification tools	B.2.4	HR	Tools supporting the chosen methods.
2a Semi-formal methods	table B.7	HR	Block diagrams, sequence diagrams, state transition diagrams.
2b Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	R	Only exceptionally.

a) The software safety requirements specification will always require a description of the problem in natural language and any necessary mathematical notation that reflects the application.

b) The table reflects additional requirements for specifying the software safety requirements clearly and precisely.

c) Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Table E.12 — Software design and development: software architecture design (see 7.4.3 of part 3)

To specify and justify techniques and measures necessary during the software safety lifecycle phases to satisfy the specification of requirements for software safety at the required safety integrity level . To identify, evaluate and detail the significance of all hardware/software interactions. To specify appropriate software architecture integration tests to ensure that the software architecture satisfies the specification of requirements for software safety at the required safety integrity level.			
Technique/Measure	**Ref**	**SIL3**	**Interpretation in this application**
1 Fault detection and diagnosis	C.3.1	HR	used as far as dealing with sensor, actuator and data transmission failures and not covered by the measures within the embedded system according to the requirements of part 2.
2 Error detecting and correcting codes	C.3.2	R	Only for external data transmissions
3a Failure assertion programming	C.3.3	R	Results of the application functions are checked for validity.
3b Safety bag techniques	C.3.4	R	Used for some safety related functions where 3a and 3c are not used.
3c Diverse programming	C.3.5	R	Used for some functions where source code is not available.
3d Recovery block	C.3.6	R	Not used.
3e Backward recovery	C.3.7	R	Not used.
3f Forward recovery	C.3.8	R	Not used.
3g Re-try fault recovery mechanisms	C.3.9	R	Not used .
3h Memorising executed cases	C.3.10	R	Not used (measures 3a, b and c are sufficient).
4 Graceful degradation	C.3.11	HR	Yes, because of the nature of the technical process
5 Artificial intelligence - fault correction	C.3.12	NR	not used
6 Dynamic reconfiguration	C.3.13	NR	not used
7a Structured methods including for example, JSD, MASCOT, SADT and Yourdon.	C.2.1	HR	Needed because of the size of the system.
7b Semi-formal methods	table B.7	HR	Block diagrams, sequence diagrams, state transition diagrams.
7c Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	R	Not used.
8 Computer-aided specification tools	B.2.4	HR	Tools supporting the chosen method.
a) Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied. b) The measures in this table concerning fault tolerance (control of failures) should be considered with the requirements for architecture and control of failures for the hardware of the programmable electronics in part 2 of this standard.			

Table E.13 — Software design and development: support tools and programming language (see 7.4.4 of part 3)

To select a suitable set of integrated tools including languages, compilers, configuration management tools (and when applicable, automatic testing tools) to supply the relevant services over the whole lifetime of the E/E/PE safety-related system.

Technique/Measure	Ref	SIL3	Interpretation in this application
1 Suitable programming language	C.4.6	HR	Full variability high level language selected.
2 Strongly typed programming language	C.4.1	HR	Used.
3 Language subset	C.4.2	HR	Defined subset for the selected language.
4a Certificated tools	C.4.3	HR	Not available.
4b Tools: increased confidence from use	C.4.4	HR	Available, and used.
5a Certificated translator	C.4.3	HR	Not available.
5b Translator: increased confidence from use	C.4.4	HR	Available, and used.
6 Library of trusted/verified software modules and components	C.4.5	HR	Available, and used.

Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Table E.14 — Software design and development: detailed design (see 7.4.5 and 7.4.6 of part 3)

(This includes software system design, software module design and coding)

To design and implement software which satisfies the specified requirements for software safety at the required safety integrity level, and which is modular, testable, and has the capacity for safe modification.

	Technique/Measure	Ref	SIL3	Interpretation in this application
1a	Structured methods including for example, JSD, MASCOT, SADT and Yourdon	C.2.1	HR	Widely used. In particular, SADT and JSD.
1b	Semi-formal methods	table B.7	HR	Finite state machines/state transition diagrams, block diagrams, sequence diagrams.
1c	Formal methods including for example, CCS, CSP, HOL, LOTOS, OBJ, temporal logic, VDM and Z	C.2.4	R	Only exceptionally, for some very basic components only.
2	Computer-aided design tools	B.3.5	HR	Used for the selected methods.
3	Defensive programming	C.2.5	HR	All measures except those which are automatically inserted by the compiler are explicitly used in application software where they are effective.
4	Modular approach	table B.9	HR	Software module size limit, information hiding/encapsulation, one entry/one exit point in subroutines and functions, fully defined interface, ...
5	Design and coding standards	table B.1	HR	Use of coding standard, no dynamic objects, no dynamic variables, limited use of interrupts, limited use of pointers, limited use of recursion, no unconditional jumps, ...
6	Structured programming	C.2.7	HR	Used.
7	Use of trusted/verified software modules and components (if available)	C.4.5	HR	Available, and used.

Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.

Table E.15 — Software design and development: software module testing and integration (see 7.4.7 and 7.4.8 of part 3)

To test each software module, as specified during software design, and with the specified software integration tests, to show that each software module performs its intended function and does not perform unintended functions, and that all software modules and software components/subsystems interact correctly.

Technique/Measure	Ref	SIL3	Interpretation in this application
1 Probabilistic testing	C.5.1	R	Used for software modules where no source code available and the definition of boundary values and equivalence classes for test data is difficult.
2 Dynamic analysis and testing	B.6.5 table B.2	HR	Used for software modules where source code available. Test cases from boundary value analysis, performance modelling, equivalent classes and input partitioning, structure-based testing
3 Data recording and analysis	C.5.2	HR	Records of test cases and results.
4 Functional and black box testing	B.5.1 B.5.2 table B.3	HR	Used for software module testing where no source code available and for integration testing. Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, prototyping, boundary value analysis, equivalence classes and input partitioning.
5 Performance testing	C.5.20 table B.6	HR	Used during integration testing on the target hardware.
6 Interface testing	C.5.3	HR	Not used.

a) Software module and integration testing are verification activities (see table A.9 of part 3).

b) A numbered technique/measure shall be selected according to the safety integrity level.

c) Appropriate techniques/measures shall be selected according to the safety integrity level.

Table E.16 — Programmable electronics integration (hardware and software) (see 7.5 of part 3)

To combine the software and hardware in the safety-related programmable electronics to ensure their compatibility and to meet the requirements of the intended safety integrity level.

Technique/Measure	Ref	SIL3	Interpretation in this application
1 Functional and black box testing	B.5.1 B.5.2 table B.3	HR	Used as additional tests to software integration testing (see table E.15). Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, prototyping, boundary value analysis, equivalence classes and input partitioning.
2 Performance testing	C.5.20 table B.6	HR	Extensively used.

a) Programmable electronics integration is a verification activity (see table A.9 of part 3).

b) A numbered technique/measure shall be selected according to the safety integrity level.

c) Appropriate techniques/measures shall be selected according to the safety integrity level.

Table E.17 — Software safety validation (see 7.7 of part 3)

To ensure that the integrated system complies with the specified requirements for software safety (see 7.2) at the intended safety integrity level.			
Technique/Measure	**Ref**	**SIL3**	**Interpretation in this application**
1 Probabilistic testing	C.5.1	R	Not used for validation.
2 Simulation/modelling	table B.5	HR	Finite state machines, performance modelling, prototyping and animation
3 Functional and black box testing	B.5.1 B.5.2 table B.3	HR	Input data is selected to exercise all specified functional cases, including error handling. Test cases from cause consequence diagrams, boundary value analysis, and input partitioning.
a) A numbered technique/measure shall be selected according to the safety integrity level. b) Appropriate techniques/measures shall be selected according to the safety integrity level.			

Table E.18 — Modification (see 7.8 of part 3)

To make corrections enhancements or adaptations to the validated software, ensuring that the required software safety integrity level is sustained.			
Technique/Measure	**Ref**	**SIL3**	**Interpretation in this application**
1 Impact analysis	C.5.23	HR	Used.
2 Reverify changed software module	C.5.23	HR	Used.
3 Reverify affected software modules	C.5.23	HR	Used.
4 Revalidate complete system	C.5.23	HR	**Depends on the result of the impact analysis.**
5 Software configuration management	C.5.24	HR	Used.
6 Data recording and analysis	C.5.2	HR	Used.
a) A numbered technique/measure shall be selected according to the safety integrity level. b) Appropriate techniques/measures shall be selected according to the safety integrity level.			

To the extent required by the safety integrity level, to test and evaluate the outputs from a given software safety lifecycle phase to ensure correctness and consistency with respect to the outputs and standards provided as input to that phase.

Technique/Measure	Ref	SIL3	Interpretation in this application
1 Formal proof	C.5.13	R	Only exceptionally, for some very basic classes only.
2 Probabilistic testing	C.5.1	R	**Included in table E.15**
3 Static analysis	B.6.4 table B.8	HR	For all newly developed code. Boundary value analysis, checklists, control flow analysis, data flow analysis, Fagan inspections, design reviews.
4 Dynamic analysis and testing	B.6.5 table B.2	HR	**Included in table E.15**
5 Software complexity metrics	C.5.14	R	Used only marginally.
Software module testing and integration	See table A.5 of part 3		
Programmable electronics integration testing	See table A.6 of part 3		
Software system testing (validation)	See table A.7 of part 3		

a) For convenience all verification activities have been drawn together under this table. However this does not place additional requirements for the dynamic testing element of verification in table A.5 of part 3 (software module testing and integration) and table A.6 of part 3 (programmable electronics integration) which are verification activities in themselves. Nor does this clause require verification testing in addition to software validation (table A.7 of part 3), which in this standard is the phase of demonstrating conformance to the safety requirements specification (end-end verification).

b) Verification crosses the boundaries of parts 1, 2 and 3 of this standard. Therefore the first verification of the safety-related system is against the earlier system level specifications.

c) In the early phases of the software safety lifecycle verification is static, for example inspection, review, formal proof. When code is produced dynamic testing becomes possible. It is the combination of both types of information that is required for verification. For example code verification of a software module by static means includes such techniques as software inspections, walkthroughs, static analysis, formal proof. Code verification by dynamic means includes functional testing, white box testing, statistical testing. It is the combination of both types of evidence that provides assurance that each software module satisfies its associated specification.

d) A numbered technique/measure shall be selected according to the safety integrity level.

e) Appropriate techniques/measures shall be selected according to the safety integrity level.

To investigate and arrive at a judgement on the functional safety achieved by the programmable electronic safety-related system.			
Assessment/Technique	**Ref**	**SIL3**	**Interpretation in this application**
1 Checklists	B.2.5	R	Used.
2 Decision/truth tables	C.6.1	R	Used, to a limited degree.
3 Software complexity metrics	C.5.14	R	Used only marginally.
4 Failure analysis	table B.4	HR	Fault tree analysis is extensively used, and cause consequence diagrams are used to a limited degree.
5 Common cause failure analysis of diverse software (if diverse software is actually used)	C.6.3	HR	Used.
6 Reliability block diagram	C.6.5	R	Used.
Appropriate techniques/measures shall be selected according to the safety integrity level. Alternate or equivalent techniques/measures are indicated by a letter following the number. Only one of the alternate or equivalent techniques/measures has to be satisfied.			