Guideline Document & Work Instruction for testing purposes on how to implement the Annex R of IEC 60335-1 and Annex H of IEC 60730-1
1 Introduction

The use of electronic controls in appliances covered by IEC 60335 series promoted important modifications in the standard to cover the safety aspects that the differentiations between electromechanical and electronic controls in appliances require.

An obvious difference is in the operation and construction of an electronic control. The electromechanical components have only physical elements which, for example, are movable to open or close a circuit, or have a static response to electric signals which change the circuit operation. Its reliability can be evaluated by endurance tests, normally according to specific component standards.

In an electronic component there is a hardware construction with discrete electronic components and usually has a software to control the operation of the component.

Even if the construction and principles of operation are different, the component standards can include the way to evaluate the control operation reliability in the same way that does that for electromechanical components. But this is possible only when the electronic control is clearly classified either for a specific discrete function (timer, thermostat, switch, etc) and if it has a functional or a protection purposes (i.e thermostat or thermal cut-out).

The main problem for considering the component standard evaluation enough for the safety assessment of the appliance is related with the fact that the appliance control usually integrate many functions and protections needed for the appliance and it is not always easy to separate in the control the discrete components that perform the different tasks that are the base of the safety analysis of the appliance, either in normal operation (Cl 11), or abnormal conditions (Cl 19).

By other hand, depending on the construction and functions of the appliance, the same control may provide a normal operation function in one of them and protection functions in abnormal conditions in other, or even both of the above functions in a third appliance construction.

For that reason it is necessary to perform in each appliance an analysis of the electronic control and its functionality in the conditions of the test requirement of IEC 60335. This analysis is the previous step to determine the requirements to be assigned to the control.

The analysis consequently will determine the test to be performed in the whole appliance according to clause 19, considering the different conditions applicable to functional controls and protection controls.

By other hand, the previous analysis may reduce the tests to be performed because sometimes the control, or a part of it, provides the same appliance protection in more than one of the test of Cl 19. In other cases the reduction of the tests is due to the consideration that some of them are only applicable to PEC (protective electronic circuits) and not to functional electronic circuits. Consequently for that purpose it is necessary to perform the analysis before starting the tests.

Finally, in the same way that an analysis is necessary for the identification of the parts of the control that performs the safety functions, the test report shall include this information for certification traceability purposes. Consequently it is necessary to specify the tables, list of documents and the list of the relevant control parts, to be added to TRF.
2 Objectives of this Guide

The main objectives of this Guide are the following:

Procedure for identification of clauses of IEC 60335 to be considered in electronic controls using software for safety purposes.

General procedures to detect and identify the software that require special measures to control fault/error conditions specified in table R of IEC 60335.

Documents and information to be provided for the relevant audit of the SW measures and the way in which this documentation is identified in the TRF.

The way to identify the SW and the way to reflect it in the TRF including the versions and modifications of it.

The way in which the process of audit and evaluation of the SW allows to fulfils the Table R1 or R2.

Addendum to the TRF’s of IEC 60335 series to include the minimum information required to identify the safety requirements applicable to the software and hardware.

Examples for the application of this guideline.

3 Procedure for identification of clauses of IEC 60335 to be considered

In IEC 60335-1 ed 4.2 software measures are only required for PEC’s, which according the definition in 3.9.3 are those electronic controls that prevent hazardous situation under abnormal operation conditions.

Considering that Cl 19 represents the test specifications that covers the abnormal operation conditions, only during the test of this clause the electronic circuit need to be analysed, but it shall be noted that even if there are specific sub-clauses (19.11.1 and 19.11.2) for testing hardware failures on electronic components, all 19 sub clauses applies to electronic circuits when operate during the corresponding tests.

It is necessary to consider the difference between the test in 19.11.2 and the other 19 sub-clauses. When a 19 sub-clause (including 19.10X in Part’s 2) requires to fail a control that operate in normal operation conditions, it shall be considered that the control shall be short-circuited or rendered inoperative as a whole (note 5 of 19.1). When doing an analysis of an electronic control, it shall be noted that it is different to short-circuit the control than to short-circuit electronic components of the control which is covered by 19.11.2.

The reason to require failures in the normal operating controls as a whole is because there are no separate requirements for functional electronic controls in the standard to achieve the reliability needed for a safe use. Therefore the electronic component failures covered in 19.11.2 do not cover all possible failure conditions that may be expected and covered in the other Cl 19 sub-clauses. In the other side it is important to note that the specific requirements for PEC’s are Cl 19.11.3 (electronic component failure), Cl 19.11.4 (EMP) and Cl 22.46 (SW protection) that gives higher confidence level than the required for functional controls.

At this point it should be noted that not all of the electronic circuits or components which actuate during clause 19 test has to meet the requirements for PEC’s. Only when the electronic control is necessary for the compliance with the relevant compliance criteria shall be considered as PEC and subjected to the test for it.
4 General procedure to detect and identify the software that requires special measures

In IEC 60335-1 it is recognised that an electronic control may have functional parts and PEC’s (note of 3.9.3) and then a procedure to detect which parts are one or the other is basic for the exercise.

In order to have a systematic process for it, each requirement of Cl 19 shall be evaluated analysing the behaviour of the appliance when the relevant failure in the control is simulated. At this stage when a functional electronic control (or the functional part of the electronic control) is involved, the consideration of all the possible feasible inputs or outputs of this part of the control that may affect the compliance criteria is necessary for the analysis.

It shall be noted that, in general, the test of Cl 19 has to be performed in all normal operation conditions of the appliance, but do not include the “off” or “stand-by” conditions, even if these conditions are performed by the same electronic control which actuate in normal operation (see DSH-722). Only 19.11.4 test shall be done in “off” or “stand-by” conditions unless otherwise is required in a specific Part 2 (i.e 19.11.2 of Part 2-6 for hobs). By other hand this test has to be performed when an electronic disconnection is provided, regardless if there is an electromechanical switch in series (see DSH-713).

When the compliance with each requirement relies on an electronic circuit, or on a part of it, it is then considered a PEC for this test and then subjected to the subsequent test and evaluations.

The hardware failures in a PEC are covered by 19.11.3 which requires to repeat the 19.X tests with one electronic component failure in the PEC’s as in 19.11.2. Taking into account that PEC’s operate after an abnormal operation conditions in an appliance, it shall be expected that the failures in the PEC can be produced before its actuation. Consequently the order in which the test of 19.11.3 shall be done is to start with the 19.11.2 test in the PEC and after it to repeat the 19.X test which is protected by this PEC (see DSH-721). Additionally it shall be remarked that the compliance criteria of 19.11.3 is 19.13, regardless the particular compliance criteria of the correspondent 19.X test.

The above considerations do not imply that PEC’s itself shall be necessarily a separate electronic control, or a specific part of it, because it is not forbidden the use of software as a PEC. When the failure of the functional control is monitored only by separate software which detects the failure and leave the appliance in a safe mode, then the software itself becomes a PEC.

Additionally, software may be used for monitoring the possible failures of PEC’s. In this aspect and according to 19.11.3 testing specification, a PEC electronic component failure shall be done first and after the relevant 19.X test protected by this PEC is performed, but if there is a software that detect the PEC failure and leave the appliance inoperative after that, then it would be not possible after to apply the relevant 19.X test, unless both tests (19.11.3 and 19.X) are done simultaneously, conditions which are not likely to be produced at the same time in practise (see DSH-721). There are no special requirements for protection measures of the software that protect PEC failures, but the analysis of the different failures in Cl 19.11.2 apply to PEC’s as part of electronic circuit and sometimes the software protecting PEC’s shall be software B.

With respect to EMP it shall be considered that test of 19.11.4 has to be performed after the actuation of the PEC in each 19.X test, but not after PEC failure in 19.11.3.

With all of these considerations, it is needed a clear information about the role of the control with respect to the appliance standard compliance, that has to be included in the test reports.
5 Addendum to the TRF’s of IEC 60335 series to include the minimum information required to identify the safety requirements applicable to the software and hardware

In order to specify the conditions in which Cl 19 is fulfilled in an appliance tested according IEC 60335 and in which the electronic control is necessary for the compliance, the following tables shall be included in the tables annexed to IEC 60335 TRF’s. When more than one operating condition may apply in the same sub-clause the relevant line shall be duplicated.

**CLAUSE 19: ABNORMAL OPERATION CONDITIONS**

<table>
<thead>
<tr>
<th>Operational Characteristics</th>
<th>YES/NO</th>
<th>Operational conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are there electronic circuits to control the appliance operation?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Are there “off” or “stand-by” position?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The unintended operation of the appliance results in dangerous malfunction?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Subclause Description</th>
<th>Operating Conditions description</th>
<th>Test results description</th>
<th>PEC Description</th>
<th>EMP 19.11.4</th>
<th>Software type required 22.46</th>
<th>19.11.3 PEC</th>
<th>Final result</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.2</td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.6</td>
<td></td>
<td>N/A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.11.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.11.4.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.10X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
6 Examples for the software detection and identification

Two examples are included to clarify the application of the guideline. For simplification and as example only, some of the sub-clauses were applied and only part 1 requirements was taken in consideration.

6.1 Case 1: Heating appliance with double control (duplicity)

The appliance has a heating element that if operated uncontrolled may create an overheating with fire risk. The heating element is controlled by a microprocessor with two sensors (PT100 y NTC) and three electronic switches (T1, T2 and T3) as output to drive the heating elements. The NTC+Micro+T1 construction actuate as functional control that operate in normal operation. The PT100+Micro+T2/T3 operate as redundant system with independent software (separate from the functional software in terms of software structure and data process) as temperature limiter, in such a way that cut the operation of the heating element when temperature gets a maximum value predefined. To reduce the extension of the example only sub-clauses 19.4 and 19.11.2 will be applied.

![Diagram of heating appliance with double control](image)

**CLAUSE 19: ABNORMAL OPERATION CONDITIONS**

<table>
<thead>
<tr>
<th>Operational Characteristics</th>
<th>YES/NO</th>
<th>Operational conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are there electronic circuits to control the appliance operation?</td>
<td>YES</td>
<td>Normal operation with heating element controlled by thermostat</td>
</tr>
<tr>
<td>Are there “off” or “stand-by” position?</td>
<td>NO</td>
<td></td>
</tr>
<tr>
<td>The unintended operation of the appliance results in dangerous malfunction?</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>Subclause</td>
<td>Operating Conditions description</td>
<td>Test results description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------------------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>19.2</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.3</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.4</td>
<td>Short-circuit of thermostat. (NTC in fixed value (1))</td>
<td>Control actuates at $\Delta t$ °C. No fires or deformation can be produced</td>
</tr>
<tr>
<td>19.5</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.6</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.7</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.8</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.9</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.10</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.11.2</td>
<td>Short-circuit of T1.</td>
<td>Control actuates at $\Delta t$ °C. No fires or deformation is produced</td>
</tr>
<tr>
<td>19.11.2</td>
<td>Short-circuit of T2/T3.</td>
<td>Control actuates as normal operation</td>
</tr>
<tr>
<td>19.11.2</td>
<td>Short/open circuit of PT100(3).</td>
<td>SW for the reading of PT100 detects the short/open circuit and disconnects the heating element</td>
</tr>
<tr>
<td>19.11.4.8</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.10X</td>
<td>Not checked</td>
<td>--</td>
</tr>
</tbody>
</table>
(1) – The NTC in fixed value is the better way that was found in this case to simulate that the thermostat is rendered inoperative, leaving the appliance heating uncontrolled (see DSH 725A)

(2) – Other possible hardware failures in the micro according 19.11.2 are considered covered by the SC of T2 or PT100 because its consequences are the same.

(3) – The short/open circuit of NTC or of the other possible hardware failures in the micro according 19.11.2 are already covered by the test above.

6.2 Case 2: Heating appliance with simple control and monitoring software

The appliance has a heating element that if operated uncontrolled may create an overheating with fire risk. The heating element is controlled by a microprocessor with one sensor (PT 100) and two electronic switches (T1 and T2) as output to drive the heating elements. The PT100+Micro+T1 construction actuate as functional control that operate in normal operation. When the functional control fails (PT100 in fixed value, functional software failure, T1 short-circuit, EMP failures, etc), there is a monitoring software which compare the input/output values with the expected program values along the time, in order to detect failures in the functional control and disconnect the heating element through T2. For the short-circuit of T1 or T2, there is special T monitoring software which runs a periodical test in combination with a sensor (T monitoring in the figure) that can detect whether T1 or T2 is short-circuit and stops the operation of the appliance.

To reduce the extension of the example only sub-clauses 19.4 and 19.11.2 will be applied.

CLAUSE 19: ABNORMAL OPERATION CONDITIONS

<table>
<thead>
<tr>
<th>Operational Characteristics</th>
<th>YES/NO</th>
<th>Operational conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Are there electronic circuits to control the appliance operation?</td>
<td>YES</td>
<td>Normal operation with heating element controlled by thermostat</td>
</tr>
<tr>
<td>Are there “off” or “stand-by” position?</td>
<td>NO</td>
<td>----</td>
</tr>
<tr>
<td>The unintended operation of the appliance results in dangerous malfunction?</td>
<td>N/A</td>
<td>----</td>
</tr>
<tr>
<td>Subclause</td>
<td>Operating Conditions description</td>
<td>Test results description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>19.2</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.3</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.4</td>
<td>Short-circuit of thermostat. (PT100 in fixed value (^1))</td>
<td>Control detects an incorrect (\Delta t) °C/time. No fires or deformation is produced</td>
</tr>
<tr>
<td>19.5</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.6</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.7</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.8</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.9</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.10</td>
<td>N/A</td>
<td>--</td>
</tr>
<tr>
<td>19.11.2</td>
<td>Short-circuit of T1/T2.</td>
<td>T monitoring system stops the operation of the appliance</td>
</tr>
<tr>
<td>19.11.2</td>
<td>Short/open circuit of PT100 (^2).</td>
<td>SW for the reading of PT100 detects the short/open circuit and disconnects the heating element</td>
</tr>
<tr>
<td>19.11.4.8</td>
<td>Not checked</td>
<td>--</td>
</tr>
<tr>
<td>19.10X</td>
<td>Not checked</td>
<td>--</td>
</tr>
</tbody>
</table>

\(^1\) – The NTC in fixed value is the better way that was found in this case to simulate that the thermostat is rendered inoperative, leaving the appliance heating uncontrolled

\(^2\) – Other possible hardware failures in the micro according 19.11.2 are considered covered by the SC of T2 because its consequences are the same.
7 Process of Audit Trail, Items & Evaluation for Data Exchange Among NCB’s and CBTL’s

The general process shall include the following phases:

Project definition
  o Concept of operation
  o Requirements and architecture
  o Detailed design

Implementation

Project test and integration
  o Integration test and verification
  o System verification and validation
  o Operation and maintenance

For each phase the following items could be documented and used to define the information that needs to be gathered by NCB A and conveyed to NCB B, with documentation references from the original certification that are traceable throughout the lifecycle and demonstrate the objective evidence gather during the audit of the manufacturer’s software and processes.

7.1 Concept of Operation / Safety Requirements specifications

The specification of the software safety requirements shall include:

(IEC 60335-1 ed5 R.3.2)

a description of each safety related function to be implemented, including its response time(s)
  • functions related to the application
    including their related software faults required to be controlled;
  • functions related to the detection,
    annunciation and management of software or hardware faults;
  • a description of interfaces between software and hardware;
  • a description of interfaces between any safety and non-safety related functions;
  • a description of any compiler used to generate the object code from the source code, including details of any compiler switch settings used such as library function options, memory model, optimization, SRAM details, clock rate and chip details;
  • a description of any linker used to link the object code to executable library routines.

Technique / Measure:

• Semi-formal methods
• Logical/functional block diagrams
• Sequence diagrams
• Finite state machines/state transition diagrams
• Decision/truth tables
7.2 Hard and Software architecture

- interactions between hardware and software
- partitioning into modules and their allocation to the specified safety functions;
- hierarchy and call structure of the modules (control flow);
- interrupt handling;
- data flow and restrictions on data access;
- architecture and storage of data;
- time-based dependencies of sequences and data.

**Technique / Measure:**
- Data flow diagrams

7.3 Module design and coding

Software module design and coding shall be implemented in a way that is traceable to the software architecture and requirements.

The module design shall specify:
- function(s),
- interfaces to other modules,
- data.

**Defensive programming** (IEC 61508-7, Sub-clause C.2.5) is recommended (e.g. range checks, check for division by 0, plausibility checks).

**Technique / Measure:**
- Limited size of software modules IEC 61508-7, C.2.9
- Information hiding / encapsulation IEC 61508-7, C.2.8
- One entry / one exit point in subroutines and functions IEC 61508-7, C.2.9
- Fully defined interface IEC 61508-7, C.2.9

7.4 Software code shall be structured

- keep the number of possible paths through a software module small, and the relation between the input and output parameters as simple as possible;
- avoid complicated branching and, in particular, avoid unconditional jumps (GOTO) in higher level languages;
- where possible, relate loop constraints and branching to input parameters;
- avoid using complex calculations as the basis of branching and loop decisions.

**Technique / Measure:**
- Use of coding standard (see NOTE) IEC 61508-7, C.2.6.2
- No use of dynamic objects and variables (see NOTE) IEC 61508-7, C.2.6.3
- Limited use of interrupts IEC 61508-7, C.2.6.5
- Limited use of pointers IEC 61508-7, C.2.6.6
- Limited use of recursion IEC 61508-7, C.2.6.7
- No unconditional jumps in programs in higher level languages IEC 61508-7, C.2.6.2
Note: Dynamic objects and/or variables are allowed if a compiler is used which ensures that sufficient memory for all dynamic objects and/or variables will be allocated before runtime, or which inserts runtime checks for the correct online allocation of memory.

7.5 Audit Trail and Evidence of Compliance for NCB Information Exchange

In following the above requirements, an audit trail of both actual data and document references may be provided as the basis for exchange of information between the original certifier (NCB A) and the secondary certifier (NCB B) accepting data via the IECEE SB Scheme:

7.5.1 The following details should be captured in the original certification report (per the above mentioned requirements of IEC 60335):

- a description of each safety related function to be implemented, including its response time(s)
  - functions related to the application
    - including their related software faults required to be controlled;
  - functions related to the detection,
    - annunciation and management of software or hardware faults;
  - a description of interfaces between software and hardware;
  - a description of interfaces between any safety and non-safety related functions;

7.5.2 The following items should be captured briefly (make, model, version) in the report and details may be captured via manufacturer’s documentation reference(s):

- a description of any compiler used to generate the object code from the source code, including details of any compiler switch settings used such as library function options, memory model, optimization, SRAM details, clock rate and chip details;
- a description of any linker used to link the object code to executable library routines.

7.5.3 The following items should be captured in the original certification report (per above requirements) supported by references to the manufacturer’s documentation for specific details:

- Semi-formal methods
- Logical/functional block diagrams
- Sequence diagrams
- Finite state machines/state transition diagrams
- Decision/truth tables

7.5.4 The following should be documented in the report and the specific Technique / Measure referenced in the report via reference to manufacturer’s documentation of data flows:

- interactions between hardware and software
- partitioning into modules and their allocation to the specified safety functions;
- hierarchy and call structure of the modules (control flow);
- interrupt handling;
- data flow and restrictions on data access;
- architecture and storage of data;
- time-based dependencies of sequences and data.

7.5.5 The following may be demonstrated by providing excerpts of safety related code modules during the information exchange between NCB’s or via NCB B’s contact with the manufacturer (comparable to the situation where NCB B requests a product sample from the manufacturer):

- interactions between hardware and software
- partitioning into modules and their allocation to the specified safety functions;
7.5.6 The elements of item 1 (above) should be traceable to the test plan for the product / software in that specific test cases, expected results, and actual results of testing should be identified and reflected in the report (via manufacturer’s test report reference) establishing that all safety-related functions have been tested under normal and abnormal conditions for relevant failure mode and stress conditions (e.g. single bit faults, DC fault, etc...)

7.5.7 The items 1-6 above should establish a level of confidence from documented audit trail evidence that all safety relevant portions of the code have been evaluated by NCB A for the following attributes:

- keep the number of possible paths through a software module small, and the relation between the input and output parameters as simple as possible;
- avoid complicated branching and, in particular, avoid unconditional jumps (GOTO) in higher level languages;
- where possible, relate loop constraints and branching to input parameters;
- avoid using complex calculations as the basis of branching and loop decisions.

Technique / Measure:

- Use of coding standard (see Note) IEC 61508-7, C.2.6.2
- No use of dynamic objects and variables (see Note) IEC 61508-7, C.2.6.3
- Limited use of interrupts IEC 61508-7, C.2.6.5
- Limited use of pointers IEC 61508-7, C.2.6.6
- Limited use of recursion IEC 61508-7, C.2.6.7
- No unconditional jumps in programs in higher level languages IEC 61508-7, C.2.6.2

Note: Dynamic objects and/or variables are allowed if a compiler is used which ensures that sufficient memory for all dynamic objects and/or variables will be allocated before runtime, or which inserts runtime checks for the correct online allocation of memory.

8 Process to identify the SW and the way to reflect it in the TRF

8.1 Software designation

Definitions for the purpose of this chapter of the guide:

Software designation: Name given by the programmer to the software included in a programmable system which allows its traceability through the documentation required by the applicable standards.

In order to obtain that traceability the Software designation shall have the following properties:
- The designation must be unique, in such a way that modifications of the software imply modification of the designation.
- In order to track software modifications, a document with a dated historic register (see notes below) must be kept updated by the manufacturer.
Notes:

- A product may include several programmable systems which can include different software. Each shall have its own designation. For instance, an induction hob may have a user interface with a microcontroller and an electronic power control with another microcontroller. Each microcontroller shall have its own designation.

- A possible format of a dated historic register referred above may be a document including a table with a row for each modified version of the software. The table should include at least the essence of the following columns: “software designation”, “date of issue” and “summary of the changes from previous versions”.

- Different products of a family covered in a test report may include software with small functional differences. The uniqueness of the designation, imply that each variation must have a different designation. The designation system used by the manufacturer should cover the variations and the modifications in such a way that the evolution of the software may be tracked in the historic register.

Example: The following table could be a simple example of an historic register:

<table>
<thead>
<tr>
<th>SW designation</th>
<th>Issue date</th>
<th>Software variations</th>
<th>Summary of changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Washing software WVx</td>
<td>2010-07-12</td>
<td>“x” in software designation is one digit to refer to software variations giving different information in the user display.</td>
<td>Initial version</td>
</tr>
<tr>
<td>Washing software WVx-1</td>
<td>2010-10-11</td>
<td>All software variations include this modification</td>
<td>Updating frequency of the user display in stand-by mode is reduced.</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Information about the designation system used must be included in the software documentation.

8.2 Reference to the software in the test reports

In the test reports the software shall be referenced by its designation.

As the software may be modified, the software designation in the test report could become obsolete with respect to the software included in a future product. This situation may be handled using the following rules:

If the modification is deemed by the manufacturer as non-safety related, the only requirement is to keep updated the historic register mentioned above.

If the manufacturer considers necessary the confirmation of the NCB-A of the non-safety relevance of the modifications, the updated historic register mentioned before is also enough with a reference in it to the document of the test lab in which the non-safety character is confirmed. This document must be kept with the software documentation.

If the modifications of the software are considered as safety relevant a new software assessment is considered necessary and modifications of the test report and certificate must be issued.

Example: In the above example of an historic register, an additional column could include the information above the safety or non-safety character of the software modifications.

NCB-B can ask to the manufacturer for the historic register mentioned above to check that the product to be certified is covered by the software version initially certified, taking into account the rules above.
8.3 Evidence of identity of the software in a product

In a product, the software cannot be identified in the same way that can be done with a physical component. To address this situation it is advisable (to be changed for mandatory for certification?) that the product have provisions to give information about the software included in it. The information provided should allow tracking the software designation in its documentation and in the test report.

Examples: Products that have visual displays could show the software designation (or some associated code explained in the documentation) during some time after switch-on. In products which have also a user interface, this information can be showed after using a certain key combination or other operations. Physical labels in the hardware are also possible.

9 Process of audit and evaluation of the SW allows to fulfils the Table R1 or R2

According to requirements of sub-clause R.3.2.2.1 the architecture for techniques and measures to control fault/errors shall be specified by the manufacturer and checked by NCB A. Acceptable techniques for the specification are described in R.3.2.2.1. The requirements of R.2.2.5 and R.2.2.6 defines that the source code of the applied measures has to be inspected by the NCB A. We recommend that this will be carried out together with the software developer (software development team). In the test report shall be introduced the relevant document reference, in which the chosen measures are described. In the column Verdict NCB A shall state which measures were applied.

Furthermore the sub-clause R.2.2.5 requires the testing of the source code. To fulfil this requirement, the software tester (software test team) of the manufacturer shall define suitable test cases. The test cases have to be carried out and documented together with the test plan accordingly. A description of the test environment (Debugger, Emulator, Simulator, Test equipment) shall be included. Afterwards relevant tests shall be repeated together with NCB A by spot checks. In the column “Document reference for applied tests” shall be introduced the document reference for the applied test cases, test data and test environment.

For the more detailed description in the test report IEC60335_1m, the following tables shall be added. With a document reference a better confidence for NCB B can be provided. Details about the content of the documents shall not be requested by NCB B.
### Table R.1 – General fault/error conditions

<table>
<thead>
<tr>
<th>Component 1)</th>
<th>Fault/error</th>
<th>Acceptable measures 2) 3) 4)</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference for applied test</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. CPU</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.1 Registers</td>
<td>Stuck at</td>
<td>Functional test, or periodic self-test using either:</td>
<td>H.2.16.5</td>
<td>H.2.16.6</td>
<td>H.2.19.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– static memory test, or</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>– word protection with single bit redundancy</td>
<td>H.2.19.8.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.2 Void</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3 Programme counter</td>
<td>Stuck at</td>
<td>Functional test, or periodic self-test, or independent time-slot monitoring, or logical monitoring of the programme sequence</td>
<td>H.2.16.5</td>
<td>H.2.16.6</td>
<td>H.2.18.10.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Interrupt handling and execution</td>
<td>No interrupt or too frequent interrupt</td>
<td>Functional test; or time-slot monitoring</td>
<td>H.2.16.5</td>
<td>H.2.18.10.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Clock</td>
<td>Wrong frequency (for quartz synchronized clock: harmonics/subharmonics only)</td>
<td>Frequency monitoring, or time slot monitoring</td>
<td>H.2.18.10.1</td>
<td>H.2.18.10.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Memory</td>
<td>All single bit faults</td>
<td>Periodic modified checksum; or multiple checksum, or word protection with single bit redundancy</td>
<td>H.2.19.3.1</td>
<td>H.2.19.3.2</td>
<td>H.2.19.8.2</td>
<td></td>
</tr>
<tr>
<td>Component 1</td>
<td>Fault/error</td>
<td>Acceptable measures 2 3 4</td>
<td>Definitions</td>
<td>Document reference for applied measure</td>
<td>Document reference for applied test</td>
<td>Verdict</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>---------------------------</td>
<td>-------------</td>
<td>--------------------------------------</td>
<td>-------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>4.2 Variable memory</td>
<td>DC fault</td>
<td>Periodic static memory test, or word protection with single bit redundancy</td>
<td></td>
<td>H.2.19.6</td>
<td>H.2.19.8.2</td>
<td></td>
</tr>
<tr>
<td>4.3 Addressing (relevant to variable and invariable memory)</td>
<td>Stuck at</td>
<td>Word protection with single bit parity including the address, or</td>
<td></td>
<td>H.2.19.18.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Internal data path</td>
<td>Stuck at DC fault</td>
<td>Word protection with single bit redundancy Comparison of redundant CPUs by either: – reciprocal comparison – independent hardware comparator, or</td>
<td></td>
<td>H.2.19.8.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.1 Void</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.2 Addressing</td>
<td>Wrong address</td>
<td>Word protection with single bit redundancy including the address</td>
<td></td>
<td>H.2.19.8.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1) Component
2) Fault/error
3) Acceptable measures
4) Definitions

Table R.1 (continued)
<table>
<thead>
<tr>
<th>Component 1)</th>
<th>Fault/error</th>
<th>Acceptable measures 2) 3) 4)</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference For applied test</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 External communication</td>
<td>Hamming distance 3</td>
<td>Word protection with multi-bit redundancy, or CRC – single word, or transfer redundancy, or protocol test</td>
<td>H.2.19.8.1 H.2.19.4.1 H.2.18.2.2 H.2.18.14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.1 Void</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.2 Void</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3 Timing</td>
<td>Wrong point in time</td>
<td>Time-slot monitoring, or scheduled transmission Time-slot and logical monitoring, or comparison of redundant communication channels by either: – reciprocal comparison – independent hardware comparator Logical monitoring, or time-slot monitoring, or scheduled transmission (same options as for wrong point in time)</td>
<td>H.2.18.10.4 H.2.18.10.3 H.2.18.15 H.2.18.3 H.2.18.10.2 H.2.18.10.4 H.2.18.18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Wrong sequence</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Input/output periphery</td>
<td>Fault conditions specified in 19.11.2</td>
<td>Plausibility check Comparison of redundant CPUs by either: – reciprocal comparison – independent hardware comparator, or</td>
<td>H.2.18.13 H.2.18.15 H.2.18.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.1 Void</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table R.1 (continued)

<table>
<thead>
<tr>
<th>Component 1)</th>
<th>Fault/error</th>
<th>Acceptable measures 2) 3) 4)</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference for applied test</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2 Analog I/O</td>
<td>Fault conditions specified in 19.11.2</td>
<td>Plausibility check</td>
<td></td>
<td>H.2.18.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2.1 A/D- and D/A- converter</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.2.2 Analog multiplexer</td>
<td>Wrong addressing</td>
<td>Plausibility check</td>
<td></td>
<td>H.2.18.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Void</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Custom chips 5) e.g. ASIC, GAL, Gate array</td>
<td>Any output outside the static and dynamic functional specification</td>
<td>Periodic self test</td>
<td></td>
<td>H.2.16.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When applicable in the specific Part 2 the software may be analysed with Table R2
<table>
<thead>
<tr>
<th>Component</th>
<th>Fault/error</th>
<th>Acceptable measures</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference for applied test</th>
<th>Verdict</th>
</tr>
</thead>
</table>
| 1. CPU    | DC fault    | Comparison of redundant CPUs by either:  
- reciprocal comparison  
- independent hardware comparator, or  
Internal error detection, or  
redundant memory with comparison, or  
periodic self-tests using either  
- walkpat memory test  
- Abraham test  
- transparent GALPAT test; or  
word protection with multi-bit redundancy, or  
static memory test and word protection  
with single bit redundancy                                                                 | H.2.18.15   | H.2.18.3                               | H.2.18.9                           | H.2.18.5                           |         |
| 1.1 Registers |            | Comparison of redundant CPUs by either:  
- reciprocal comparison  
- independent hardware comparator, or  
Internal error detection, or  
redundant memory with comparison, or  
periodic self-tests using either  
- walkpat memory test  
- Abraham test  
- transparent GALPAT test; or  
word protection with multi-bit redundancy, or  
static memory test and word protection  
with single bit redundancy                                                                 | H.2.18.15   | H.2.18.3                               | H.2.18.9                           | H.2.18.5                           |         |
| 1.2 Instruction decoding and execution | Wrong decoding and execution | Comparison of redundant CPUs by either:  
- reciprocal comparison  
- independent hardware comparator, or  
Internal error detection, or  
redundant memory with comparison, or  
periodic self-tests using equivalence class test                                                                 | H.2.18.15   | H.2.18.3                               | H.2.18.9                           | H.2.18.5                           |         |
| 1.3 Programme counter | DC fault | - independent time-slot and logical monitoring  
- internal error detection, or  
comparison of redundant functional channels by either:  
- reciprocal comparison  
- independent hardware comparator                                                                 | H.2.18.10.3 | H.2.18.9                              | H.2.18.3                           |         |
| 1.4 Addressing | DC fault | Comparison of redundant CPUs by either:  
- reciprocal comparison  
- independent hardware comparator; or  
Internal error detection; or  
periodic self-test using  
-a testing pattern of the address lines; or  
-full bit bus parity including the address  
-a multi bus parity including the address                                                                 | H.2.18.15   | H.2.18.3                               | H.2.18.9                           | H.2.16.7                           | H.2.18.22 | H.2.18.1.1 | H.2.18.1.2 |
<table>
<thead>
<tr>
<th>Component 1)</th>
<th>Fault/error</th>
<th>Acceptable measures 2) 3) 4)</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference for applied test</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 Data paths instruction decoding</td>
<td>DC fault and execution</td>
<td>Comparison of redundant CPUs by either: reciprocal comparison, or independent hardware comparator, or internal error detection, or periodic self-test using a testing pattern, or data redundancy, or multi-bit bus parity</td>
<td>H.2.18.15, H.2.18.3, H.2.18.9, H.2.16.7, H.2.18.22, H.2.18.1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Interrupt handling and execution</td>
<td>No interrupt or too frequent interrupt related to different sources</td>
<td>Comparison of redundant functional channels by either reciprocal comparison, independent hardware comparator, or independent time-slot and logical monitoring</td>
<td>H.2.18.15, H.2.18.3, H.2.18.10.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. Clock</td>
<td>Wrong frequency (for quartz synchronized clock: harmonics/subharmonics only)</td>
<td>Frequency monitoring, or time-slot monitoring, or comparison of redundant functional channels by either: reciprocal comparison, independent hardware comparator</td>
<td>H.2.18.10.1, H.2.18.10.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Memory 4.1 Invariable memory</td>
<td>99.6% coverage of all information errors</td>
<td>Comparison of redundant CPUs by either: reciprocal comparison, independent hardware comparator, or redundant memory with comparison, or periodic cyclic redundancy check, either single word, double word, or word protection with multi-bit redundancy</td>
<td>H.2.18.15, H.2.18.3, H.2.19.5, H.2.19.4.1, H.2.19.4.2, H.2.19.8.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Component 1)</td>
<td>Fault/error</td>
<td>Acceptable measures 2) 3) 4)</td>
<td>Definitions</td>
<td>Document reference for applied measure</td>
<td>Document reference for applied test</td>
<td>Verdict</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-------------------------------</td>
<td>-------------</td>
<td>---------------------------------------</td>
<td>---------------------------------</td>
<td>---------</td>
</tr>
</tbody>
</table>
| 4.2 Variable memory and dynamic cross links | DC fault | Comparison of redundant CPUs by either:  
– reciprocal comparison  
– independent hardware comparator, or redundant memory with comparison, or  
periodic self tests using either:  
– walkpat memory test  
– Abraham test  
– transparent GALPAT test, or word protection with multi-bit redundancy |  
H.2.18.15  
H.2.18.3  
H.2.19.5 |  
H.2.19.7  
H.2.19.1  
H.2.19.2.1 |  
H.2.19.8.1 |---------|
| 4.3 Addressing (relevant to variable and invariable memory) | DC fault | comparison of redundant CPUs by either:  
– reciprocal comparison, or  
– independent hardware comparator, or full bus redundancy  
Testing pattern, or  
periodic cyclic redundancy check, either:  
– single word  
– double word, or word protection with multi-bit redundancy including the address |  
H.2.18.15  
H.2.18.3  
H.2.18.1.1 |  
H.2.18.22  
H.2.19.4.1  
H.2.19.4.2 |  
H.2.19.8.1 |---------|
| 5 Internal data path  
5.1 Data | DC fault | Comparison of redundant CPUs by either:  
– reciprocal comparison  
– independent hardware comparator, or word protection with multi-bit redundancy including the address, or data redundancy, or protocol test |  
H.2.18.15  
H.2.18.3  
H.2.19.8.1 |  
H.2.18.2.1  
H.2.18.14 | |---------|
| 5.2 Addressing Wrong address and multiple addressing | | Comparison of redundant CPUs by:  
– reciprocal comparison  
– independent hardware comparator, or word protection with multi-bit redundancy including the address, or full bus redundancy; or testing pattern including the address |  
H.2.18.15  
H.2.18.3  
H.2.19.8.1 |  
H.2.18.1.1  
H.2.18.22 | |---------|
<table>
<thead>
<tr>
<th>Component</th>
<th>Fault/error</th>
<th>Acceptable measures</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference for applied test</th>
<th>Verdict</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1 Data</td>
<td>Hamming distance 4</td>
<td>CRC – double word, or data redundancy or comparison of redundant functional channels by either: – reciprocal comparison – independent hardware comparator</td>
<td>H.2.19.4.2</td>
<td>H.2.18.2.1</td>
<td>H.2.18.3</td>
<td></td>
</tr>
<tr>
<td>6.2</td>
<td>Wrong address</td>
<td>Word protection with multi-bit redundancy, including the address, or CRC single word including the addresses, or transfer redundancy or protocol test</td>
<td>H.2.19.8.1</td>
<td>H.2.19.4.1</td>
<td>H.2.18.2.2</td>
<td>H.2.18.14</td>
</tr>
<tr>
<td></td>
<td>Wrong and multiple addressing</td>
<td>CRC – double word, including the address, or full bus redundancy of data and address, or comparison of redundant communication channels by either: – reciprocal comparison – independent hardware comparator</td>
<td>H.2.18.15</td>
<td>H.2.18.3</td>
<td>H.2.18.2.2</td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>Wrong point in time</td>
<td>Time-slot monitoring, or scheduled transmission</td>
<td>H.2.18.10.4</td>
<td>H.2.18.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.1</td>
<td>Fault conditions specified in 19.11.2</td>
<td>Comparison of redundant CPUs by either: – reciprocal comparison – independent hardware comparator, or input comparison, or multiple parallel outputs; or output verification, or testing pattern, or code safety</td>
<td>H.2.18.15</td>
<td>H.2.18.3</td>
<td>H.2.18.8</td>
<td>H.2.18.11</td>
</tr>
</tbody>
</table>
### Table R2 (concluded) \(^4\)

<table>
<thead>
<tr>
<th>Component (^1)</th>
<th>Fault/error</th>
<th>Acceptable measures (^2) (^3) (^4)</th>
<th>Definitions</th>
<th>Document reference for applied measure</th>
<th>Document reference for applied test</th>
<th>Verdict</th>
</tr>
</thead>
</table>
| 7.2 Analog I/O 7.2.1 A/D- and D/A- convertor | Fault conditions specified in 19.11.2 | Comparison of redundant CPUs by either:  
- reciprocal comparison  
- independent hardware comparator, or input comparison, or  
multiple parallel outputs, or output verification, or testing pattern | H.2.18.15 | H.2.18.3 | H.2.18.8 | H.2.18.11 | H.2.18.12 | H.2.18.22 |
| 7.2.2 Analog multiplexer | Wrong addressing | Comparison of redundant CPUs by either:  
- reciprocal comparison  
- independent hardware comparator, or input comparison or testing pattern | H.2.18.15 | H.2.18.3 | H.2.18.8 | H.2.18.11 | H.2.18.12 | H.2.18.22 |
| 8. Monitoring devices and comparators | Any output outside the static and dynamic functional specification | Tested monitoring, or redundant monitoring and comparison, or error recognizing means | H.2.18.21 | H.2.18.17 | H.2.18.6 |
| 9. Custom chips \(^5\) e.g. ASIC, GAL, Gate array | Any output outside the static and dynamic functional specification | Periodic self-test and monitoring, or dual channel (diverse) with comparison, or error recognizing means | H.2.16.7 | H.2.16.2 | H.2.18.6 |
10 Process to evaluate the software related safety independently of functional software and the way to proceed when changes are done on it

The software safety related and functional software shall be separated to avoid that functional part causes malfunction to safety related segments. With modular approach is easy to divide safety from functional code avoiding to insert safety and functional part of the software in the same module.

Modular approach rule for the design and coding software requests that a software module should have a single well-defined task or function to fulfil (not all software design follow this rule).

Separation of safety and functional modules shall be implemented at data exchange, also. Functional and safety part of the software can exchange data each other but all safety related data must be managed directly from safety code which make available the information required to functional part and not the opposite.

When is necessary to operate changes to safety software is very important that concepts explain before are respected. In this way software changes can be managed by manufacturer and validate more easily.

When manufacturer decide to make changes, at the beginning, shall provide documentation to describe modification. Information required should be:

- Detailed description of the modification
- Impact of the modification related to the entire safety software with modified modules list with new module versions
- Source code comparison before/after modification
- New software version, date of release and sign (checksum, CRC)

About source code comparison is possible to use software tools that make the comparison faster (text comparison). From this tools are possible to check source code changing and comments inside line, also.

From documentation analysis can result:

- modifications are related to a specifically module or part of the program
- modifications are related to several modules or parts of the program

If modifications are related to case 1 only affected module or part of the program have to be revalidated and the related function must be tested.

If modifications are related to case 2 every affected module, or parts of the program have to be revalidated and related functions shall be tested.

In case modified modules (or parts of the program) have more than one function, the impact of the modification shall indicate if functions inside the module are linked than all functions must be tested.