

Publisher: Gesamtverband der Deutschen Versicherungswirtschaft e.V. (GDV)
 Büro Schadenverhütung
 Publishing house: © VdS Schadenverhütung

Rules for fire prevention and security technology

Software

Requirements and test methods

VdS 2203en : 2001-03 (02)

Content

Preliminary remarks	2
1 General	2
1.1 Scope	2
1.2 Validity	3
2 Normative references	3
3 Terms and definitions	3
4 Requirements and test methods	4
4.1 General	4
4.2 Main program flow	5
4.3 Memory map	6
4.4 Interfaces between hard- and software	7
4.5 Detailed program documentation	7
4.6 Source code listing	8
4.7 Software tools	9
4.8 Modularity	9
4.9 Interfaces, plausibility	9
4.10 Deadlock protection	10
4.11 Program monitoring	10
4.12 Storage of program and data	11
4.13 Monitoring of the memory contents	11
Changes	12
Annex A Third-party products (normative)	13
Annex B Version scheme (normative)	14
Annex C Bibliographic references (informative)	15

Preliminary remarks

Like other areas of technology, the proportion of programmable electronic devices and systems is also increasing steadily in the field of fire protection and security technology. The functionality offered by these devices depends to a very large extent on their programs which therefore are of special meaning regarding their performance and documentation.

These rules are a revision of the previous “*Rules for security systems, software-controlled system parts, supplementary requirements and test methods, VdS 2203 12/88 (01)*” and essentially convert the requirements laid down in the EN 54 series of European standards for test methods. They describe how to verify the requirements of these series of standards and lay down criteria for demonstrating conformity. To this end, the relevant requirements from the above mentioned EN European standards and draft standards have been analogically adopted.

The concept behind the requirements and their implementation in the test methods described in this document is based on the condition that well-documented software shall satisfy the following points:

- A comprehensible and reproducible design process has been adopted.
- It is guaranteed that the software is correctly maintained and updated by expert software engineers even if the original designers and planners are no longer available.
- Measures have been implemented to avoid certain frequent causes of unreliability or to identify errors in conjunction with the software.

The documentation should therefore describe the software in such a way that it is possible to trace back the design process and programming operations. Furthermore the documentation should be complete, correct and consistent. The test based on the requirements is, however, no verification of the software and may therefore not guarantee that the software or the documentation are free of faults.

1 General

1.1 Scope

These rules apply to all programs and program components in the field of security technology which are used for processing, presenting and transmitting messages, data, signals and information. They are valid in addition to the rules for security products (e.g. intrusion control panels). Requirements and test methods which do not apply for security devices are marked explicitly by an appropriate note. The same applies to requirements and test methods which, according to EN 54 (parts 2,5, and 7) are only applied for certain devices (e.g. fire alarm control and indicating equipment - CIE).

These rules also apply to control equipment (e.g. drives) in the field of fire prevention and security technology, components of access control systems, and electronic locks. They do not cover software with assistance purposes (e.g. project planning tools).

1.2 Validity

These rules are valid from 03-01-2001; they replace the edition VdS 2203 12/88 (01) which may be applied, however, for a transition period until 03-01-2002.

Note: This is a translation of the German rules; if there are any discrepancies, the German version shall be binding.

2 Normative references

These rules contain dated and undated references to other publications. The normative references are cited at the appropriate places in the clauses, the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to these rules only when announced by a change of these rules. For undated references the latest edition of the publication referred will be applied.

- **EN 54-2 : 1997-10-00** Automatic fire detection systems – Part 2: Control and indicating equipment, German version EN 54-2,1997 (1997-12-00)
- **EN 54-2/AC : 1999-02-00** Automatic fire detection systems – Part 2: Control and indicating equipment; AC (1999-02-00)
- **prEN 54-5 : 2000-02-00** Automatic fire detection systems – Part 5: Heat detector: Point detectors (2000-02-00)
- **prEN 54-7 : 2000-02-00** Automatic fire detection systems– Part 7: Smoke detector – Point detectors using scattered light, transmitted light or ionisation (2000-02-00)

3 Terms and definitions

Site-specific data: Alterable data required for defining a specific system configuration for a device, e.g. an alarm system control and indicating equipment (CIE). It includes e.g.

- Assignment of detector zones
- Logical links of detector zones
- Auxiliary operation-related functions (e.g. alarm buffering etc.)
- Assignment of alarm outputs
- Assignment of areas
- Defining the function of the lines

Data: Pattern of characters or continuous functions which provide information on the basis of known or assumed conventions, primarily for the purpose of processing or as a result of processing.

Volatile memory: Memory elements which require the presence of energy source for the retention of their contents.

Global / local data: A name agreed in a module and used only in this specific module is local to this module. If a name occurs in a module which has not been agreed in this module's declaration section, the name shall be agreed in a superordinate module or program or shall be the name (i.e. a function name) or

formal parameter of the superordinate module or program. Such names – which are neither names (function names) nor formal parameters of this module, are known as global and we therefore speak of global constants, global variables etc.

Module: A part which the logic of a program sees as separate and which performs a specific and unique task. The subdivision of a program into individual modules can also be based on hierarchical criteria. Where procedures consist of several programs, a modular structure can exist if each individual program performs an independent function which can be viewed in isolation.

Non-volatile memory: Memory elements which do not require the presence of energy source for the retention of their contents.

Program: A specific syntactic unit defined in accordance with the rules of the language used and consisting of statements and agreements and comprising the elements required for resolving a specific task.

Programming language: A language created for writing programs.

Interface: Theoretical or actual transition at a boundary between two functional units with agreed rules for transferring data and signals.

4 Requirements and test methods

4.1 General

4.1.1 Provision of documentation

The manufacturer shall elaborate and maintain a documentation giving an overview of the performance of the software. This shall be detailed enough to enable a test on the verification of the requirements of these rules.

Except single documents where the decision of the forwarding is explicitly left to the manufacturer the documentation shall be submitted together with the device at the testing centre.

Where documents do not need to be submitted to the testing laboratory, but shall be held available at all times for inspection (see EN 54 – 2, fire detection systems – Part 2: fire alarm Control and Indicating Equipment, German version EN 54-2, 1997 (1997-12-00)), the following procedure may be applied:

- Using the program hierarchy and complete program structure (e.g. flow chart for the main program), the program modules or program paths to be tested are specified by the test laboratory. These documents shall then be separated from the available documentation and presented without delay to the test laboratory.
- Alternatively, where documentation is particularly confidential (e.g. algorithms), either the manufacturer will make this documentation available on a temporary basis and/or only in his presence or the documentation in question will be examined at the manufacturer's premises.
- In the case of classified secret materials, the documentation in question can be excluded from the test.

4.1.2 Forms of documentation

Software documentation does not need to correspond to any predefined form (e.g. paper). The scope, content and relevance of the individual documents are subject of the test. The documents form related sets of information showing clearly and uniquely the required content.

The software documentation should be structured as part of the entire document for a particular device, such that is possible to trace signals or information from the detection over processing up to a resulting action or display. There are restrictions in this regard for parameterisable control and indicating equipment for alarm systems where information processing is depending on the parameters set.

The documentation shall be complete, clear, precise and consistent.

All parts of the documentation shall be clearly identifiable. References to other documents shall also be clear and precise. It shall also be possible using the documentation to determine the presence and characteristics of all functions and properties required by these rules.

If the programmable electronic device or system consists of several components which are program-controlled, each of these shall be fully documented separately of each other.

The same requirements apply for user-programmed components for which programming has been performed using a programming language for abstracting the circuit design.

The requirements described in Annex A apply for programs and their documentation which have not been produced by the manufacturer himself or which have not been created by third parties on behalf of the manufacturer.

The entire documentation or individual documents can also be recorded and stored on electronic media. However, they shall be made available to the testing centre in legible form.

4.2 Main program flow

4.2.1 Requirements

The documentation on the main program flow shall provide at least a functional description of the main program flow comprising following items:

- a) a brief description of each module and the task it performs
- b) a description showing the interactions and dependencies of the modules and the objects to each other
- c) the overall hierarchy of the program
- d) a description of the way in which the modules are called, including details of any interrupt processing (e. g. description of software-interfaces including input- and outputparameters)

The functional description of the main program flow shall be based on clear and precise methods which are suitable for the software in question (e.g. graphical presentations of the system design, data flow and control flow).

4.2.2 Test

The documentation will be examined to verify that it contains a functional description of the main program flow as well as the contents listed in paragraph 4.2.1 a) to d). It is checked if the chosen form of the documentation is structured in clear and precise methods which are suitable for the software in question and describing clearly the required contents.

Remark 1: This will generally include a presentation of the program structure in diagram form (e.g. flow charts or state event charts). Where programs are simple, this may take the form of a single diagram. Where programs are more complex, a hierarchical series of diagrams may be needed (the examples cited are merely examples, many other options are also possible).

Remark 2: It is not necessary that the items mentioned in paragraph 4.2.1 a) to d) are proven separately but may be fulfilled together. Their description must not be documented separately but may be integrated in another document (e. g. source code header files).

The documents/paragraphs containing the required information as well as the form of presentation (e. g. diagram type) are ascertained and recorded. The results are included in the test report.

4.3 Memory map

4.3.1 Requirements

Regarding the execution process the documentation shall include at least a description on which areas of the memory are used for storing the program data, the site-specific data and the runtime data.

Where dynamic memory management is applied, a separation shall be implemented between program data, site-specific data and runtime data and this shall be described in connection with the method of memory allocation.

4.3.2 Test

The documentation is examined to determine whether it describes the memory areas for the various purposes such as program storage, site-specific data and runtime data. It shall be at least possible to identify the memory type for program storage, site-specific data and runtime data. The documentation shall give proof that the site-specific data are separated from the dynamic data (runtime data) and cannot be overwritten by the latter.

Remark: This is frequently done in the form of diagrammatic or tabular memory maps or using linker/locator listings. For parameterisable devices (e.g. alarm system control and indicating equipment – CIE) it is only necessary to state the maximum possible storage area for parameterisation data due to the variable size of the parameterisation data.

The documents/paragraphs containing the above information and the type of description are ascertained and recorded. The results are included in the test report.

4.4 Interfaces between hard- and software

4.4.1 Requirements

The documentation shall contain a description of the interfaces between hardware and software showing how the software interacts with the hardware of the unit being tested. The manufacturer shall provide at least two examples to show how the software acts on the hardware.

4.4.2 Test

The documentation will be used to examine whether there is a description of how the software interacts with the hardware of the unit.

Remark: An account of the port assignments is one way of demonstrating that this is documented.

The documents/paragraphs containing the above information and the type of description are ascertained and recorded. The results are included in the test report.

4.5 Detailed program documentation

4.5.1 Requirements

The manufacturer shall prepare and maintain detailed documentation on software execution. This does not need to be submitted to the test laboratory. However, it shall be made available for inspection in a means which respects the manufacturer's rights of confidentiality. The detailed program documentation shall contain a description of each program module as implemented in the source code and containing the following information:

- a) name of the modules
- b) date and/or version reference (see also Annex B)
- c) a description of the tasks performed
- d) a description of the interfaces including type of data transfer, the valid value range and a check of the valid data unless this is implicitly given by the test procedures on the used language

An overview of the entire system configuration, together with all software and hardware components shall also be included.

4.5.2 Test

A check is carried out to determine whether a detailed documentation is available containing a description of each program module. Each of these descriptions should contain at least the content of clause 4.5.1 a) to d).

Remark: It is not necessary for the description to be provided in a separate document, it can also be integrated into another document, e.g. into the source code header files.

At least 3 of the modules are randomly sampled to check whether the descriptions are consistent with the program flow and can be identified hierarchically. This can be done by selecting a path through the hierarchy and following this path through the module descriptions and/or source code listings.

A check is conducted to determine whether the documentation provides an overview of the entire system, together with all software and hardware components.

The documents/paragraphs containing the above information and the type of description are ascertained and recorded. The results are included in the test report.

4.6 Source code listing

4.6.1 Requirements

The manufacturer shall draw up and maintain detailed documentation on software execution. This does not need to be submitted to the test laboratory. However, it shall be made available for inspection in a means which respects the manufacturer's rights of confidentiality. The source code documentation shall cover the source code listing together with all global and local variables, constants and labels and comments which are sufficient for the program flow to be identified.

4.6.2 Test

A check of the consistency between module descriptions and source code listings is carried out using sampling procedures (e.g. the check examines whether a source code listing exists for each described module and that it is conform with the description of the module in question).

A check is carried out to determine whether the source code has been suitably commented for a software engineer who is familiar with the programming language and whether the program flow can be followed.

Remark: In general, more comments are required for lower-level languages (e.g. Assembler) than for higher-level languages (e.g. C++) in order that the program flow can be identified. The following list should be used as an aid in assessing appropriate annotation; the individual points do not represent fixed criteria, however.

- *Routines within a module should be commented with their purpose and the variables influenced by them.*
- *Global constants, variables and entry markers (jump addresses) should be commented to provide information on their meaning, at least the first time they are mentioned. The data types and permissible ranges of numerical data should be included where appropriate and the purpose of each bit should be specified for a variable used as a bit field (e.g. for the purpose of providing status flags).*
- *Points where data forms input to or output from modules should be commented.*
- *Control structures (e.g. "if-then statements", "case statements" etc.) and other decision points should be commented with details of their purpose and results in order to identifying the program flow.*
- *The start and end of iterative loops should be commented in order to show their purpose and any nesting.*
- *Setting and resetting of flags should be identified with comments.*

- *When Assembler code is used, each logical step or group of connected steps which perform an identifiable function should be commented with details of their purpose in order to identify the program flow.*

4.7 Software tools

4.7.1 Requirements

The documentation shall at least contain details of the software tools used for programming (e. g. high level design tools, compiler, assembler, etc.) In particular the compilers (including the version) for generating the source code shall be listed. This does not need to be submitted to the test laboratory. However, it shall be made available for inspection in a means which respects the manufacturer's rights to confidentiality.

4.7.2 Test

A check is conducted to determine whether the documentation provides information on the software tools used for developing this software.

The documents/paragraphs containing the above information are ascertained and recorded. The results are included in the test report.

4.8 Modularity

Remark: This paragraph is not necessary provided modules according to clause 4.5 are available.

4.8.1 Requirements

The software shall have a modular structure to ensure reliable operation of the unit.

4.8.2 Test

The documentation is used to check whether the software has a modular structure.

The documents/paragraphs containing the above information are ascertained and recorded. The results are included in the test report.

4.9 Interfaces, plausibility

4.9.1 Requirements

To ensure the unit operates reliably, the design of the interfaces for data generated manually and automatically shall prevent faults occurring in the program flow as a result of invalid data.

4.9.2 Test

The documentation is used to determine whether the software contains protection against invalid data.

Remark: In general, this is achieved by using mechanisms such as error testing at input interfaces and limiting the validity range. However, it can also be proven by implicit data security based on a highly developed language concepts such as object oriented programming.

The documents/paragraphs containing the above information are ascertained and recorded. The results are included in the test report.

4.10 Deadlock protection

4.10.1 Requirements

To ensure the unit operates reliably, the software shall be designed in order to prevent endless loops (“deadlocks”) occurring.

4.10.2 Test

The software will be examined for possible causes of deadlocks and measures to avoid or eliminate them.

Remark: The program flow and sampling procedures should be used to identify sources of possible deadlocks (e.g. interfaces with hardware, recursive routines, iterative loops, unconditional jumps pointing backwards). Appropriate measures for avoiding endless loops and deadlocks should then be examined using these possible sources of error. Possible measures can include time-outs at interfaces and appropriate software concepts based on the operating system.

The documents/paragraphs containing the above information are ascertained and recorded. The results are included in the test report.

4.11 Program monitoring

This clause is not valid for fire detectors and security technology equipment where program monitoring is not required in the relevant product rules.

4.11.1 Requirements

The execution of the program shall be monitored. If routines linked to main functions of the program are not executed, the unit shall display a system fault reps. react as required in the relevant product rules.

However, the requirements of EN 54 – 2/AC, fire detection systems – Part 2: control and indicating equipment; AC (1999-02-00) apply to program flow monitoring of fire alarm central units.

4.11.2 Test

A test is conducted to determine whether program flow monitoring is performed in accordance with clause 4.11.1.

Remark: In general, this should be demonstrated using the program documentation and in particular the position of the watchdog trigger points in the source code. Proof that this functionality is implemented in the unit is also possible using an embedded system which simulates a fault in the program flow (however, not just a time-based error, e.g. by shorting the crystal). The required fault message shall then be signalled.

The documents/paragraphs containing the above information are ascertained and recorded. The results are included in the test report. Alternatively a fault message triggered by a simulated hardware fault may be checked. The results are included in the test report.

4.12 Storage of program and data

4.12.1 Requirements

The program and data (the data includes presettings such as manufacturer's settings, but not running data) shall be stored in non-volatile memories. It shall only be possible to write in the storage areas holding the program and data by using special tools or access codes and this shall not be possible during normal operation.

Additionally for **fire detectors** site-specific data shall be stored in memories capable of maintaining their content for at least two weeks without external power, unless measures are taken to guarantee the renewal of data automatically within one hour of the power failure.

At variance to this, the program and data storage requirements set out in EN 54 – 2/AC, fire detection systems – Part 2: control and indicating equipment; AC (1999-02-00) apply for **fire alarm control and indicating equipment (CIE)**.

4.12.2 Test

The documentation is used to check whether the requirements regarding program and data storage in clause 4.12.1 are met.

The documents/paragraphs containing the above information are ascertained and recorded. The results are included in the test report.

4.13 Monitoring of the memory contents

This clause is only valid for automatic fire detection systems control and indicating equipment (CIE).

4.13.1 Requirements

The contents of the memories containing site-specific data shall be automatically checked at intervals not exceeding one hour. The intervals have to fulfil the requirements of the respective product rules. The monitoring device shall signal a system fault if the corruption of the memory contents is detected.

4.13.2 Test

The program documentation is used to check whether the contents of the memory containing site-specific data are automatically checked at regular intervals (in accordance with the requirements). The test examines whether the monitoring device signals a system fault if the corruption of the memory contents is detected.

Remark: An alternative way of verifying that a unit supports this functionality is possible using an embedded system which simulates a memory error of this type. The required fault signalling shall occur in this case, too.

The documents/paragraphs containing the above information are ascertained and recorded. Alternatively, fault signalling can also be verified following an embedded simulated memory error. The results are included in the test report.

Changes

Compared with edition VdS 2203 12/88 (01) the following changes have been made:

- The VdS rules have been harmonised with other existing regulations covering identical areas of application. In particular, the requirements defined in section 13 “Additional requirements on software-controlled fire alarm central units” of EN 54 - 2” (Requirements on fire alarm central units) and the requirements defined in EN 54 - 5 and - 7 (Requirements on smoke detectors) have been used as a basis for these rules.
- The fact that defined programming standards have been introduced in a number of companies is reflected in the increased flexibility in the means for demonstrating compliance.
- The scope has also been extended and is not just restricted to components of alarm systems.
- It is explicitly stated that alternative means of providing detailed program documentation are possible.
- Also emphasised is the fact that documents are not examined for their form, but solely for their contents.
- Annex A of these rules formulates criteria for assessing third-party products which are used but the manufacturer has not developed himself.
- Annex B also provides information about version assignment with regard to ensuring the distinguishability and uniqueness of different program versions.

Annex A Third-party products (normative)

The following requirements set out criteria for evaluating so-called “third-party products”:

Preliminary remarks

Third party products are purchased programs or program components which are freely available and used for different applications.

Purchased programs or program components can take the form of

- Operating systems
- Compilers / debuggers
- Databases
- Driver programs, e.g. for graphics systems, LCD displays
- Network software

These programs or program components do considerably influence the reliability of programs of the programmable electrical device or system in question. Therefore all information concerning the reliability on purchased products shall be available.

The expression third-party products in the sense of these rules does not include programs or program components which are developed on behalf and for the exclusive use of the manufacturer of the device or system in question at an external company. These program components are expressively only to test according to these rules.

A1 Tools, libraries

If the design is to use programs or program components which are purchased or supplied from third parties, these shall be clearly identifiable from or dealt with separately in the documentation.

A2 Suitability of third-party products, approval, dealer certification

The manufacturer shall have tested and approved purchased programs or program components regarding their suitability.

A dealer certification shall be submitted to show that the third-party products in question are freely available.

Annex B Version scheme (normative)

The manufacturer shall fix rules for a version scheme which he has to keep. This scheme shall be used to differentiate between the scope and effect of modifications to these programs. It shall be possible to identify clearly between different program versions. At least it shall be possible to distinguish between those changes to the product (the software in this case) which are subject to reporting requirements and those which are not.

Annex C Bibliographic references (informative)

Alper, M.:

Professionale Softwaretests; Praxis der Qualitätsoptimierung kommerzieller Software
Friedr. Vieweg & Sohn Verlagsgesellschaft, Braunschweig, Wiesbaden 1994
ISBN 3-528-05454-9

Hindel, Dr. Bernd:

Beitrag zum Technologie-Forum "Embedded Software '97"
Qualität ist messbar: Software-Metriken
Firma 3Soft GmbH, Erlangen

Koreimann, Dieter S.:

Lexikon der angewandten Datenverarbeitung
de Gruyter, Berlin 1977
ISBN 3-110-06991-1

Luck, Prof. Dr. Ing. H.; Schlossarek, U.:

Studie über softwaregesteuerte Gefahrenmeldeanlagen: Anforderungs- und Prüfkriterien der Universität Duisburg, Fachgebiet Nachrichtentechnik, Verlag VdS Schadenverhütung, Köln
VdS 2173

Nassi, I.; Shneiderman, B.:

Flow chart techniques for structured programming
in: Sigplan Notices 8 (1973), H. 8, S. 12-26

VDI-GIS (Hg.):

Software-Zuverlässigkeit, Grundlagen,
konstruktive Maßnahmen, Nachweisverfahren
VDI Verlag
ISBN 3-528-05454-9