

Software for computers in the safety systems of nuclear power stations

ICS 27.120.20; 35.080

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW



Document Sharing Hub

National foreword

This British Standard reproduces verbatim IEC 60880:1986 and implements it as the UK national standard.

The UK participation in its preparation was entrusted to Technical Committee NCE/8, Reactor instrumentation, which has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international/European committee any enquiries on the interpretation, or proposals for change, and keep the UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

A list of organizations represented on this committee can be obtained on request to its secretary.

From 1 January 1997, all IEC publications have the number 60000 added to the old number. For instance, IEC 27-1 has been renumbered as IEC 60027-1. For a period of time during the change over from one numbering system to the other, publications may contain identifiers from both systems.

Cross-references

The British Standards which implement these international publications referred to in this document may be found in the BSI Standards Catalogue under the section entitled “International Standards Correspondence Index”, or by using the “Find” facility of the BSI Standards Electronic Catalogue.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, the EN title page, pages 2 to 67 and a back cover.

The BSI copyright notice displayed in this document indicates when the document was last issued.

This British Standard, having been prepared under the direction of the Engineering Sector Committee, was published under the authority of the Standards Committee and comes into effect on 15 July 2001

© BSI 07-2001

Amendments issued since publication

Amd. No.	Date	Comments

ISBN 0 580 36880 7

**COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE
NORME DE LA CEI**

**INTERNATIONAL ELECTROTECHNICAL COMMISSION
IEC STANDARD**

Publication 880

Première édition – First edition
1986

**Logiciel pour les calculateurs utilisés
dans les systèmes de sûreté des centrales nucléaires**

**Software for computers in the safety systems
of nuclear power stations**



CONTENTS

Foreword.....3
Preface.....3
Introduction.....4

	Clause
1. Scope and object.....	4
2. Terms and definitions.....	5
3. Project structure.....	7
4. Software requirements.....	7
5. Development (design and coding) of safety system software.....	9
6. Verification.....	12
7. Hardware/software integration.....	14
8. Computer system validation.....	17
9. Maintenance and modification.....	18
10. Operation.....	39
Appendix A – System development life cycle and details of software requirements.....	22
Appendix B - Detailed recommendation for the development (design and coding) of safety related software.....	29
Appendix C – Outline for the software performance specification.....	48
Appendix D – Language, translator, linkage editor, etc.....	50
Appendix E – Software testing.....	52
Appendix F – List of documents needed.....	59
ALPHABETICAL INDEX.....	63

INTERNATIONAL ELECTROTECHNICAL COMMISSION

SOFTWARE FOR COMPUTERS IN THE SAFETY SYSTEMS OF NUCLEAR POWER STATIONS

FOREWORD

- 1) 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.
- 2) They have the form of recommendations for international use and they are accepted by the National Committees in that sense.
- 3) In order to promote international unification, the IEC expresses the wish that all National Committees should adopt the text of the IEC recommendation for their national rules in so far as national conditions will permit. Any divergence between the IEC recommendation and the corresponding national rules should, as far as possible, be clearly indicated in the latter.

PREFACE

This standard has been prepared by Sub-Committee 45A: Reactor Instrumentations, of IEC Technical Committee No. 45: Nuclear Instrumentation.

The text of this standard is based upon the following documents:

Six Months' Rule	Report on Voting
45A(CO)88-I/II/III	45A(CO)90

Further information can be found in the Report on Voting indicated in the table above.

The following IEC publications are quoted in this standard:

- Publications Nos. 557 (1982): IEC Terminology in the Nuclear Reactor Field.
 639 (1979): Nuclear Reactor. Use of the Protection System for Non-safety Purposes.
 643 (1979): Application of Digital Computers to Nuclear Reactor Instrumentation and Control.
 671 (1980): Periodic Tests and Monitoring of the Protection System of Nuclear Reactors.

Other publications quoted:

- ISO standard 2382/1 (1984): Data processing — Vocabulary — Part 01: Fundamental terms.
 IAEA Guide 50-SG-D3 (1980): Safety guide.

SOFTWARE FOR COMPUTERS IN THE SAFETY SYSTEMS OF NUCLEAR POWER STATIONS

INTRODUCTION

The basic principles for the design of nuclear instrumentation as specifically applied to the safety systems of nuclear power plants have been interpreted in existing standards with reference to hard-wired systems as the “Safety Guide 50-SG-D3” of the IAEA.

This standard has been developed to interpret these principles for the utilization of digital systems — multiprocessor distributed systems as well as larger scale central processor systems — in the safety systems of nuclear power plants. It discusses the software system principles and requirements and should be read in association with appropriate standards on computer hardware and system integration.

It is important to note that this standard establishes no additional functional requirements for safety systems.

Aspects for which special recommendations have been produced, due to the unique nature of computer systems and their software are:

- a) established hardware criteria as far as they affect the software, taking careful account of the high degree of interdependency between hardware and software;
- b) a general approach to software development to assure the production of the highly reliable software required;
- c) a general approach to software verification and computer system validation;
- d) procedures for software maintenance, modification and configuration control.

1. Scope and object

This standard is applicable to highly reliable software required for computers to be used in the safety systems of nuclear power plants for safety functions — Class 1 functions according to IEC Publication 643: Application of Digital Computers to Nuclear Reactor Instrumentation and Control. This includes the safety actuation systems, the safety system support features and the protection systems.

For the utilization of computer systems for non-safety functions the principles of IEC Publication 639: Nuclear Reactors — Use of the Protection System for Non-safety Purposes, are applicable.

This standard provides requirements for each stage of software generation, including design, development, qualification and operation as well as the documentation for each stage of the software generation for the purpose of achieving highly reliable software.

The principles applied in developing these requirements include:

- best available practice;
- top-down design methods;
- modularity;
- verification of each phase;
- clear documentation;
- auditable documents;
- validation testing.

Additional guidance and information on how to comply with the requirements of the main part of this standard is given in Appendices A to F. References to those appendices are given in brackets.

If practices differing from those of the appendices are used, they shall be documented and auditable according to the requirements of the main part of this standard.

2. Terms and definitions

Terms are used as defined in ISO Standard 2382 (up to 1984), except as given below.

2.1 *Application program*

A computer program that performs a task related to the process being controlled rather than to the functioning of the computer itself.

2.2 *Code compaction*

The purposeful reduction in memory size required for a program by the elimination of redundant or extraneous instructions.

2.3 *Computer (Ref. ISO Standard 2382/1)*

A programmable functional unit that consists of one or more associated processing units and peripheral equipment, that is controlled by internally stored programs and that can perform substantial computation, including numerous arithmetic operations or logic operations, without human intervention during a run.

Note. — A computer may be a stand-alone unit or may consist of several interconnected units.

2.4 *Computer program*

A set of ordered instructions and data that specify operations in a form suitable for execution by a digital computer.

2.5 *Computer system*

A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for the execution of the program.

2.6 *Data*

A representation of facts, concepts or instructions in a formalized manner suitable for communication, interpretation or processing by a computer.

2.7 *Defence in depth*

Provision of several overlapping subsequent limiting barriers with respect to one threshold, such that the threshold can only be surpassed, if all barriers have failed.

2.8 *Diversity*

Existence of different means of performing a required function (e.g. other physical principles, other ways of solving the same task).

2.9 *Fault tolerance*

The built-in capability of a system to provide continued correct execution in the presence of a limited number of hardware or software faults.

2.10 *Graceful degradation*

Stepwise reduction of system functions in response to detected failures while essential functions are maintained.

2.11 *Initialize*

To set counters, switches, addresses, or contents of storage devices to zero or other starting values at the beginning of, or at prescribed points in, the operation of a computer program.

2.12 *Integration tests*

Tests performed during the hardware/software integration process prior to computer system validation to verify compatibility of the software and the computer system hardware.

2.13 *Procedure*

A portion of a computer program which is named and which performs a specific task.

2.14 *Redundancy*

Provision of alternative (identical or diverse) elements or systems so that any one can perform the required function regardless of the state of operation or failure of any other (Ref. IAEA 50-SG-D8).

2.15 *Single failure criterion (IEC Publication 557: IEC Terminology in the Nuclear Reactor Field)*

A criterion applied to a system such that it is capable of performing its safety task in the presence of any single failure.

2.16 *Software*

Programs, procedures, rules and any associated documentation pertaining to the operation of a computer system.

2.17 *Software life cycle*

The period of time that starts when a software product is conceived and ends when the product is no longer available for use. The software life cycle typically includes a requirements phase, design phase, implementation phase, test phase, installation and check-out phase, operation and maintenance phase.

2.18 *Software modification*

Changes of already agreed documents leading to a change to the executable code or its data.

2.19 *Software modularity*

The software attribute that provides a structure of highly independent computer program units that are discrete and identifiable with respect to translating, testing and combining with other units.

2.20 *Validation*

The test and evaluation of the integrated computer system (hardware and software) to ensure compliance with the functional, performance and interface requirements.

2.21 *Verification*

The process of determining whether or not the product of each phase of the digital computer system development process fulfils all the requirements imposed by the previous phase.

3. Project structure

Any project will normally be divided up into a number of phases. Each phase is to some extent self-contained but will depend on other phases and will, in its turn, be depended on by others. These phases are informally recognizable by the specific activities pertinent to them.

For safety related applications, these phases shall be formalized and none of the identified phases shall be omitted.

3.1 General

The following general factors determine the activities in implementing a project:

- 3.1.1 The whole software life cycle shall be considered.
- 3.1.2 Each phase of the software life cycle shall be divided into elementary tasks with a well-defined activity for each of them.
- 3.1.3 Every product shall be systematically checked after each phase (B4.g).
- 3.1.4 The tasks of quality assurance should be run generally in parallel with the other tasks of the life cycle.
- 3.1.5 Each phase shall include generation of the appropriate documents (F3).
- 3.1.6 Each phase shall be systematically terminated by a critical review. Critical reviews form a major part of the verification process of the software project (see Clause 6).
- 3.1.7 Every verification step or critical review shall result in a report on the analysis performed, the conclusions reached and the resolutions agreed. This report shall be included in the documentation.
- 3.1.8 A list of documents required as a minimum through the software life cycle is given in Appendix F.

3.2 Software quality assurance plan

A quality assurance plan shall exist or be established at an early stage either as a part of the computer system specification (see Clause 4) or as a companion document.

Although the whole of this standard and its appendices can be considered as a quality assurance plan for safety related software, special quality assurance plans may be adopted for individual product phases or particular software components according to national standards or company regulations.

These plans should address all quality assurance procedures required during all phases of the software life cycle.

4. Software requirements

The software requirements (F.M2) shall be derived from requirements of the safety systems and are part of the computer system specification. The computer system specification is a description of the combined hardware/software system and states the objectives and functions assigned to the computer system.

The software requirements describe the product, not the project. They shall describe what has to be done and not how it has to be done. An acceptable approach to the development and content of the software requirements, which is consistent with this clause, is given in Appendix A.

Whereas the main purpose of the software requirements document is to form the basis for software development, the licensing aspects should not be neglected. Therefore, it may

contain aspects of minor importance to software design which are, however, a background for licensing. Such aspects may be:

- risk considerations;
- recommendations for functions or engineered safety features;
- other items that provide the background for specific requirements.

The software functional requirements represent an expansion of the functions which are assigned to the computer system and which are to be implemented by the computer system.

The software reliability requirements represent an expansion of the reliability requirements of the computer system. They shall be derived in a similar way to the functional requirements.

- 4.1 The computer system specification shall give an external and synthetic view of the functions to be implemented by the software of the computer system (A2.1).
- 4.2 The computer system configuration shall be described using as a background the reliability requirements and the environment of the system, since reliability properties and system configuration are closely connected (A2.2).
- 4.3 For the interaction between the computer system and the plant operator or any other person, the principles of IAEA “Guide 50-SG-D3” in general and especially those of Sub-clauses 7.12 and 7.3.2 are applicable.

This man-machine dialogue shall be specified with regard to:

- commissioning of the system and start up;
 - system operation;
 - system maintenance, and testing after commissioning (A2.3).
- 4.4 The interface between the computer system and any other system either within or outside the nuclear plant, wherever a direct connection exists or is planned, shall be identified and documented indicating the specific interfaces and the related software requirements (Ref. IEC Publication 639) (A2.4).
 - 4.5 Functions dedicated to the prevention of the release of radioactivity under accident conditions (A2.1) and their related system functions shall be described (A2.5). The functions to be performed should be defined rather than the means of implementation.
 - 4.6 The constraints between hardware and software shall be described (A2.6). A reference to the hardware requirements document shall be made.
 - 4.7 Special operating conditions such as plant commissioning and refuelling shall be described (A2.7).
 - 4.8 The computer system software shall continuously supervise both itself and the hardware (A2.8). This is considered a primary factor in the achievement of the overall system reliability requirements.
 - 4.8.1 Self-supervision shall meet the following requirements, unless it is proved that other means provide the same degree of safety:
 - a) no single failure shall be able to block directly or indirectly any function dedicated to the prevention of the release of radioactivity;
 - b) those parts of the memory that contain code or invariable data shall be monitored to prevent any changes.
 - 4.8.2 The safety system software shall be designed in such a manner that essential functions of the whole safety system be testable during the operation of the nuclear power generating station.

4.8.3 If any failure is detected during plant operation, appropriate and timely automatic actions shall be taken. This may require giving due consideration to avoiding spurious trips.

4.8.4 System self-checking shall not adversely affect the intended system functions.

4.8.5 The safety system software shall be designed so as to meet the requirements of periodic testing which takes place within specified maximum intervals (e.g. shut-down periods).

- a) Every single function shall be coverable by periodic testing.
- b) Any degradation of the execution of safety functions shall be detected.
- c) The basic self-checking functions shall also be testable during periodic tests.

The software should be able to collect automatically all information gained during periodic testing. Further details for periodic testing are given in Sub-clause 10.3.

4.9 Software functional requirements shall be presented according to a standard whose formality should not preclude readability (A2.9). The requirements shall be unequivocal, testable or verifiable, and realizable.

4.10 The use of a formal specification language may be a help to show coherence and completeness of the software functional requirements. Automatic tools may be used for this purpose.

4.11 The software functional requirements shall be provided:

- to the authors of the computer system requirements documents for assessment and approval prior to programming;
- to the software design team manager for approval and with respect to feasibility and readability;
- to the licensors with respect to compliance with the overall plant safety requirements for licensing approval.

5. Development (design and coding) of safety system software

The following recommendations provide a guide to good practice for writing software which is as error-free as possible from the very beginning and which can easily be verified.

The software functional requirements specification should be available, before the design and coding phase of program development begins.

5.1 Principles for development (design and coding)

5.1.1 The following general principles for development are based on experience in producing error-free and understandable software.

- The software design shall include self-supervision of control flow and data. On failure detection, appropriate action shall be taken in accordance with Sub-clause 4.8.
- The program structure should be based on a decomposition into modules.
- The program structure should be simple and easy to understand, both in its overall design and in its details. Tricks, recursive structures and unnecessary code compaction should be avoided.
- The final source program should be readable from start to end.
- Good documentation shall be provided.

5.1.2 From these principles the following recommendations are derived:

- a) measures to obtain the required reliability including self-supervision should be chosen at the outset of the development (B3);
- b) a top down approach to software development should be preferred to a bottom up approach (B1);
- c) a conceptual model of system structure should be adopted at the beginning of each software project (B2);
- d) the program should be written to allow easy testing (B4, B5);
- e) where standard software from a manufacturer or supplier is used, it should be shown to have operated satisfactorily (B2.c).

5.2 *Language, translator and other tools*

Guidance for selection of language, translator, etc., is given in Appendix D.

Even though a specific programming language cannot be required, the following may be considered as common basic rules for safety system programming languages.

- 5.2.1 Languages with a thoroughly tested translator should be used. If no thoroughly tested translator is employed, additional verification shall show that the result of the translation is correct.
- 5.2.2 The language should be completely and unambiguously defined, otherwise the use of the language shall be restricted to completely and unambiguously defined features. This applies in a similar way if there is any doubt about the correct translation of a specific language feature or a particular combination of such features.
- 5.2.3 Problem oriented languages are strongly preferred to machine oriented ones.
- 5.2.4 As well as the specific points mentioned in Appendix D, a programming language for safety systems and its translator should not prevent by their design:
 - error-limiting constructs;
 - translation-time type checking;
 - run-time type and array bound check, and parameter checking.
- 5.2.5 Automatic testing aids should be available.
- 5.2.6 The use of automatic tools is recommended.

5.3 *Detailed recommendations*

5.3.1 *Development*

In Appendix B a set of recommendations is given which specifies in detail the aspects identified in Sub-clause 5.1.

The headings of the individual recommendations of Appendix B as well as relevant clauses of this standard can also be attributed to the two major parts of program development, design and coding, according to the following table.

Procedural and structural aspects of design and coding of user program for safety system

	Procedural aspects	Item/ Clause	Structural aspects	Item/ Clause
Design	Changeability	1.a/4.	Control and access structures	2.a
	Top-down approach	1.b	Modules	2.b
	Intermediate verification	1.c/6.	Operating system	2.c
	Changes during development	1.d/9.	Execution time	2.d/4.
	System reconfiguration	1.e/9.	Interrupts	2.e
			Arithmetic expressions	2.f
			Plausibility check	3.a/4.
			Safe output	3.b/4.
			Branches and loops	4.a
			Subroutines and procedures	4.b
			Nested structures	4.c
			Data structures	4.e
Coding	Intermediate verification	1.c/6.	Modules	2.b
	Changes during development	1.d/9.	Execution time	2.d/4.
	Intermediate tests	4.g/6.	Interrupts	2.e
	Coding rules	5.d	Arithmetic expressions	2.f
			Plausibility check	3.a/4.
			Safe output	3.b/4.
			Memory contents	3.c
			Error checking	3.d
			Branches and loops	4.a
			Subroutines and procedures	4.b
			Nested structures	4.c
			Addressing and arrays	4.d
			Data structures	4.e
			Dynamic changes	4.f
			Sequences and arrangements	5.a
			Comments	5.b
			Assembler	5.c

5.3.2 Use of the recommendations

At the outset of any safety related programming project those recommendations should be selected and recorded which apply to the intended program development. The selection may include the change of priorities and further subdivide into details of individual recommendations.

In connection with particular problems it may be reasonable to modify the recommendations selected from Appendix B, in order to meet the goal of overall simplicity and understandability. These modifications shall be documented, taking account of the following:

- during the selection procedure of the recommendations, the safety relevance of particular software parts should be considered;
- the more limiting recommendations should be selected and applied to those program sections which are more critical to the safety of the system;
- program sections of outstanding safety relevance should be clearly identifiable from the system and data structure used;

- d) the selection procedure should take into account the available testing facilities and the intended validation strategy. If important parts are programmed diversely, a different verification strategy may be used for those parts;
- e) if difficulties arise during verification, a retrospective change of programming style may be necessary.

5.4 Documentation

- 5.4.1 During program development the end of the design phase shall be marked by production of a formal document, the software performance specification (F.M3). This document serves as the basis for the formal design review and the subsequent program coding.

Sufficient detail shall be included so that program coding can proceed without further design clarification. The document should be structured for continued expansion in parallel to the design process. An outline of the suggested content is given in Appendix C.

- 5.4.2 In addition to the software performance specification, other documents may be required. They are used for both intermediate and final program verification steps. Some of these documents relate to the first design steps. They are derived from the functional requirements specification and provide the basis for the software performance specification. Others are derived from this particular document and represent the basis for later coding.

If the recommendations for the development procedure according to Appendix B are met, appropriate documentation is provided as a byproduct of program development.

- 5.4.3 The aim is an integrated set of documents. Each document shall have a defined relationship to the other document and contain a well bounded subject-matter.

- 5.4.4 Documentation formats should be selected according to the specific purpose, including:

- narrative description;
- arithmetic expressions;
- appropriate diagrams and drawings.

As a general rule it is preferable to choose a graphical representation. The documentation itself shall follow national standards.

6. Verification

6.1 Software verification process

After the software functional requirements have been established and before the next phase begins, verification addresses the adequacy of the software functional requirements in fulfilling the safety system requirements assigned to the software by the computer system specification.

After the design phase and before the next phase begins, verification addresses the adequacy of computer system software design as documented in the software performance specification to the software functional requirements.

After the coding phase and before the next phase begins, verification addresses the compliance of the coded computer system software to the software performance specification as derived by the design phase.

Each phase shall be completed by verification (A1).

6.2 *Software verification activities*

6.2.1 *Verification plan*

Concurrently with the phases of the software development cycle described above a software verification plan shall be established. The plan shall document all the criteria, the techniques and tools to be utilized in the verification process. It shall describe the activities to be performed to evaluate each item of software and each phase to show whether the functional and reliability requirements specification is met. The level of detail shall be such that an independent group can execute the verification plan and reach an objective judgement on whether or not the software meets its performance requirements.

Note. — The requirements for an independent group implies verification either by an individual or an organization which is separate from the individual or organization developing the software. The most appropriate way is to engage a verification team.

The verification plan shall be prepared by a verification team addressing:

- a) selection of verification strategies, either systematic, random or both, with test case selection according to either required functions, special features of program structure, or both, see Appendix E;
- b) selection and utilization of the software test equipment;
- c) execution of verification;
- d) documentation of verification activities;
- e) evaluation of verification results gained from verification equipment directly and from tests, evaluation of whether the reliability requirements are met.

The management of the verification team shall be separate and independent from the management of the development team.

6.2.2 *Design verification, critical reviews*

The design verification addresses:

- a) the adequacy of the software performance specification for the software functional requirements with respect to consistency and completeness down to and including the modular level;
- b) the decomposition of the design into functional modules and the manner of specification with reference to:
 - feasibility of the performance required;
 - testability for further verification;
 - readability by the development and verification team;
 - maintainability to permit further evolution;
- c) the respect of quality requirements.

The result of the design verification shall be documented in a report (F.M5).

This report shall include:

- items which do not conform to the software functional requirements;
- items which do not conform to the design standards;
- modules, data, structures and algorithms poorly adapted to the problem.

6.2.3 *Coding phase verification*

The code verification activities begin with module testing and go up through the software by a bottom-up strategy. At the module level, the software is not yet integrated into the system, therefore it can be extensively tested. The purpose of module testing is to show that each module performs its intended function and does not perform unintended functions. A

module integration test shall be performed to show at the earliest stage of development that all modules interact correctly to perform the intended function.

Guidance for code verification activities is given in the software test specification.

6.2.3.1 *Software test specification*

The software test specification is one of the principal documents of the verification plan for the coding phase. This document shall be based upon a detailed examination of the software functional requirements and shall give detailed information on the tests to be performed addressing each of the components of the software (modules and their constituents).

The software test specification is based on a general description of the software being tested. The description is supplied by the design team.

The software test specification shall include:

- a) the environment in which the tests are run;
- b) the test procedures;
- c) acceptance criteria i.e. a detailed definition of the criteria to be fulfilled in order to accept modules and major software components on subsystem and system level;
- d) error detection and corrective action procedures;
- e) a list of all documents that should be produced during the code verification phase.

6.2.3.2 *Software test report*

The software test report shall present the results of the test program described in the software test specification stating whether or not the software has achieved the performance requirements given in the software functional requirements. This document (F.M4) shall allow the complete diagnosis of design and performance deficiencies. It shall also include the resolution of all software deficiencies and test discrepancies discovered during the test.

The software test report shall include the following items both for the module and major design levels:

- a) hardware configuration used for the test;
- b) storage medium used and access requirements of the final code tested;
- c) input test listing;
- d) output test listing;
- e) additional data regarding timing, sequence of events, etc.;
- f) conformance with acceptance criteria given in the test specification;
- g) error incident log which describes the character of the error and the remedies taken by the design team.

7. **Hardware/software integration**

The process of hardware/software integration is the combining of verified hardware and software modules into a system that is capable of performing specified functions. This process consists of four parts:

- a) assembling hardware modules by interconnecting wiring according to the system design drawings;
- b) assembling software modules by a linkage processor;

- c) loading the software into the hardware;
- d) verifying by testing that the hardware/software interface requirements have been satisfied and that the software is capable of operating in that particular hardware environment.

7.1 *System integration plan*

The system integration plan shall be prepared and documented in the integration requirements (A1) and verified against the safety system requirements. The system integration plan (F.M2) describes the organizational and procedural aspects of the hardware/software integration.

This plan shall specify the standards and procedures to be followed in the hardware/software integration and shall document those provisions of the overall quality assurance plan that are applicable to the system integration.

The system integration plan shall discuss any constraints made by the specific design of the hardware and the software. The plan shall include the requirements for procedures and control methods covering:

- system configuration control;
- system integration;
- integrated system verification testing;
- error resolution.

In the process of verifying the individual hardware and software modules, certain aspects of the design of these modules may be better tested at the integrated system level. But if it is not feasible to test fully at this level, then all functional requirements of the individual module design shall be tested by other means. All interdependencies between the verification of the individual modules and the verification testing of the integrated system shall be documented in the system integration plan.

7.2 *Relationship of system integration to hardware and software development*

The system integration plan shall be prepared sufficiently early in the development process to allow any integration requirements to be included in the design of the hardware and software.

7.3 *System configuration control*

The system integration plan shall establish a library of software and hardware modules as the means for system configuration control. This library shall provide revision control for all hardware and software modules to be used in the system.

Adequate procedures shall be established to implement this library function and shall provide for the following tasks of the library function:

- the procedure shall ensure that no module or revision is entered into the library until it has been verified;
- the procedure shall ensure that the system integration is performed using the proper revision levels of the individual modules;
- the procedure shall provide for an index of the use of each module in the system so that the impact of future revisions to the modules can be adequately assessed;
- the procedure shall be conducted in such a way that the overall quality assurance plan is met.

7.4 *System integration phase*

The specific procedures for the hardware/software integration necessarily depend on the nature of the system design and cannot be included in this standard. However, such pro-

cedures shall be established and documented by the integration plan, and shall cover the following activities:

- the acquisition of the proper modules using the library and procedures of Sub-clause 7.3;
- the integration of the hardware modules into the system (e.g. module position, memory address, selection, interconnection wiring);
- the linkage of software modules and the loading of the software into the hardware;
- the preliminary functional test of the integrated system function (see requirements below);
- the documentation of the integration process and the system configuration that will be subjected to verification testing;
- the formal release of the integrated system to verification testing.

The testing performed in this phase of the hardware/software integration is the designer's functional test of the system. It is the system analogy to the debugging done by a programmer before he releases his software to be verified. Any errors detected during the test that are strictly mistakes in the integration itself, and do not affect any project document, may be corrected without formal error report. However, if the resolution of any error requires a change to verified software or any design document, that error shall be reported according to the procedures established by Sub-clause 7.6.

7.5 *Integrated system verification*

The system verification is the process of determining whether or not the verified hardware and software modules have been properly integrated into the system and that the hardware and software are compatible and perform as a system as required by the integration requirements.

The system shall be as complete as is practical for this testing.

The test cases selected for system verification shall exercise all module interfaces as well as the basic operation of the modules themselves. Justification shall be provided in the system integration plan for the simulation of any part of the system or its interfaces.

Integrated system verification shall be conducted in accordance with a formal test plan. The plan shall identify the tests for each computer system interface requirement. The integrated system test shall be developed and the test results evaluated by individuals who are familiar with the system specification and who did not participate in the development of the system.

The level of independence provided between the designers and the system verifiers shall be sufficient to ensure that all communication between the two parties, whether for clarification or error reporting, is only conducted formally in writing at a level of detail which may be audited by persons not involved in the development of the system.

Equipment used for system verification shall be calibrated. Quality assurance measures shall be established for software tools used for verification, commensurate with the importance of those tools for verification.

7.6 *Error resolution procedures*

Specific procedures for the reporting and resolution of errors shall be prepared as a part of the system integration plan.

These procedures shall apply to all errors found during the system verification as well as those found during the integration functional test that require changes to verified software or system design documents. The procedure shall ensure that any required reverification of

hardware or software modules is performed and that the revision to the modules is carried out through the library procedures of Sub-clause 7.3.

An evaluation of each error reported shall be made to determine whether the error was of such a nature that it should have been detected at an earlier phase (such as module testing) of the verification. If this is found to be the case, then an investigation of that earlier stage of the verification shall be conducted to determine whether any systematic deficiency of the verification exists.

7.7 *Integrated system verification test report*

The results of the integrated system verification shall be documented in a test report (F.M5). This report shall identify the hardware and software used, the test equipment used and its calibration, the simulation of system or interface components, and any test results discrepancy found along with the corrective actions taken according to Sub-clause 7.6. The test results shall be retained in a form that is auditable by persons not directly engaged in the test program.

The resolution of all reported errors and the results of the subsequent evaluation shall be documented in sufficient detail and in a manner that is auditable by persons not directly engaged in the system development and verification program.

8. **Computer system validation**

Testing shall be performed to validate the hardware and software as a system in accordance with the safety systems requirements to be satisfied by the computer system. The computer system validation shall be conducted in accordance with a formal validation plan (F.M5). The plan shall identify the validation for static and dynamic cases.

The computer system shall be exercised by static and dynamic simulation of input signals present during normal operation, anticipated operational occurrences and accident conditions requiring computer system action. Each reactor safety function should be confirmed by representative tests of each trip or protection parameter singly and in combination.

It is recommended that the tests:

- cover all signal ranges, and the ranges of computed or calculated parameters in a fully representative manner;
- cover the voting and other logic and logic combinations comprehensively;
- be made for all trip or protective signals in the final assembly configuration;
- ensure that accuracy and response times are confirmed, and that correct action is taken for any equipment failure or failure combination.

In addition, the required input signals and their values, the anticipated output signals and the acceptance criteria shall be stated in the validation plan. The computer system validation plan shall be developed and the results of the validation evaluated by individuals who did not participate in the development phase of Clause 5.

Equipment used for validation shall be calibrated. Hardware and software tools used for this validation need no special verification. They should, however, be shown to be suited to their purpose.

8.1 *Computer system validation report*

The results of the computer system validation shall be documented in a report (F.M6). This report shall identify the hardware and software used, the equipment used and its calibration, the simulation models used, and any discrepancies and corrective actions according to the modification procedure of Clause 9.

The report shall summarize the results of the computer system validation and shall assess the system compliance with all requirements.

The results shall be retained in a form that is auditable by persons not directly engaged in the validation. Software tools used in the validation process should be identified as an item in the validation report. Simulations of the plant and its systems used for the validation shall be documented.

9. Maintenance and modification

A formal modification control procedure shall be established including verification and validation.

9.1 *Modification request procedure*

For a modification to be considered, the following procedure shall be followed.

9.1.1 A software modification request shall be generated, and uniquely identified stating:

- reason for request;
- aim;
- functional scope;
- originator.

The modification may be requested during the development phase because of a change of functional requirements caused by:

- functional modification;
- technological evolution;
- changes of operating conditions.

The modification may be requested because of an anomaly report as a result of tests. Modification for these reasons may cause a change in the software requirements document. The modification may also be requested after delivery; it is then considered to be maintenance.

9.1.2 The modification request shall be evaluated independently, the result being either:

- to reject the request; in this case it is sent back with comments;
- to approve a request of minor importance and impact immediately if it is made during initial development after analysis;
- to require a detailed analysis resulting in a software modification analysis report. This report shall be written by software personnel knowledgeable in the system software;
- to accept the request.

9.1.3 The following items shall be examined in the evaluation of the modification request:

- technical feasibility;
- impact upon hardware (e.g. memory extension) or upon other equipment (e.g. test systems) in which case the request for modification addressing this impact area must be documented for the equipment;
- impact upon software including a list of affected modules;

- impact upon performance (including speed, accuracy, etc.);
- necessary effort for verification and validation; the analysis of the software reverification needed shall be documented in an auditable form;
- the set of documents to be reviewed.

9.1.4 The software modification request is pending until the decision is made:

- to accept it (immediately, or after examination of the software modification analysis report) and to execute it, or
- to reject it and justify the rejection.

9.2 *Procedure for executing a modification*

9.2.1 For modifications on site, the software supplier should have access to a test configuration which is identical to the real system in all relevant aspects (including installed machine, translator, testing tools, plant simulator, etc.) to ensure the validity of the modifications.

9.2.2 A modification procedure depends upon the level at which it is introduced:

- for a change of the software functional specification, the whole software development process for any part of the system impacted by the change shall be re-examined;
- a change during the development phase shall be reviewed in terms of its potential impact upon corresponding lower levels;
- the modification shall be carried out according to the rules given in Clause 5.

9.2.3 After implementation of the modification, the whole or part of the verification and validation process described in Sub-clause 6.1 and Clause 8, shall be performed again according to the software modification analysis (see Sub-clause 9.1.3).

9.2.4 All the documents affected by the modification shall be corrected and refer to the identification of the software modification request. A software modification report shall sum up all the actions made for modification purposes.

All these documents shall be dated, numbered and filed in the software modification control history for the project.

9.3 *Maintenance*

The reasons for maintenance include:

- an anomaly report;
- a functional requirements change after delivery;
- technological evolution;
- a change in operating conditions.

9.3.1 In case of an anomaly, an anomaly report shall be written giving the symptoms, the system environment and system status at the time at which the anomaly was discovered and the suspected causes. Anomaly correction requires the generation of a software modification request, the execution of which shall follow the procedure described in Sub-clause 9.1.

9.3.2 In case of a change in software functional requirements or in operating conditions, the whole software development process shall be re-examined for that part of the system impacted by the change.

9.3.3 Any new hardware requirements and capabilities shall be examined with respect to their potential impact on the software systems. This evaluation should include all hardware considerations reviewed in the original software design. If it can be shown that the new system