UL 1998

STANDARD FOR SAFETY

Software in Programmable Components

UL Standard for Safety for Software in Programmable Components, UL 1998

Third Edition, Dated December 18, 2013

Summary of Topics

This revision of ANSI/UL 1998 dated October 30, 2026 includes the following:

Incorporating Inclusive Language; 2.1A, 2.29A, 2.33, 6.10, Appendix B1.1

Text that has been changed in any manner or impacted by UL's electronic publishing system is marked with a vertical line in the margin.

The new and revised requirements are substantially in accordance with Proposal(s) on this subject dated August 21, 2026.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form by any means, electronic, mechanical photocopying, recording, or otherwise without prior permission of UL.

UL provides this Standard "as is" without warranty of any kind, either expressed or implied, including but not limited to, the implied warranties of merchantability or fitness for any purpose.

In no event will UL be liable for any special, incidental, consequential, indirect or similar damages, including loss of profits, lost savings, loss of data, or any other damages arising out of the use of or the inability to use this Standard, even if UL or an authorized UL representative has been advised of the possibility of such damage. In no event shall UL's liability for any damage ever exceed the price paid for this Standard, regardless of the form of the claim.

Users of the electronic versions of UL's Standards for Safety agree to defend, indemnify, and hold UL harmless from and against any loss, expense, liability, damage, claim, or judgment (including reasonable attorney's fees) resulting from any error or deviation introduced while purchaser is storing an electronic Standard on the purchaser's computer system.

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

No Text on This Page

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

December 18, 2013
(Title Page Reprinted: October 30, 2026)

ANSI/UL 1998-2026

UL 1998

Standard For Software in Programmable Components

First Edition – January, 1994
Second Edition – May, 1998

Third Edition

December 18, 2013

This ANSI/UL Standard for Safety consists of the Third Edition including revisions through October 30, 2026.

The most recent designation of ANSI/UL 1998 as an American National Standard (ANSI) occurred on October 30, 2026. ANSI approval for a standard does not include the Cover Page, Transmittal Pages, and Title Page.

Comments or proposals for revisions on any part of the Standard may be submitted to UL at any time. Proposals should be submitted via a Proposal Request in UL's On-Line Collaborative Standards Development System (CSDS) at https://csds.ul.com.

Our Standards for Safety are copyrighted by ULSE Inc. Neither a printed nor electronic copy of a Standard should be altered in any way. All of our Standards and all copyrights, ownerships, and rights regarding those Standards shall remain the sole and exclusive property of ULSE Inc.

©2026 UL All rights reserved.

No Text on This Page

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

Contents

1 Scope..... 5

2 Definitions of Terms Used..... 5

3 Risk Analysis..... 9

4 Process Definition..... 10

5 Qualification of Design, Implementation, and Verification Tools..... 10

6 Software Design..... 11

7 Critical and Supervisory Sections of Software..... 11

8 Measures To Address Microelectronic Hardware Failure Modes..... 12

9 Product Interface..... 13

10 User Interfaces..... 13

11 Software Analysis and Testing..... 14

11.1 Software analysis..... 14
11.2 Software testing..... 14
11.3 Failure mode and stress testing..... 15

12 Documentation..... 15

12.1 User documentation..... 15
12.2 Software plan..... 16
12.3 Risk analysis approach and results..... 16
12.4 Configuration management plan..... 16
12.5 Programmable system architecture..... 16
12.6 Programmable component and software requirements specification..... 16
12.7 Software design documentation..... 17
12.8 Analysis and test documentation..... 17

13 Off-the-Shelf (OTS) Software..... 17

14 Software Changes and Document Control..... 18

15 Identification..... 18

APPENDIX A (INFORMATIVE) – EXAMPLES OF MEASURES TO ADDRESS MICROELECTRONIC HARDWARE FAILURE MODES..... 19

A1 Scope..... 19
A2 Examples of Acceptable Measures for Microelectronic Hardware Failure Modes..... 19
A3 Software Classes..... 25
A4 Description of Fault Models..... 26
A5 Description of System Structures..... 26

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

A6 Example of the Application of Table A2.1 26
A7 Descriptions of Acceptable Measures for Providing the Required Fault/Error Coverage Specified in Table A2.1 27
A7.1 Descriptions of fault/error control techniques 27
A7.2 Description of memory tests 29
A7.3 Word protection 30

APPENDIX B (Informative) – EXAMPLES OF MEASURES TO ADDRESS SOFTWARE FAILURE MODES 31

B1 General 31

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

1 Scope

1.1 These requirements apply to non-networked embedded software residing in programmable components performing safety-related functions whose failure is capable of resulting in a risk of fire, electric shock, or injury to persons.

1.2 This is a reference standard in which the requirements are to be applied when specifically referenced by other standards or product safety requirements.

1.3 These requirements address the risks unique to product hardware controlled by software in programmable components.

1.4 These requirements are intended to supplement applicable product or component standards and requirements, and are not intended to serve as the sole basis for investigating the risk of fire, electric shock, or injury to persons.

1.5 These requirements are intended to address risks that occur systematically or randomly in the software or in the process used to develop and maintain the software, such as the following:

a) Requirements conversion faults that cause differences between the specification for the programmable component and the software design;

b) Design faults such as incorrect software algorithms or interfaces;

c) Coding faults, including syntax, incorrect signs, endless loops, and other coding faults;

d) Timing faults that cause program execution to occur prematurely or late;

e) Microelectronic memory faults, such as memory failure, not enough memory, or memory overlap;

f) Induced faults caused by microelectronic hardware failure;

g) Latent, user, input/output, range, and other faults that are only detectable when a given state occurs;

h) Failure of the programmable component to perform any function at all; and

i) Failures in data communication such as transmission errors, repetitions, deletion, insertion, resequencing, corruption, delay and masquerade.

1.6 Product standard requirements may amend or supersede the requirements in this standard, as appropriate.

1.7 These requirements are not intended to address risk of remote software update and cybersecurity. Risks associated with unauthorized access or attack through a network shall be addressed in the product standard or other referenced standards such as the Standard for Remote Software Updates, UL 5500 and the series of Standards for Software Cybersecurity for Network-Connectable Products, UL 2900.

2 Definitions of Terms Used

2.1 For the purpose of this standard, the following definitions apply.

2.1A ABNORMAL – Whenever the normal operating path cannot be taken.

2.2 APPLICATION-SPECIFIC INTEGRATED CIRCUIT (ASIC) – An electronic device comprised of many transistors and other semiconductor components which integrate standard cells and arrays from a library into one piece of silicon intended for a particular use.

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

2.3 BUILT-IN TEST – A design method that allows a product to test itself by adding logic for test signal generation and analysis of test results.

2.4 CENTRAL PROCESSING UNIT (CPU) – The unit of a computing and controlling system that includes the circuits controlling the interpretation of instructions and their execution.

2.5 CRITICAL SECTION – A segment of the software that is intended to perform the functions that address or control risks.

2.6 DATA – A representation of facts, concepts, or instructions in a manner suitable for storage, communication, interpretation, or processing.

2.7 DESIGN – The process of defining the software architecture, components, modules, interfaces, test approach, and data for a software system to satisfy specified requirements.

2.8 ELECTRONICALLY ERASABLE PROGRAMMABLE READ ONLY MEMORY (EEPROM) – A reprogrammable read-only memory in which cells may be erased electrically and in which each cell is capable of being reprogrammed electrically.

2.9 EMBEDDED SOFTWARE – Software that is physically part of a product and whose primary purpose is to maintain some property or relationship between other components of the product in order to achieve the overall system objective.

2.10 ERASABLE PROGRAMMABLE READ ONLY MEMORY (EPROM) – A type of programmable memory device which can only be read and not altered under normal use. The memory is capable of being erased by ultraviolet light and reprogrammed.

2.11 ERROR – A discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.

2.12 FAIL-OPERATIONAL PROCEDURE – A procedure executed in the event that a failure has occurred which continues product operation but provides degraded performance or reduced functional capabilities.

2.13 FAIL-SAFE PROCEDURE – A procedure executed to maintain the Risks Addressed (RA) state of a product while transitioning into a non-operational mode.

2.14 FAILURE – The inability of a product or component to perform its specified function.

2.15 FAILURE MODE – The physical or functional manifestation of a failure.

2.16 FAILURE MODE TEST – A suite of tests that have been specifically developed based upon the failure modes that exist in a programmable component or product.

2.17 FAULT – A deficiency in a product or component which is capable of, under some operational conditions, contributing to a failure.

2.18 FAULT-TOLERANT – The capability of software to provide continued correct execution in the presence of a defined set of microelectronic hardware and software faults.

2.19 FLASH MEMORY – A type of non-volatile memory which is capable of being erased electrically and reprogrammed, but only in blocks, as opposed to one byte increments.

2.20 HAZARD – A potential source of physical injury to persons.

2.21 INSTRUCTION – A statement that specifies an operation to be performed and that is capable of identifying data involved in the operation.

2.22 INTEGRITY – The degree to which a system or component prevents unauthorized access to, or modification of, computer programs or data.

2.23 MICROCONTROLLER – A microcomputer chip capable of executing instructions.

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

2.24 MICROELECTRONICS – Monolithic, hybrid, or module circuits, where the internal connections are not accessible, which satisfy one or more of the following criteria:

a) More than 1000 gates are used in digital mode;

b) More than 24 functionally different external electrical connections are available for use; or

c) The functions can be reprogrammed.

The circuits are capable of functioning in the analogue mode, digital mode, or a combination of the two modes. Examples of microelectronics include: ASICs, ROMs, RAMs, PROMs, EPROMs, PALs, and PLDs.

2.25 NEGATIVE CONDITION BRANCH – A code construct implementing an alternate path of program control flow intended to provide a deterministic path from the normal operating path to the Risks Addressed (RA) state whenever the normal operating path cannot be taken.

2.26 NON-DETERMINISM – The state in which output is not predictable in advance because data dependencies effect task execution order.

2.27 NON-NETWORKED EMBEDDED SOFTWARE – Embedded software that executes safety related functionality on a single microprocessor/microcontroller or on microprocessors/microcontrollers residing in the same physical enclosure, which does not depend on data received or transmitted outside of the enclosure.

2.28 NON-TERMINATING STATE – Any state of the software characterized by the execution of instructions that do not permit exiting from that state.

2.29 NON-VOLATILE MEMORY – A storage device not alterable by the interruption of the power to the memory, for example; ROM, FLASH, PROM, EPROM, and EEPROM.

2.29A NORMAL – An expected or supported path or function per the manufacturer's documentation.

2.30 OFF-THE-SHELF (OTS) SOFTWARE – Software which is made available on the market and intended for broad distribution, generally without the need for additional tailoring, including, but not limited to, operating system software, runtime libraries, real-time executives, kernel primitives, shareable re-entrant library-type routines, and the like. This includes software that has been developed by another developer.

2.31 OPERATIONAL TESTING – Evaluations on a product or component using an input profile representative of its operational environment.

2.32 PARAMETER SETTINGS – A finite collection of values assigned to variables to select, enable or disable known pre-existing function(s) or features of the software.

2.33 PARTITIONING – The act of separating the functions of a system into verifiably distinct and protected collections of functions.

2.34 POST-RELEASE TESTING – All testing for changes implemented after the final production software is released.

2.35 PROCEDURE – A course of action taken to perform a task.

2.36 PROCESS – A sequence of steps performed to produce a given result.

2.37 PRODUCT – An instrument, apparatus, implement, or machine intended for personal, household, industrial, laboratory, office, or transportation use.

2.38 PRODUCT HARDWARE – Any hardware that is part of a product and that provides electrical, mechanical, and (or) electromechanical functions.

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

2.39 PROGRAMMABLE COMPONENT – Any microelectronic hardware that can be programmed in the design center, the factory, or in the field. Here the term "programmable" is taken to be "any manner in which one can alter the software wherein the behavior of the component can be altered."

2.40 PROGRAMMABLE COMPONENT CONFIGURATION – The configuration of the microelectronic hardware and software that must be present and functioning for the software to operate and control the equipment as intended. This configuration includes, but is not limited to, the operating system or executive software, communication software, microcontroller, network, input/output hardware, any generic software libraries, database management and user interface software.

2.41 PROGRAMMABLE READ-ONLY MEMORY (PROM) – A memory chip whose contents are capable of being programmed by a user or manufacturer for a specific purpose.

2.42 RANDOM ACCESS MEMORY (RAM) – Data storage where the time required for data access is independent of the location of the data most recently obtained or placed in storage.

2.43 READ ONLY MEMORY (ROM) – Data storage that stores data not alterable by computer instructions.

2.44 RISK – The potential for fire, electric shock, or injury to persons associated with the intended use of the product as specified by the product safety requirements.

2.45 RISKS ADDRESSED (RA) STATE – A state that is characterized by all reasonably foreseeable risks associated with the intended use of the product being addressed such that there is no longer a likelihood of the risk.

2.46 SAFETY-RELATED FUNCTIONS – Control, protection, and monitoring functions which are intended to reduce the risk of fire, electric shock, or injury to persons.

2.47 SOFTWARE – Computer programs, procedures, and data pertaining to the operation of a programmable component that provides safety-related functions as follows:

a) Exercises direct control over the state of microelectronic or product hardware. When not performed, performed out of sequence, or performed incorrectly, such programs, procedures, and data are capable of resulting in a risk.

b) Monitors the state of microelectronic or product hardware. When not performed, performed out of sequence, or performed incorrectly, such programs, procedures, and data provide data that is capable of resulting in a risk.

c) Exercises direct control over the state of the microelectronic or product hardware. When not performed, performed out of sequence, or performed incorrectly, such programs, procedures, and data are capable of, in conjunction with other human actions, product hardware or environmental failure, resulting in a risk.

2.48 SOFTWARE CODE – Computer instructions and data definitions expressed in a programming language or in a form output by an assembler, compiler, or other translator.

2.49 SOFTWARE DESIGN – The process of defining the software architecture, components, modules, interfaces, test approach, and data for a software implementation to satisfy specified requirements.

2.50 STATE – The values assumed, at a given instant in time, by a collection of variables that define certain specified characteristics of the product or its components.

2.51 STRESS TESTING – Testing conducted to evaluate a product or component at or beyond the limits of its specified requirements.

2.52 SUPERVISORY SECTION – The main section of software that controls the execution of the critical and non-critical sections of the software (e.g., the operating system of a microprocessor).

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.

2.52A SYSTEMATIC FAULT – Type of faults which occurs when a certain set of particular input conditions causes a process or product to react in a certain manner. Typical examples are specification errors, design errors and software coding errors.

2.53 TEST CONFIGURATION – The detailed identification and layout of the test and measurement equipment used for testing. The test configuration is often documented directly in the test plan or the test procedures and specifies all special test fixtures, supplementary software, software tools, and data files along with a description of the test set-up and any preparation activities.

2.54 TEST CRITERIA – Objectives that a product or component must meet in order to comply with a given test.

2.55 TEST PARAMETERS – Variables used to alter and define a test.

2.56 TEST PLAN – A document describing the scope, approach, resources, and schedule of intended test activities. It defines test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

2.57 TEST PROCEDURES – Detailed instructions for the set-up, execution, and evaluation of results for a given test case.

2.58 TOOL – Any equipment (e.g., logic analyzers, oscilloscopes, multimeters, digital and analog computers), devices, or software programs [e.g., simulators, computer-aided software/systems engineering (CASE) tools, compilers, type checkers, static analyzers, automated testing scripts, debuggers, linkers, loaders, assemblers, code generators, code librarians, editors, and software analyzers] used to automate or partially automate software development activities, including design, implementation, and testing.

2.59 TOOL QUALIFICATION – The acceptance of analysis, testing, and resulting evidence for which confidence is obtained regarding the correctness of a tool's outputs.

2.60 TOOL VENDOR – The organization responsible for providing the tool.

2.61 UNIQUE IDENTIFIER – A value that distinctively characterizes each individual version or revision of a manufacturer's critical and supervisory sections of software.

2.62 USER – A qualified service person or an operator of the programmable component.

2.63 VALIDATION – The test and evaluation of the integrated computer system (hardware and software) and its specification to determine whether it carries out its intended purpose.

2.64 VERIFICATION – The process of determining whether the products of a given development phase are correct and consistent with the products and standards provided as input to that phase.

3 Risk Analysis

3.1 A risk analysis shall be conducted to determine:

a) The set of risks (for scope of risk analysis see Section 1, Scope); and

b) That the software addresses the identified risks.

3.2 The risk analysis shall be based on the safety requirements for the programmable component.

3.3 An analysis shall be conducted to identify the critical, non-critical, and supervisory sections of the software.

3.4 An analysis shall be conducted to identify states or transitions that are capable of resulting in a risk.

ULSE INC. COPYRIGHTED MATERIAL – NOT AUTHORIZED FOR FURTHER REPRODUCTION OR DISTRIBUTION WITHOUT PERMISSION FROM ULSE INC.