# Results of the IEC 61508 Baseline Safety Case Functional Safety Assessment

**Project: Codethink**

**Trustable Reproducible Linux Operating System**

**Customer: Codethink Limited Manchester, England**

**Contract No.: Q22/08-135**

**Report No.:**

**Q22/08-135 COD 22-08-135 R002**

**Version V1, Revision R3, April 18, 2025**

**Author: Jonathan Moore**

# 1 Purpose and Scope

This document describes the results of the IEC 61508 functional safety assessment of the

**Codethink Limited CTRL OS**

by *exida* according to the accredited *exida* certification scheme, which includes the requirements of IEC 61508 tailored to the scope of the CTRL OS.

The purpose of the assessment was to evaluate the compliance of:

- the systematic capability of CTRL OS for SIL 3 and the derived product safety requirements.

and

- the CTRL OS development processes, procedures and techniques as implemented for the safety-related deliveries with the lifecycle requirements, including the relevant parts of IEC 61508 for SIL 3.

The assessment was carried out by *exida* based on the quality procedures and scope definitions of *exida*.

In this document IEC 61508 refers to IEC 61508:2010.

## 2 Management Summary

This report summarizes the results of the functional safety assessment according to IEC 61508:2010 performed by *exida* for the

**Codethink Limited Codethink Trustable Reproducible Linux Operating System**

which has been using the Trustable Software Framework (TSF) *Process Framework* made open source by Codethink in February 2025. Codethink Trustable Reproducible Linux (CTRL) Operating System (OS) applies the Risk Analysis, Fault Injection and Automation (RAFIA) safety approach also proposed by Codethink.

The functional safety assessment performed by *exida* consisted of the following activities:

- *exida* assessed the Safety Case prepared by Codethink limited against the applicable requirements of IEC 61508.
- *exida* performed detailed audits to assess the applied Codethink limited development processes and the work products generated in the CTRL OS development project for compliance with the requirements of IEC 61508. The assessment was executed using a subset of IEC 61508 requirements tailored to the work scope of the development project.
- *exida* reviewed the CTRL OS Safety Concept and design solutions and assessed their suitability to fulfil the given safety requirements and the requirements of IEC 61508.
- *exida* reviewed and assessed the Safety Analyses prepared by Codethink against the requirements of IEC 61508.

The functional safety assessment was performed to the SIL 3 requirements of the applicable parts of IEC 61508 with experts knowledgeable in ISO 26262 ensuring compatibility of the activities with the specific additional requirements and requests of the automotive industry. The results of the functional safety assessment can be summarized as follows:

Considering the preliminary safety goals proposed for the Operating System, the tools, process framework and software development lifecycle proposed by Codethink the activities planned for CTRL OS satisfy the baseline safety case expectations of the *exida* certification scheme.

The assessment of the process framework as applied to CTRL OS has shown that the relevant safety requirements of IEC 61508 at SIL 3 are met and a process compliance argument is complete with this baseline safety case assessment.

The next step to ensure CTRL OS is capable for use in SIL 3 (and ASIL D) applications when properly designed and implemented into an item is to complete the process according to the TSF framework and submit for final safety case assessment by *exida*.

## 2.1 Tools and Methods used for the Assessment

This assessment was planned and carried out using the *exida* IEC 61508 assessment checklist and assessment status sheet.

The assessment was performed based on the objectives and requirements of IEC 61508. For the fulfilment of the objectives, expectations are defined which builds the acceptance level for the assessment. The expectations are reviewed to verify that each single requirement is covered or classified as not applicable. Because of this methodology, comparable assessments in multiple projects with different assessors are achieved. The arguments for the positive judgment of the assessor are documented and summarized within this report.

The assessment was planned by *exida* and agreed with Codethink (see [E1]). The assessment steps and progress were continuously documented by *exida* (see [E2], [E3]).

# 3 Project Management

## 3.1 *exida*

*exida* is one of the world's leading accredited Certification Bodies and knowledge companies, specializing in automation system safety, availability, and cybersecurity with over 500 person-years of cumulative experience in functional safety. Founded by several of the world's top reliability and safety experts from assessment organizations and manufacturers, *exida* is a global company with offices around the world. *exida* offers training, coaching, project-oriented system consulting services, safety lifecycle engineering tools, detailed product assurance, cyber-security and functional safety certification, and a collection of on-line safety and reliability resources. *exida* maintains a comprehensive failure rate and failure mode database on process equipment based on 350 billion hours of field failure data.

## 3.2 Roles of the Parties Involved

| Party | Role |
|---|---|
| Codethink Limited | Developer and manufacturer of CTRL OS |
| *exida* | Performed the Functional Safety Assessment per the accredited *exida* scheme.<br>Alexander Eggerer – Assessor<br>Jonathan Moore – Co-Assessor |

Codethink Limited contracted *exida* with the IEC 61508 Functional Safety Assessment of the CTRL OS project.

## 3.3 Reference Documents

The services delivered by *exida* were performed based on the following standards and literature:

| ID | Document name and revision | Description |
|---|---|---|
| N1 | IEC 61508-1:2010 | General requirements |
| N2 | IEC 61508-2:2010 | E/E/PES related systems |
| N3 | IEC 61508-3:2010 | Software requirements |

The services delivered by *exida* were performed based on the following customer documentation:

| ID | Document name and revision 2025-02-12-18_26 |
|---|---|
| D1 | Certification Agreement.pdf |
| D2 | certificates/23899-001-Codethink-Limited-23899-Stage-2.pdf |
| D3 | certificates/23899-001-Codethink-Limited-23899-Surveillance.pdf |
| D4 | certificates/23899_Codethink_-*Headline*-_Certificate.pdf |
| D5 | ctrl.html |
| D6 | doorstop/CTRL-ANALYSIS.html |
| D7 | doorstop/CTRL-AOU.html |
| D8 | doorstop/CTRL-ARCH-EVIDENCE-CENTRALIZED.html |
| D9 | doorstop/CTRL-ARCH-EVIDENCE.html |
| D10 | doorstop/CTRL-ARCH-EVIDENCE-MAC.html |
| D11 | doorstop/CTRL-ARCH-EVIDENCE-SAFETY.html |
| D12 | doorstop/CTRL-ARCH-EVIDENCE-SUPERVISED.html |
| D13 | doorstop/CTRL-ARCH-EVIDENCE-SYSTEM.html |
| D14 | doorstop/CTRL-ARCH.html |
| D15 | doorstop/CTRL-CONFIDENCE.html |
| D16 | doorstop/CTRL-CONSTRUCTION.html |
| D17 | doorstop/CTRL-CONTROL_FLOW.html |
| D18 | doorstop/CTRL-DOCUMENTATION.html |
| D19 | doorstop/CTRL-EVIDENCE.html |
| D20 | doorstop/CTRL.html |
| D21 | doorstop/CTRL-ITERATION.html |
| D22 | doorstop/CTRL-MIRRORS.html |
| D23 | doorstop/CTRL-MONITOR.html |
| D24 | doorstop/CTRL-MONITOR-Performance.html |
| D25 | doorstop/CTRL-MONITOR-Robustness.html |
| D26 | doorstop/CTRL-MONITOR-systematic_ability.html |
| D27 | doorstop/CTRL-PROCESS.html |
| D28 | doorstop/CTRL-PROCESS-METRICS.html |
| D29 | doorstop/CTRL-SCHEDULING.html |

| ID | Document name and revision 2025-02-12-18_26 |
|---|---|
| D30 | doorstop/CTRL-STPA-CLAIMS.html |
| D31 | doorstop/CTRL-STPA-REQUESTS.html |
| D32 | doorstop/CTRL-TEST-SPEC.html |
| D33 | doorstop/DA.html |
| D34 | doorstop/DCS.html |
| D35 | doorstop/DMS.html |
| D36 | doorstop/IEC-61508-ASSERTIONS.html |
| D37 | doorstop/IEC-61508.html |
| D38 | doorstop/IEC-61508-SAFETY-PROPERTIES.html |
| D39 | doorstop/index.html |
| D40 | doorstop/RAFIA.html |
| D41 | doorstop/RAFIA-TESTING.html |
| D42 | doorstop/RE.html |
| D43 | doorstop/REL.html |
| D44 | doorstop/SIF-CHANGES.html |
| D45 | doorstop/SIF-CONSTRUCTION.html |
| D46 | doorstop/SIF-ENVIRONMENT.html |
| D47 | doorstop/SIF.html |
| D48 | doorstop/SIF-INFRA.html |
| D49 | doorstop/SIF-PROVENANCE.html |
| D50 | doorstop/SIF-TOOLS.html |
| D51 | doorstop/STPA.html |
| D52 | doorstop/TA.html |
| D53 | doorstop/TRUSTABLE.html |
| D54 | doorstop/trustable_report_for_CTRL.html |
| D55 | doorstop/TT.html |
| D56 | index.html |
| D57 | maintainer/downstream-changes.html |
| D58 | maintainer/local-linting.html |
| D59 | maintainer/process/bug-management.html |
| D60 | maintainer/process/change-management.html |

| ID | Document name and revision 2025-02-12-18_26 |
|---|---|
| D61 | maintainer/process/config-management.html |
| D62 | maintainer/process/GitLabci.html |
| D63 | maintainer/process/personnel.html |
| D64 | maintainer/process/project-plan.html |
| D65 | maintainer/process/requirements-management.html |
| D66 | maintainer/releasing.html |
| D67 | maintainer/style-guidelines.html |
| D68 | maintainer/test/specs-applications.html |
| D69 | maintainer/test/specs-boot.html |
| D70 | maintainer/test/specs-isolation.html |
| D71 | maintainer/test/specs-peripherals.html |
| D72 | maintainer/test/specs-scheduling.html |
| D73 | maintainer/test/specs-storage.html |
| D74 | portal.html |
| D75 | safety/index.html |
| D76 | safety/review.html |
| D77 | safety/structure.html |
| D78 | trustable/index.html |
| D79 | trustable/results.html |
| D80 | trustable/structure.html |
| D81 | user/apparmor.html |
| D82 | user/architecture.html |
| D83 | user/building-and-running.html |
| D84 | user/cgroup-configuration.html |
| D85 | user/critical-applications.html |
| D86 | user/deadline-scheduling.html |
| D87 | user/deriving.html |
| D88 | user/external-verification.html |
| D89 | user/initial-configuration.html |
| D90 | user/kernel-config.html |
| D91 | user/non-critical-applications.html |

| ID | Document name and revision 2025-02-12-18_26 |
|---|---|
| D92 | user/safety-manual.html |
| D93 | user/safety-monitor.html |
| D94 | user/sandboxing.html |
| D95 | user/test-file-spec.html |
| D96 | user/testing.html |
| D97 | user/troubleshooting/build-shell-fails-to-start.html |
| D98 | user/troubleshooting/buildstream-fails-to-run.html |
| D99 | user/troubleshooting/build-takes-too-long.html |
| D100 | user/troubleshooting/dependency-is-missing.html |
| D101 | user/user-configuration.html |

The functional safety assessment performed by *exida* is documented in the following documents:

| ID | Document name and revision | Description |
|---|---|---|
| E1 | Codethink Inc. Q2308-135 Certification and Advisory Services.pdf (14-Dec-2023) | IEC 61508 Functional Safety Assessment Plan |
| E2 | COD 22-08-135 SC02 V1R5 IEC 61508 Assessment Checklist | IEC 61508 Functional Safety Assessment Checklist and Review Comments |
| E3 | COD 22-08-135 SS01 V1R1 IEC 61508 Assessment Status Sheet | IEC 61508 Functional Safety Assessment Status Sheet and recommended actions/ next steps |

## 3.4 Assessment Approach

The functional safety assessment was closely driven by requirements of the *exida* scheme which includes subsets filtered from IEC 61508. This means that functional safety related requirements were grouped based on their objectives. The judgement about the fulfilment of the objectives was done based on the IEC 61508 requirements and the work products which were subject to the assessment [E2].

The Codethink project teams, not individual people, were audited.

## 3.5 Assessment Schedule

The functional safety assessment was performed over a series of face-2-face meetings and offline as follows:

| Date | Assessment review meeting |
|---|---|
| August 2023 | F2F assessment Manchester |
| February 2024 | F2F assessment Munich |
| February 2025 | F2F assessment Munich |

## 3.6 Product Modifications

The modification process has been assessed and audited, so modifications are allowed by this assessment. Modified versions of the CTRL OS should follow the Codethink modification process, don’t require reassessment, and will be audited at surveillance audits.

## 4 Product Description

The scope of the certification is Codethink's Trustable and Reproducible Linux-based operating system, which supports the execution of safety-related software by its listed safety features, where the listed safety features are tracked with Trustable Software Framework (TSF) as follows:

CTRL 1: CTRL provides a mechanism for real-time scheduling of Critical Processes.

CTRL 2: CTRL provides a mechanism to detect time budget violations for Critical Processes.

CTRL 3: CTRL provides a mechanism to detect deadline violations for Critical Processes.

CTRL 4: CTRL provides a mechanism to terminate processes.

### 4.1 Hardware and Software Version Numbers

Not relevant for process baseline safety case assessment. This baseline assessment was based primarily on a snapshot of the continuous documentation. This snapshot is dated 2025-02-12-18_26.

# 5 Results of the Functional Safety Assessment

## 5.1 Functional Safety Management (FSM)

This chapter summarizes the assessment results related to the objectives and requirements of IEC 61508, as far as they are applicable to the CTRL OS project and within the defined scope of this assessment.

### 5.1.1 Overall Safety Management

**Objective**

The objective of the IEC 61508 standard is to define the requirements for the organizations that are responsible for the safety lifecycle or perform safety activities in the safety lifecycle.

#### 5.1.1.1 Personnel Plan & Personnel Competency

Key project staff are defined and the competencies and competency requirements documented.

#### 5.1.1.2 Impact Analysis for Modification and Action Item follow up

Actions are raised as issues to the relevant GitLab repositories and tracked to closure. All merge requests are reviewed and verified before merge. Any change will reflect an invalidation of dependent expectations which need to be reviewed. Impacts are explicitly highlighted.

#### 5.1.1.3 Safety Lifecycle

RAFIA processes are cyclic with built in continuous improvement. In place of a V-model a continuous development process is implemented.

#### 5.1.1.4 Documentation Management

All documentation exists in the version control repository in mostly human readable form (Markdown / YAML etc.).

**Conclusion**

The intelligent use of software tools and continuous merge request and merge review process satisfies the requirements of IEC 61508. Requirements from 1/6, 1/7, 2/7, 3/6, 3/7 and 3/8 are met.

### 5.1.2 Configuration Management (CM)

**Objective**

#### 5.1.2.1 Control & Identification

Change Authorization, Version Control and Configuration are all well established and enforced via GitLab and GitLab Project Configurator (GPC):

- Change of a configuration can only be applied after a review of the merge request (everything is handled as code) and the successful pass of pre-merge tests ([3/6.2.3.a]).
- Only one configuration exists within the project (main branch) to apply changes. Previous versions can be identified through tags. New merge requests are applied on the development branch; the customer therefore must use the latest version of the product (main branch), or a tagged version associated with a release.
- Authorization is managed through roles and groups within GitLab.

All product configurations, including input source code, are controlled and documented. All product artefacts are uniquely identified and traceable to their inputs.

- The product being assessed is identified through the project unique branch identification.
- Configuration consists of the toolchain and the CTRL OS - Scheduler configuration.
- Any configuration can be rebuilt as everything is under version control.
- Every artefact (work products, tools, code, reviews, verification, etc.) is handled as code.

- Merge requests need to pass the pre-merge tests and a review first before being integrated into GitLab source control.

- Every artefact is linked to an expectation (requirement).

#### 5.1.2.2 Release

Release notes are added for each release containing version number, change records, and open issues.

**Conclusion**

Applicable parts of requirements in IEC 61508 1/6 and 3/6 are fulfilled.

### 5.1.3 Modification Process (MOD)

**Objective**

#### 5.1.3.1 Initiation, authorization, impact analysis and recording

A modification procedure exists which identifies how a modification request is undertaken via GitLab Issues.

- Hardware is out of scope

A modification procedure requires all modifications to the CTRL product to be undertaken via GitLab, with discussion, impact analysis and resolution tracking via GitLab issues.

The modification procedure calls for discussion, with resolution tracking in GitLab ensuring the development progresses through appropriate phases before acceptance.

**Conclusion**

IEC 61508 clauses 1/7, 2/7, 3/7 and 3/A have been considered, and the process objectives are achieved.

### 5.1.4 System Architecture Design (SAD)

**Objective**

#### 5.1.4.1 Partitioning, design, functionality and interfaces

The RAFIA workflow includes a safety analysis that partitions components. All documents (code, architectural documents, etc.) are reviewed with established change management processes. All safety feature specifications derived from safety analysis are tracked via expectations. Safety related interfaces are identified with RAFIA.

#### 5.1.4.2 Maintainability and Testing

Advanced warning indicators for misbehaviours are identified and monitored for enabling maintainability and testability is ensured via testing tracked via expectations derived from RAFIA.

**Conclusion**

IEC clauses 1/5, 1/7, 2/7, 2/A, 2/B, 2/E and 3/7 are evident in the lifecycle described in RAFIA.

### 5.1.5 Safety Requirements Specification (SRS) & Software Architecture (SWA)

**Objective**

#### 5.1.5.1 Methods, Safety Functions, DTI, Independence

The SRS goes through peer review, with review meetings. The results of the review are documented in GitLab, and all action items are tracked through resolution.

Safety functions are derived from linked safety analysis.

Software requirements include requirements for self-monitoring of program flow and data flow. All requirements are available to the entire organization. Expectations are derived from high-level safety functions + STPA results and include some assumptions of use (AoU) on the applications.

### 5.1.6 Software Detailed Design (SWD) & Coding Standard (CS)

**Objective**

#### 5.1.6.1 Features for safety, Warning, Development Environment

The Expectations that maintain safety integrity of data are identified during safety analysis

CTRL OS - Scheduler is monitored by the Safety Monitor

Misbehaviour mitigations that rely on diagnostics are identified and verified (through Safety Monitor)

Misbehaviour mitigations are derived from safety analysis that consider hardware faults

- Suitability of hardware timer is checked externally and through watchdog

An Integrated Development Environment for software development and debugging is used to develop the software.

- Codethink Trustable Software Framework (TSF) is used

#### 5.1.6.2 Coding Standard

Source code standards are chosen as appropriate per language per context that are then implemented as CI tests. This choice is then maintained via randomized quality control checks.

- Coding standards are defined by upstream teams (e.g. Linux Kernel Team).
- Coding standard exists for Safety Monitor.
- Programming languages used include C (Scheduler, Toolchain), Python (Toolchain), and Rust (Safety Monitor).

**Conclusion**

IEC 61508 requirements in 3/7, 3/A and 3/B relevant to SWD and CS are evident.

### 5.1.7 Software Implementation (SWI)

**Objective**

#### 5.1.7.1 Programming Language, Automation, Static Analysis

CTRL OS implementation is done with suitable configuration languages to meet raised expectations

- Kernel and Glibc use C, but only kernel uses a meaningful subset.
- Safety Monitor is written in Rust.
- To address weaknesses in Glibc: specification is very stable, essentially POSIX since 80s. Little feature development done here either.

Notation used for any structured methods are represented diagrammatically with a legend when necessary

Safety analysis and resulting expectations and linked evidence are tracked with TSF

- A semi-formal method for diagramming the interactions between interfaces of software elements exists (but not the internals of elements).

No implicit trust in software is assumed. The approach assumes any software module may be compromised, so tests are repeated on every merged change request.

- Any binaries in the kernel would lower the trustable score

All tests are applied automatically via GitLab CI

Static analysis results are generated as appropriate during CI tests.

Validation sufficiency is established using confidence measurements. The assigned validation confidence is then determined via the RAFIA workflow, applying the statistical approach where appropriate.

All source code is subject to review and approval via GitLab Merge request

- Upstream reviews are performed by maintainers; second review by Freedesktop SDK maintainers, third review by Codethink

All interfaces and related sources are maintained in fully controlled environments used for validation. These are made available for each iteration, along with the corresponding expectations that define them.

Coding standards are enforced via CI.

During code reviews standards will be looked at

Module Test results become part of the trustable score. Adequacy of test results is established in the confidence score.

All changes are subject to established change management processes.

**Conclusion**

Applicable clauses in IEC 61508 1/5, 2/7, 3/7, 3/A and 3/B have been reviewed.

### 5.1.8 Safety Manual (SM)

The Safety Manual is managed alongside the other artefacts in the repository and subject to review, merge request, and reviewed as part of merge acceptance.

**Objective**

#### 5.1.8.1 Software Functions and Configuration

The safety manual describes functions, input and output interfaces via AoUs.

- The safety manual is created out of expectations.

All component versions are traceable to sources within fully controlled version control systems.

- Configuration is stored in GitLab as being considered source.
- Source code will be branched once released and each branch name is unique.

#### 5.1.8.2 Instructions

CTRL OS may only be integrated via competent parties.

- The CTRL OS - Scheduler is part of the kernel which is integrated upstream.

All iterations come with instructions and attestations.

Change requests are tracked via GitLab Issues

- The integration is done within the Factory on the main branch.

The Safety Manual includes a description of the design safe state.

- The CTRL OS - Scheduler is guarded by a watchdog which is triggered by the safety monitor.

**Conclusion**

Relevant parts of the clauses in IEC 61508 have been checked. 1/7, 2/7, 2/B. 2/D, 3/7, 3/D.

### 5.1.9 Tool Qualification (TOOLS) and Tool Validation (TVAL)

**Objective**

#### 5.1.9.1 Classification and Failure Analysis

Online tools are treated as components with Expectations from Safety Analysis.

- Every tool is treated as a tool within TSF.

Tools are identified via the toolchain generation process; the resulting toolchain runs in an environment without internet access. CI tools with internet access are explicitly documented.

External databases (e.g., NIST) may trigger the creation of new bug tickets (e.g., CVE). This has an impact on the trustable score.

- GCC is classified as T3

All tool documentation is available in Codethink mirrored sources

Tools impacting system safety have mitigations recorded as Expectations from Safety Analysis

Configuration changes are managed through change control process

- reviews are part of the change control

Tool Expectations link to evidence as part of the RAFIA process.

**Conclusion**

Treating tools the same as source code ensures the behaviours of the tools can be trusted. The requirement in IEC 61508 3/7 and 3/A will be achieved.

### 5.1.10 Integration and Validation Test Planning and Execution (ITP/VTP/ITE/VTE)

**Objective**

Test and validation activities already outlined in the various CIs have been reviewed and confirmed sufficient and suitable to be able to satisfy the requirements of IEC 61508. Specific test results and scores have not been assessed.

**Conclusion**

Test Planning and Execution activities are compliant.

### 5.1.11 Traceability (TRA)

**Objective**

#### 5.1.11.1 Traceability

Safety related feature tests provide confidence score visible in the trustable report

- Trustable report will show the validation test reports down to each element.
- TSF will ensure traceability
- Every review is stored in GitLab. No need for a report as a review needs to be performed by process for every change
- Forward traceability is tracked via TSF
- Backward traceability only via GitLab
- Consider including TRA-3 to TRA-56 to the 61508-compliance argumentation.

**Conclusion**

Clauses in 1/5, 2/7 and 3/A are satisfied.

# 6 Terms and Definitions

The following abbreviations are used throughout this document:

| Abbreviation | Description |
|---|---|
| AoU | Assumption-of-Use |
| ASIL | Automotive Safety Integrity Level |
| CM | Configuration Management |
| CR | Change Request |
| FFI | Freedom from Interference |
| FMEA | Failure Mode and Effect Analysis |
| FMEDA | Failure Mode, Effect and Diagnostic Analysis |
| FSC | Functional Safety Concept |
| FSM | Functional Safety Manager / Management |
| FSR | Functional Safety Requirement |
| FTTI | Fault Tolerance Time Interval |
| IEC | International Electrotechnical Commission |
| ISO | International Organization for Standardization |
| SG | Safety Goal |
| SM | Safety Manual |
| SR | Safety Requirement |
| SW | Software |
| SWSR | Software Safety Requirement |
| SYS | System |
| TLSR | Top-Level Safety Requirement |
| TLSS | Top-Level Safe State |
| V&V | Verification and Validation |
| WP | Work Product |
| Process context | Process context is the OS environment used when executing a given process. |
| TSF | Trustable Software Framework |
| Upstream | In open source, upstream is the source repository and project where contributions happen, and releases are made. The contributions flow from upstream to downstream. |

| Abbreviation | Description |
|---|---|
| User Mode | A restricted mode of execution used when executing application code. In user mode, the CPU has limited access to specific hardware (through drivers only protected by access control) and memory resources. |
| Kernel Mode | A supervisor-level privileged mode of execution used when executing kernel code. In kernel mode, the CPU has full and unrestricted access to all hardware and memory resources. |
| Interrupt Context | The OS environment entered when a hardware interrupt occurs. This context is not associated with any process in the system. See also: Process context. |

# 7 Status of the Document

## 7.1 Liability

*exida* prepares reports based on methods and failure modes advocated in international standards. Failure rates are obtained from a collection of industrial databases. *exida* accepts no liability whatsoever for the use of these numbers or for the correctness of the standards on which the general calculation methods are based.

## 7.2 Revision History

| Contract Number | Report Number | Revision Notes |
|---|---|---|
| Q22/08-135 | COD 22-08-135 R001 V1R3 | Final; Jonathan Moore; 4/18/2025 |
| Q22/08-135 | COD 22-08-135 R001 V1R2 | Draft; Jonathan Moore; 3/25/2025 |
| Q22/08-135 | COD 22-08-135 R001 V1R1 | Draft; Jonathan Moore; 3/20/2025 |

Review: Alexander Eggerer

Status: Released; 4/18/2025

THIS IS THE FINAL PAGE