# MICROSAR Safe

Safety Manual

# 1 General Part

## 1.1 Introduction

### 1.1.1 Purpose

This document describes the assumptions made by Vector during the development of MICROSAR Safe [2]. This document provides information on how to integrate MICROSAR Safe into your safety-related project.

This document is intended for the use by the user of MICROSAR Safe. It should be read by project managers, safety managers, and engineers to allow proper integration of MICROSAR Safe.

### 1.1.2 Scope

This document covers aspects of components that are marked with an ASIL in the delivery description provided by Vector. Neither QM Vector components, nor components by other venders are in the scope of this document.

### 1.1.3 Definitions

| Term | Definition |
|---|---|
| User of MICROSAR Safe | Integrator of components provided by Vector. |
| MICROSAR Safe | MICROSAR Safe comprises MICROSAR SafeBSW and MICROSAR SafeRTE as Safety Element out of Context. MICROSAR SafeBSW is a set of components, that are developed according to ISO 26262 [1], and are provided by Vector in the context of this delivery. The list of MICROSAR Safe components in this delivery can be taken from the documentation of the delivery. |
| Critical section | A section of code that needs |
| Configuration data | TBD |
| Generated code | Source code that is generated as a result of the configuration in DaVinci Configurator Pro |
| Partition | A set of memory regions that is accessible by tasks and ISRs. Synonym to OSApplication. |

The words *shall*, *shall not*, *should*, *can* in this document are to be interpreted as described here:

1. *Shall* means that the definition is an absolute requirement of the specification.
2. *Shall not* means that the definition is an absolute prohibition of the specification.
3. *Should* means that there may exist valid reasons in particular circumstances to ignore a particular definition, but the full implications must be understood and carefully weighed before choosing a different course.
4. *Can* means that a definition is truly optional.

The user of MICROSAR Safe can deviate from all constraints and requirements in this Safety Manual in the responsibility of the user of MICROSAR Safe, if equivalent measures are used.
If a measure is equivalent can be decided in the responsibility of the user of MICROSAR Safe.

### 1.1.4 References

| No. | Source | Title | Version |
|---|---|---|---|
| [1] | ISO | ISO 26262 Road vehicles — Functional safety (all parts) | 2011/2012 |
| [2] | Vector | ISO 26262 Compliance Documentation | TBD |
| [3] | Vector | Product Information MICROSAR Safe | TBD |
| [4] | Vector | Technical Reference MICROSAR Safe Silence Verifier | 1.4 |

### 1.1.5 Overview

This document is automatically generated. The content of this document depends on the components and micro-controller of your delivery.

The structure of this document comprises:

> ▪ a general section that covers all assumptions and constraints that are always applicable, and
> ▪ a micro-controller specific section that covers all aspects of the selected micro-controller (only if micro-controller specific components are part of the delivery), and
> ▪ a section for each component that covers its constraints and necessary verification steps.

Vector's assumptions on the environment of the MICROSAR Safe components as well as the integration process are described.

## 1.2 Concept

MICROSAR Safe comprises a set of components developed according to ISO 26262. These components can be combined - together with other measures - to build a safe system according to ISO 26262.

### 1.2.1 Assumed technical safety requirements

### 1.2.2 Assumed environment

ID: SMI-1

**Vector assumes that hardware faults are adequately addressed by the user of MICROSAR Safe.**

The components of MICROSAR Safe can support in the detection and handling of some hardware faults.
MICROSAR Safe does not provide redundant data storage.
The user of MICROSAR Safe especially has to address faults in volatile random access memory, non-volatile memory, e.g. flash or EEPROM, and the CPU.
See also SMI-14.

ID: SMI-17

**Vector assumes that hardware and compiler manuals are correct and complete.**

Vector uses the hardware reference manuals and compiler manuals for the development of MICROSAR Safe. Vector has no means to verify correctness or completeness of the hardware and compiler manuals.
Example information that may be critical from these manuals is the register assignment by compiler. This information is used to built up the context that is saved and restored by the operating system.

ID: SMI-2

**Vector assumes that the platform types selected by the user of MICROSAR Safe adequately reflect the hardware and compiler environment.**

The user of MICROSAR Safe is responsible for selecting the correct platform types (PlatformTypes.h). Especially the size of the predefined types must match the target environment.
Example: A uint32 must be mapped to an unsigned integer type with a size of 32 bits.
The user of MICROSAR Safe can use the platform types provided by Vector. Vector has created and verified the platform types mapping according to the information provided by the user of MICROSAR Safe.

ID: SMI-10

**Vector assumes that the reset or powerless state is a safe state of the system.**

This assumption is added to this Safety Manual, because it is used in Vector's safety analyses and development process.

ID: SMI-12

**Vector assumes that the user of MICROSAR Safe initializes all components of MICROSAR Safe prior to using them.**

This constraint is required by AUTOSAR anyway. It is added to this Safety Manual, because Vector assumes initialized components in its safety analyses and development process.
Correct initialization can be verified, e.g. during integration testing.

ID: SMI-16

**Vector assumes that the user of MICROSAR Safe only passes valid pointers at all interfaces to MICROSAR Safe components.**

Plausibility checks on pointers are performed by MICROSAR Safe (see also SMI-18), but they are limited.
Also the length and pointer of a buffer provided to a MICROSAR Safe component need to be consistent.
This assumption also applies to QM as well as ASIL components.
This can e.g. be verified using static code analysis tools, reviews and integration testing.

ID: SMI-20

**Vector assumes that the user of MICROSAR Safe implements a timing monitoring using e.g. a watchdog.**

The components of MICROSAR Safe do not provide mechanisms to monitor their own timing behavior.
Vector's MICROSAR Safe.Watchdog can be used to fulfill this assumption.
If the functional safety concept also requires a logic monitoring, Vector's MICROSAR Safe.Watchdog can be used to implement it.
See also SMI-14.

ID: SMI-33

**Vector assumes that the user of MICROSAR Safe provides sufficient resources in RAM, ROM, stack and CPU runtime for MICROSAR Safe.**

Selection of the micro-controller and memory capacities as well as dimensioning of the stacks is in the responsibility of the user of MICROSAR Safe.
If MICROSAR Safe components have specific requirements, these are documented in the respective Technical Reference document.

ID: SMI-9

**Vector assumes that for one AUTOSAR functional cluster (e.g. System Services, Operating System, CAN, COM, etc.) only components from Vector are used.**

This assumption is required because of dependencies within the development process of Vector.
This assumption does not apply to the MCAL or the EXT cluster.
Vector may have requirements on MCAL or EXT components depending on the upper layers that are used and provided by Vector. For example, the watchdog driver is considered to have safety requirements allocated to its initialization and triggering services. Details are described in the component specific parts of this safety manual.
The only exception to this assumption is the Flash EEPROM Emulation (FEE) for Infineon micro-controllers. Vector does not provide a FEE for Infineon micro-controllers.

ID: SMI-32

**The user of MICROSAR Safe shall provide an argument for coexistence for software that resides in the same partition as components from MICROSAR Safe.**

Vector considers an ISO 26262-compliant development process for the software as an argument for coexistence (see [1] Part 9 Clause 6).
Redundant data storage as the only measure by the other software is not considered a sufficient measure.
If ASIL components provided by Vector are used, this requirement is fulfilled.

### 1.2.3 Assumed process

ID: SMI-14

**The user of MICROSAR Safe shall be responsible for the functional safety concept.**

The overall functional safety concept is in the responsibility of the user of MICROSAR Safe. MICROSAR Safe can only provide parts that can be used to implement the functional safety concept of the item.

ID: SMI-3

**The user of MICROSAR Safe shall evaluate all tools (incl. compiler) that are used by the user of MICROSAR Safe according to ISO 26262:8-11.**

Evaluation especially has to be performed for the compiler, linker, debugging and test tools.
Vector provides a guideline for the evaluation of the Tool Confidence Level (TCL) for the tools provided by Vector (e.g. DaVinci Configurator).
Vector has evaluated the tools exclusively used by Vector during the development of MICROSAR Safe.

ID: SMI-4

**The user of MICROSAR Safe shall perform the integration (ISO 26262:6-10) and verification (ISO 26262:6-11) processes as required by ISO 26262.**

Especially the safety mechanisms must be verified in the final target ECU.
Vector assumes that by performing the integration and verification processes as required by ISO 26262 the generated configuration data, e.g. data tables, task priorities or PDU handles, are sufficiently checked. An additional review of the configuration data is then considered not necessary.
Integration does not apply to a MICROSAR Safe component that consists of several subcomponents. This integration is already performed by Vector. However, integration of all MICROSAR Safe components in the specific use-case of the user of MICROSAR Safe is the responsiblity of the user of MICROSAR Safe.
Support by Vector can be requested on a per-project basis.

ID: SMI-5

**The user of MICROSAR Safe shall verify all code that is changed during integration of MICROSAR Safe.**

Code that is typically changed by the user of MICROSAR Safe during integration comprises generated templates, hooks, callouts, or similar.
This assumption also applies if interfaces between components are looped through user-defined functions.
Vector assumes that this verification also covers ISO 26262:6-9.
Support by Vector can be requested on a per-project basis.

ID: SMI-30

**The user of MICROSAR Safe shall only modify source code of MICRSAR Safe that is explicitly allowed to be changed.**

Usually no source code of MICROSAR Safe is allowed to be changed by the user of MICROSAR Safe.
The user of MICROSAR Safe can check if the source code was modified by e.g, comparing it to the original delivery.

ID: SMI-8

**The user of MICROSAR Safe shall verify generated functions according to ISO 26262:6-9.**

Generated functions can be identified when searching through the generated code.
Support by Vector can be requested on a per-project basis.
An example of generated functions are the configured rules of the Basic Software Manager (BSWM). Their correctness can only be verified by the user of MICROSAR Safe. Please note, however, that BSWM does not provide safety features.

ID: SMI-11

**The user of MICROSAR Safe shall ensure data consistency for its application.**

Data consistency is not automatically provided when using MICROSAR Safe. MICROSAR Safe only provides services to support enforcement of data consistency. Their application is in the responsibility of the user of MICROSAR Safe.
To ensure data consistency in an application, critical section need to be identified and protected.
To identify critical sections in the code, e.g. review or static code analysis can be used.
To protect critical sections, e.g. the services to disable and enable interrupts provided by the MICROSAR Safe operating system can be used.
To verify correctly implemented protection, e.g. stress testing or review can be used.
Note the AUTOSAR specification with respect to nesting and sequence of calls to interrupt enabling and disabling functions.

ID: SMI-15

**The user of MICROSAR Safe shall follow the instructions of the corresponding Technical Reference of the components.**

Especially deviations from AUTOSAR specifications are described in the Technical References.
The possible implementations of exclusive areas are described in the Technical References.

ID: SMI-19

**The user of MICROSAR Safe shall execute the MICROSAR Safe Silence Verifier (MSSV).**

Details on the required command line arguments and integration into the tool chain can be found in [4].
If the report shows “Overall Check Result: Fail", please contact the Safety Manager at Vector. See the Product Information MICROSAR Safe [3] for more the contact details.

ID: SMI-31

**The user of MICROSAR Safe shall verify the consistency of the binary downloaded into the ECU's flash memory.**

This also includes re-programming of flash memory via a diagnostics service. The consistency of the downloaded binary can be checked by the bootloader or the application. MICROSAR Safe assumes a correct program image.

ID: SMI-18

**The user of MICROSAR Safe shall enable plausibility checks for the MICROSAR Safe components.**

This setting is necessary to increase fault detection and prevent fault propagation.
This setting can be found at /MICROSAR/EcuC/EcucGeneral/EcuCSafeBswChecks in the DaVinci Configurator.
This setting does not enable error reporting to the DET component.
This setting is enforced by an MSSV plug-in.

# 2 Derivative-specific Part

# 3 Component-specific Part

## 3.1 Safety Manual Crc

### 3.1.1 Safety features

Crc provides the following safety features:

| ID | Safety feature |
|---|---|
| CREQ-858 | Crc shall provide a service to calculate 8-bit SAE-J1850 CRC. |
| CREQ-859 | Crc shall provide a service to calculate 8-bit 0x2F CRC. |
| CREQ-860 | Crc shall provide a service to calculate 16-bit CCITT CRC. |
| CREQ-861 | Crc shall provide a service to calculate 32-bit IEEE-802.3 CRC. |
| CREQ-862 | Crc shall provide a service to calculate 32-bit E2E Profile 4 CRC. |

### 3.1.2 Configuration constraints

This component has no configuration constraints.

### 3.1.3 Additional verification measurse

ID: SMI-49

**The user of MICROSAR Safe shall verify that the CRC is calculated for the intended data.**

This includes the intended buffer and its size (see also SMI-16), start value and if it is the first call to the service.
Verification can be performed by the "magic check" (see AUTOSAR SWS Crc).
If Crc is used by a MICROSAR Safe component (e.g. E2E, NvM), this requirement is fulfilled for the MICROSAR Safe component.

### 3.1.4 Dependencies to other components

#### 3.1.4.1 Safety features required from other components

This component does not require safety features from other components.

#### 3.1.4.2 Coexistence with other components

This component does not require coexistence with other components.

It is assumed that the user of Crc has the adequate ASIL.

### 3.1.5 Dependencies to hardware

This component does not use a direct hardware interface.

## 3.2 Safety Manual NvM

### 3.2.1 Safety features

ID: SMI-21

**NvM provides the following safety features:**

| ID | Safety feature |
|---|---|
| CREQ-724 | NvM shall provide a service to read a single NvM block from NVRAM. |
| CREQ-725 | NvM shall provide a service to write a single NvM block to NVRAM. |
| CREQ-730 | NvM shall provide a service to read all possbile NvM blocks from NVRAM. |
| CREQ-731 | NvM shall provide a service to write all possible NvM blocks to NVRAM. |
| CREQ-746 | NvM shall provide configurable callbacks to synchronize block data. |

NvM can detect missing, corruption and masquerading (lower layers provide the wrong block) of NvM blocks.

ID: SMI-29

**The user of MICROSAR Safe must design the system in a way that in case of the absence of non-volatile data it is still safe (e.g. safe state or degradation). It cannot be assured by the memory stack that data is saved completely or at all because a reset or loss of energy might happen at any time, e.g. brown-out, black-out.**
**This also implies that it is in general impossible to guarantee that the latest information is available in the non-volatile memory, e.g. the system is reset before memory stack is even notified to write data to non-volatile memory.**
**Thus, safety-related functionality may not rely on the availability of data in non-volatile memory.**
**Since the availability of data in non-volatile memory cannot be guaranteed in any case, the only sensible use-case is reading safety-related calibration data.**
**Writing of data into non-volatile memory must be verified to assure that the information is available in non-volatile memory. Verification can only be done manually in a protected environment, e.g. at end of line, in a workshop, etc.**
**ECU software cannot verify if data was written, since at any time a reset could occur and the information that had to be written is lost immediately.**
**Reading of data does not modify data stored in non-volatile memory. Thus, reading can be used by safety-related functionality. The memory stack assures that the read data is identical to the data stored in non-volatile memory.**
**The availability may be increased by e.g. redundant storage.**

### 3.2.2 Configuration constraints

ID: SMI-25

**The user of MICROSAR Safe shall configure and verify the following attributes for each NvM block that contains safety-related data:**

> ▪ Set /MICROSAR/NvM/NvMBlockDescriptor/NvMBlockUseCrc to TRUE.
> ▪ Set /MICROSAR/NvM/NvMBlockDescriptor/NvMBlockCrcType to NVM_CRC32.

ID: SMI-26

**The user of MICROSAR Safe shall configure and verify the following attribute:**

> ▪ Set /MICROSAR/NvM/NvMCommon/NvMUseBlockIdCheck to TRUE.

ID: SMI-24

**The user of MICROSAR Safe shall enable and verify the following configuration parameters:**

> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkStatusUninit
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkParamPointer
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkParamDataIndex
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkDoParamChk
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkParamBlockid
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkMngmtType
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkBlockProtection
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkBlockPending
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkRamBlockLengths
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkRomBlockLengths
> ▪ /MICROSAR/NvM/NvMCommonVendorParams/NvMDevErrorChecks/NvMCfgChkBlockLengthStrict

Note: These parameters are enabled by default.

### 3.2.3 Additional verification measures

ID: SMI-22

**The user of MICROSAR Safe shall pass the intended block ID for reading and writing of a single NvM block. NvM cannot detect if an unintended, but configured block is provided by the user.**
**Verification can be performed during integration testing.**

ID: SMI-23

**The user of MICROSAR Safe shall verify that the buffer passed for reading and writing of a single NvM block has to be valid and sufficiently large for the passed block ID.**
**Verification can be performed by a review of the generated configuration and the code passing the pointer and block ID to the NvM.**

ID: SMI-48

**The user of MICROSAR Safe shall verify the size of the internal NvM buffer.**
The buffer has the symbol name NvM_InternalBuffer_au8.
The buffer is generated when at least one of the following features is used:

> ▪ explicit synchronization
> ▪ repair of redundant blocks
> ▪ blocks with CRC protection

The buffer size shall be at least the size of the largest NVM block plus the size of the configured CRC value.
Verification can be performed e.g. by review.

### 3.2.4 Dependencies to other components

#### 3.2.4.1 Safety features required from other components

ID: SMI-28

**The used Crc library shall provide the CRC calculation routines as safety feature.**
**If the Crc library from MICROSAR Safe is used, this dependency is fulfilled.**

#### 3.2.4.2 Coexistence with other components

ID: SMI-27

**This component requires coexistence with Dem, MemIf, RTE, BswM, and Det components if the interface for those components is configured.**

### 3.2.5 Dependencies to hardware

This component does not use a direct hardware interface.

## 3.3 Safety Manual EcuM

### 3.3.1 Safety features

ID: SMI-34

**EcuM provides the following safety features:**

| ID | Safety feature |
|---|---|
| CREQ-543 | EcuM shall perform validation of all postbuild configurable BSW module configuration parameters. |
| CREQ-375 | EcuM shall provide a callout to set programmable interrupts during the startup phase. |
| CREQ-525 | EcuM shall provide a callout to initialize driver prior the start of the OS. |
| CREQ-488 | EcuM shall provide a callout to determine the Postbuild configuration data. |
| CREQ-505 | EcuM shall provide a callout to initialize drivers prior Postbuild data is available. |
| CREQ-391 | EcuM shall provide a callout to reset the ECU. |
| CREQ-381 | EcuM shall provide a callout to generate a RAM Hash. |
| CREQ-440 | EcuM shall provide a callout to check the RAM Hash. |
| CREQ-509 | EcuM shall provide a callout to re-initialize drivers during a restart. |
| CREQ-445 | EcuM shall provide a service to set the current shutdown target of the ECU. |
| CREQ-372 | EcuM shall provide a service to initiate the ECU shutdown depending on the shutdown target. |
| CREQ-431 | EcuM shall provide a callout to notify Errors from the ECU management. |
| CREQ-421 | EcuM shall provide a service to complete the ECU shutdown process. |

Note: RAM Hash generation and checking callouts are only available when sleep modes are configured.

ID: SMI-38

**If EcuM service to complete the shutdown process is called prior to initiate the shutdown process, no shutdown will be performed.**

### 3.3.2 Configuration constraints

ID: SMI-37

**The user of MICROSAR Safe shall configure the following attribute:**

> ▪ Set /MICROSAR/EcuM/EcuMGeneral/EcuMEnableFixBehavior to FALSE.

This setting is enforced by a MSSV plugin.

ID: SMI-36

**The user of MICROSAR Safe shall configure the following attribute:**

> ▪ Set /MICROSAR/EcuM/EcuMFlexGeneral/EcuMEnableDefBehaviour to FALSE.

This setting is enforced by a MSSV plugin.

### 3.3.3 Additional verification measurse

ID: SMI-39

**The user of MICROSAR Safe shall verify the intended initialization procedure during integration testing.**

The user of MICROSAR Safe can verify the intended initialization procedure by performing the following tests:

> ▪ Start the ECU and verify that the intended initalization routines are called. This needs to be verified for each Postbuild-selectable configuration.
> ▪ Corrupt the Postbuild data (but not corresponding CRC) in non-volatile memory and start the ECU. Then verify that the corruption is detected by EcuM.
> ▪ Start the ECU and verify that the intended Postbuild-selectable configuration is used by the EcuM. This needs to be verified for each Postbuild-selectable configuration.

A start of the ECU includes a "cold-start", reset as well as wake-up from sleep if applicable.

This requirement only applies if TSR-1 is considered a safety requirement.

ID: SMI-35

**The user of MICROSAR Safe shall verify the intended shutdown procedure during integration testing.**

The user of MICROSAR Safe can verify the intended shutdown procedure by shutting down the ECU with all configured shutdown paths. A shutdown path is considered a call to the service that sets the current shutdown target with a relevant (e.g. combination used to achieve safe state) combination of its parameters. For each shutdown path the intended final state of the ECU (e.g. sleep, shutdown, reset) and the method of reset (e.g. using MCU or Watchdog) is used.

The user of MICROSAR Safe shall also consider the service to initiate the ECU shutdown depending on the shutdown target as a possible shutdown path.

The user of MICROSAR Safe shall also verify the default shutdown target.

This requirement only applies if TSR-3 is considered a safety requirement.

ID: SMI-40

**The user of MICROSAR Safe shall verify that the memory region used for RAM hash generation and verification is as intended.**

Verification can be e.g. performed by review.

### 3.3.4 Dependencies to other components

#### 3.3.4.1 Safety features required from other components

ID: SMI-42

**The used operating system shall provide the service to get the core ID as safety feature.**

If the operating system from MICROSAR Safe is used, this dependency is fulfilled.

This requirement only applies if TSR-1 is considered a safety requirement.

#### 3.3.4.2 Coexistence with other components

ID: SMI-41

**This component requires coexistence with RTE, BswM, ComM, Dem, Gpt, MCU and Det components if the interface for those components is configured.**

### 3.3.5 Dependencies to hardware

This component does not use a direct hardware interface.

## 3.4 Safety Manual Det

### 3.4.1 Safety features

This component does not provide safety features.

### 3.4.2 Configuration constraints

This component does not have configuration constraints.

ID: SMI-60

**The Det should not be used in series production. If it is used in series production the extended debug features shall be switched off, because they are only relevant if a debugger is attached.**

The user of MICROSAR Safe shall configure and verify the following attribute:

> ▪ /MICROSAR/Det/DetGeneral/DetExtDebugSupport = False

### 3.4.3 Additional Verification measures

This component does not require additional verification measures.

### 3.4.4 Dependencies to other components

#### 3.4.4.1 Safety features required from other components

This component does not require safety features from other components.

#### 3.4.4.2 Coexistence with other components

ID: SMI-54

**This component requires coexistence with Dlt component if the interface for this component is configured. Callouts used during series production must also provide an argument for coexistence.**

### 3.4.5 Dependencies to hardware

This component does not use a direct hardware interface.

## 3.5 Safety Manual BswM

### 3.5.1 Safety features

This component does not provide safety features.

### 3.5.2 Configuration constraints

This component does not have configuration constraints.

### 3.5.3 Additional Verification measures

This component does not require additional verification measures.

### 3.5.4 Dependencies to other components

#### 3.5.4.1 Safety features required from other components

This component does not require safety features from other components.

#### 3.5.4.2 Coexistence with other components

ID: SMI-55

**This component requires coexistence with Com, Det, Dem, Nm, RTE, NvM, PduR, FrSm, Dcm, LinIf, LinSm, EcuM, CanSM, and ComM components if the interface for those components is configured.**

### 3.5.5 Dependencies to hardware

This component does not use a direct hardware interface.

## 3.6 Safety Manual MemIf

### 3.6.1 Safety features

This component does not provide safety features.

### 3.6.2 Configuration constraints

This component does not have configuration constraints.

### 3.6.3 Additional Verification measures

This component does not require additional verification measures.

### 3.6.4 Dependencies to other components

#### 3.6.4.1 Safety features required from other components

This component does not require safety features from other components.

#### 3.6.4.2 Coexistence with other components

ID: SMI-56

**This component requires coexistence with Det, Fee, and Ea components if the interface for those components is configured.**

### 3.6.5 Dependencies to hardware

This component does not use a direct hardware interface.

# 4 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com