Safety Manual

	- -

VECTOR

Safety Manual

Version	Date	Author	Remarks
	- -		
	- -		
	- -		
	- -		
	- -		
	- -		- -
	- -		

1 General Part

1.1 Introduction

1.1.1 Purpose

-

1.1.2 Scope

-

1.1.3 Definitions

shall *shall not* *should* *can*

Shall

Shall not

Should

Can

1.1.4 References

No.	Source	Title	Version

1.1.5 Overview

▶

▶

▶

1.2 Concept

- -

1.2.1 Technical Safety Requirements

1.2.1.1 Initialization

TSR-1 The system shall initialize the CPU, MPU, watchdog, and operating system.

-

1.2.1.2 Self-test

TSR-2 The system shall perform self-tests based on the requirements of the system.

-

-

- -

1.2.1.3 Reset of ECU

TSR-3 The system shall reset itself in case of a detected fault.

1.2.1.4 Non-volatile memory

1.2.1.4.1 Saving data

TSR-4 The system shall save information in non-volatile memory.

1.2.1.4.2 Loading data

TSR-5 The system shall retrieve the last stored information from non-volatile memory.

1.2.1.5 Scheduling

1.2.1.5.1 Deterministic, hard real-time scheduling

TSR-6 The system shall execute the specified functions within their respective hard timing limits.

-

-

1.2.1.6 Partitioning

1.2.1.6.1 Memory partitioning

TSR-7 The system shall protect software applications from unspecified memory access.

1.2.1.6.2 Time partitioning

1.2.1.6.2.1 Timing protection

TSR-8 The system shall detect timing faults in the software.

1.2.1.6.2.2 Killing of applications

TSR-9 The system shall terminate software applications.

VECTOR

1.2.1.7 Communication protection

1.2.1.7.1 Inter ECU communication

1.2.1.7.1.1 End-to-end protection

TSR-10 The system shall protect communication between its elements.

-   -

1.2.1.7.1.2 Protection by cryptographic algorithms

TSR-11 The system shall protect communication between its elements using cryptographic hash algorithms to detect accidental corruption of the communication.

1.2.1.7.2 Intra ECU communication

1.2.1.7.2.1 Intra OS application communication

TSR-16 The microcontroller software shall communicate within its applications.

-

-

1.2.1.7.2.2 Inter OS application communication

TSR-12 The microcontroller software shall communicate between its applications.

-   -

-

-

-

1.2.1.8 Watchdog services

1.2.1.8.1 Program flow monitoring

TSR-13 The system shall provide a mechanism to detect faults in program flow.

1.2.1.8.2 Alive monitoring

TSR-14 The system shall provide a mechanism to detect stuck software.

1.2.1.8.3 Deadline monitoring

TSR-15 The system shall provide a mechanism to detect deadline violations.

1.2.1.9 Peripheral in- and output

1.2.1.9.1 Peripheral input

TSR-17 The system shall read input values from peripheral devices.

-

1.2.1.9.2 Peripheral output

TSR-18 The system shall write output values to peripheral devices.

-

1.2.2 Environment

1.2.2.1 Safety Concept

-

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

not **not**

- -

-

The user of MICROSAR Safe shall adequately address hardware faults.

-

-

-

-

The user of MICROSAR Safe shall ensure that the reset or powerless state is a safe state of the system.

-

The user of MICROSAR Safe shall implement a timing monitoring using e.g. a watchdog.

The user of MICROSAR Safe shall ensure an end-to-end protection for safety-relevant communication between ECUs.

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

1.2.2.2 Use of MICROSAR Safe Components

The user of MICROSAR Safe shall adequately select the type definitions to reflect the hardware platform and compiler environment.

-

The user of MICROSAR Safe shall initialize all components of MICROSAR Safe prior to using them.

-

The user of MICROSAR Safe shall only pass valid pointers at all interfaces to MICROSAR Safe components.

-

-

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

-

-

The user of MICROSAR Safe shall configure and use the interrupt system correctly.

▶

▶

▶

1.2.2.3 Partitioning

-

The user of MICROSAR Safe shall ensure that for one AUTOSAR functional cluster (e.g. System Services, Operating System, CAN, COM, etc.) only components from Vector are used.

-

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

-

-

-

The user of MICROSAR Safe shall verify that the memory mapping is consistent with the partitioning concept.

1.2.2.4 Resources

-

The user of MICROSAR Safe shall provide sufficient resources in RAM, ROM, stack and CPU runtime for MICROSAR Safe.

1.2.3 Process

-

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

-

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

-

-

-

-

-

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

-

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

-

-

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

-

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.

-

-

-

-

The user of MICROSAR Safe shall ensure that a consistent set of generated configuration is used for verification and production.

-

The user of MICROSAR Safe shall verify the integrity of the delivery by Vector.

-

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

-

-

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.

2 Safety Manual BswM

2.1 Safety features

2.2 Configuration constraints

2.3 Additional Verification measures

2.4 Dependencies to other components

2.4.1 Safety features required from other components

2.4.2 Coexistence with other components

-

2.5 Dependencies to hardware

3 Safety Manual Crc

3.1 Safety features

-

-	- -
-	-
-	-
-	- -
-	-

3.2 Configuration constraints

3.3 Additional verification measures

-

4 Safety Manual Dem

-

4.1 Safety features

4.2 Configuration constraints

-

- *＿/MICROSAR/Dem/DemGeneral/DemOBDSupport*
  DEM_OBD_NO_OBD_SUPPORT
- */MICROSAR/Dem/DemGeneral/DemJ1939Support* *FALSE*

4.3 Additional verification measures

-

*Dem_Cfg_*`<CallbackType>`
`<CallbackType>`

Dem_Cfg_Get`<CallbackType>`

Dem_Cfg_Get`<CallbackType>`
NULL_PTR

Dem_Lcfg.c
Dem_Lcfg.h
`<CallbackType>` *CallbackType*

CallbackType	Expected signature of the functions
	`Std_ReturnType <name>(boolean *IsAllowed)`
	`Std_ReturnType <name>(uint32 DTC, Dem_DTCStatusMaskType oldStatus, Dem_DTCStatusMaskType newStatus)`
	`Std_ReturnType <name>(sint8 *FDC)`
	`Std_ReturnType <name>(void)`

	`Std_ReturnType <name>(Dem_EventStatusExtendedType oldStatus, Dem_EventStatusExtendedType newStatus)`
	`Std_ReturnType <name>(Dem_InitMonitorReasonType initReason)`
	`Std_ReturnType <name>(void)`

-

NULL_PTR

Dem_Lcfg.h

Configuration Macro	Expected signature of the function
DEM_CFG_GLOBALCBKDATA_FUNC	`Std_ReturnType <name>(Dem_EventIdType EventId)`
DEM_CFG_GLOBALCBKSTATUS_FUNC	`Std_ReturnType <name>(Dem_EventIdType EventId, Dem_EventStatusExtendedType oldStatus, Dem_EventStatusExtendedType newStatus)`
DEM_CFG_GLOBALCBKCONTROLDTCSETTING_FUNC	`Std_ReturnType <name>(boolean Status)`

-

Dem_Cfg_DataElementTable[] *ElementKind*

Dem_Cfg_DataElementTable[] *Dem_Lcfg.c*

ElementKind	Expected signature of the function
DEM_CFG_DATA_FROM_CBK_STORED	`Std_ReturnType <name>(uint8 *data)`
DEM_CFG_DATA_FROM_CBK_CURRENT	`Std_ReturnType <name>(uint8 *data)`
DEM_CFG_DATA_FROM_CBK_STORED_WITH_EVENTID	`Std_ReturnType <name>(Dem_EventIdType EventId, uint8 *data)`
DEM_CFG_DATA_FROM_CBK_CURRENT_WITH_EVENTID	`Std_ReturnType <name>(Dem_EventIdType EventId, uint8 *data)`

	NULL_PTR

-

Dem_Cfg_DataElementTable[] *ElementSize*

ReadData *DemDataElementClass*

Dem_Lcfg.c

-

Dem_Cfg_MemoryBlockId[]

-

Dem_Lcfg.c

-

Dem_Cfg_MemoryBlockId
Dem_Cfg_MemoryDataSize

Dem_Cfg_MemoryDataPtr

-

Dem_Cfg_MemoryBlockId[1]
Dem_Cfg_MemoryDataSize[1]
Dem_Cfg_MemoryDataPtr[1]

Dem_Cfg_MemoryBlockId *Dem_Cfg_MemoryDataSize*
Dem_Cfg_MemoryDataPtr *Dem_Lcfg.c*

-

Dem_GetEventExtendedDataRecord *Dem_GetEventFreezeFrameData*
GetExtendedDataRecord *GetFreezeFrameData*
DiagnosticInfo *GeneralDiagnosticInfo* *DiagnosticMonitor*

▶ *DestBuffer*
sizeof(Dem_MaxDataValueSize)

Dem_MaxDataValueSize *Rte_Type.h*
DEM_CFG_SIZEOF_MAX_DATA_VALUE_TYPE
Dem_Lcfg.h

-

Dem_Cfg_MemoryDataPtr[]
Dem_Cfg_PrimaryEntryType
DEM_CFG_MEMORY_PRIMARY_INDEX
DEM_CFG_MEMORY_PRIMARY_INDEX *DEM_CFG_GLOBAL_PRIMARY_SIZE*
DEM_CFG_GLOBAL_SECONDARY_SIZE

DEM_CFG_MEMORY_PRIMARY_INDEX
DEM_CFG_GLOBAL_PRIMARY_SIZE *DEM_CFG_GLOBAL_SECONDARY_SIZE*
Dem_Lcfg.h
Dem_Cfg_PrimaryEntryType *Dem_Lcfg.h*
Dem_Cfg_MemoryDataPtr[] *Dem_Lcfg.c*

-

DemGeneral/DemTimeSeriesSnapshot

Dem_Cfg_MemoryDataPtr[]
Dem_Cfg_TimeSeriesEntryType
DEM_CFG_MEMORY_TIME_SERIES_INDEX
DEM_CFG_MEMORY_TIME_SERIES_INDEX
DEM_CFG_GLOBAL_TIMESERIES_SNAPSHOTS_SIZE

DEM_CFG_MEMORY_TIME_SERIES_INDEX
DEM_CFG_GLOBAL_TIMESERIES_SNAPSHOTS_SIZE *Dem_Lcfg.h*
Dem_Cfg_TimeSeriesEntryType *Dem_Lcfg.h*
Dem_Cfg_MemoryDataPtr[] *Dem_Lcfg.c*

-

sizeof(Dem_Cfg_CommitBuffer)
Dem_Cfg_MemoryDataSize[]

Dem_Cfg_CommitBuffer *Dem_Cfg_MemoryDataSize[]*
Dem_Lcfg.c

4.4 Safety features required from other components

4.5 Dependencies to hardware

5 Safety Manual Det

5.1 Safety features

5.2 Configuration constraints

-

5.3 Additional Verification measures

5.4 Dependencies to other components

5.4.1 Safety features required from other components

5.4.2 Coexistence with other components

-

5.5 Dependencies to hardware

6 Safety Manual EcuM

6.1 Safety features

-

-	
-	
-	

-

-	
-	
-	
-	
-	
-	
-	
-	-
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	
-	

-

6.2 Configuration constraints

-

6.3 Additional verification measurse

-

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

▶ -

▶ -

▶ -
-
- -

-

-

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

-

-

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

6.4 Dependencies to other components

6.4.1 Safety features required from other components

-

-

6.4.2 Coexistence with other components

-

6.5 Dependencies to hardware

7 Safety Manual Fee

7.1 Safety features

7.2 Configuration constraints

-

FEE_INTERNAL_BUFFER_SIZE

AddressAlignment
Fee_PartitionConfig_at

-

/MICROSAR/Fee/FeeSpecificFeatures/FeeDataConversionApi *FALSE*

7.3 Additional verification measures

-

Fee_Read
Fee_GetEraseCycle *Fee_GetWriteCycle*

Fee_Read
MemIf_Read

-

Fee_InitEx

Fee_InitEx *Fee_Init*

7.4 Safety features required from other components

7.5 Dependencies to hardware

8 Safety Manual MemIf

8.1 Safety features

8.2 Configuration constraints

8.3 Additional verification measures

8.4 Dependencies to other components

8.4.1 Safety features required from other components

8.4.2 Coexistence with other components

-

8.5 Dependencies to hardware

9 Safety Manual NvM

9.1 Safety features

-

-	
-	
-	
-	
-	

-

- -

-

-

- -

-

- -

-

-

-

-

-

9.2 Configuration constraints

-

each NvM block that contains safety-related data

▶

▶

-

9.3 Additional verification measures

-

-

-

9.4 Dependencies to other components

9.4.1 Safety features required from other components

-

9.4.2 Coexistence with other components

-

9.5 Dependencies to hardware

10 Safety Manual Rte

10.1 Safety features

-

ID	Safety feature
-	- -
-	- -
-	- -
-	
-	
-	
-	
-	
-	
-	-
-	-
-	-
-	-
-	-
-	-
-	-
-	
-	
-	

-	
-	
-	
-	
-	
-	-
-	-
-	
-	
-	

10.2 Configuration constraints

-

10.3 Additional verification measures

-

-
-

- - -

10.3.1 Guided integration testing

-

10.3.1.1 BSW configuration

-

-

-

-

void

-

void

10.3.1.2 Executable Entity Scheduling

-

-

-
-
-
-

Rte_Switch

-

-

-

-

-

-

-

- *Rte_Call*
- *Rte_Result*
- *Rte_Receive*
- *Rte_Feedback*
- *Rte_SwitchAck*

-

-

-

-

all

10.3.1.3 SWC Communication

-

-

-
-
-

- -

-

- - -

-

-

▶

-

-

-

-

-

10.3.1.4 Usage of RTE Headers

-

defines *typedefs*

▶ -

▶

▶

▶ -

▶ -

▶

Rte_Type h *Rte_* *h*
*Rte_* *h*

defines
defines

defines -

defines

-

- *Rte_Port*
- *Rte_Ports*
- *Rte_NPorts*

10.3.1.5 Usage of RTE APIs

-

-

10.3.1.6 Configuration of RTE APIs

-

Rte_IWrite *Rte_IWriteRef*

- - -
- -
-

-

- -

-

-

RTE_E_INVALID

- - -

-

- -

RTE_E_INVALID

-

RTE_E_NEVER_RECEIVED

- - -

-

RTE_E_NEVER_RECEIVED

-

-

-

-

10.4 Safety features required from other components

-

-
-
- -

10.5 Dependencies to hardware

11 Glossary and Abbreviations

11.1 Glossary

Term	Definition
	-

11.2 Abbreviations

Abbreviation	Description
	-

12 Contact

-
-
-
-
-
-