

MICROSAR Diagnostic Event Manager (DEM)

Technical Reference

Vector

Version 2.2.0

Authors	P. Speidel, A. Ditte, S. Hübner
---------	---------------------------------

Status	Released
--------	----------

1 Document Information

1.1 History

Author	Date	Version	Remarks
P. Stöhr	2008-04-08	1.0	Created for OEM "Vector" base of TechnicalReference_DEM_<OEM>.doc
P. Stöhr	2008-06-18	1.0.1	Updated to AUTOSAR Release 3
P. Stöhr	2008-06-19	1.0.2	Added reference [8]
P. Stöhr	2008-06-27	1.0.3	Modified description of configuration
P. Stöhr	2009-01-08	2.0.10	Modified include structure, added chapter NvRAM Demand, added FreezeFrame descriptions, updated description of configuration, added description of scheduling DEM and DCM
P. Stöhr	2009-02-16	2.0.13	Added internal OccurrenceCounter implementation
A. Ditte	2009-04-14	2.0.14	Added description for event de-bouncing Added DEM_E_INV_TIMER_SLOT_VAL in chapter 4.6.1
S. Hübner	2009-08-21	2.1.2	Update R7, DEM 2.08.00 Added Variant Handling (Single Identity/VSG mode)
B. Freiburger	2009-11-11	2.1.3	Add detailed description of AUTOSAR APIs
P. Speidel	2010-03-03	2.1.4	Add information to "post-build settings" description, Add Apild of Xxx_DemGetExtededDtataRecford in chapter 4.6.1, Update caller context information, Updated indicator description (configuration of IndicatorBit is now possible in CANdela and GENy)
S. Hübner	2010-08-23	2.1.5	Release Dem in config variant link-time, only
S. Hübner	2010-12-01	2.2.0	Due sleep/wakeup constraints, no check any more for NV-Block status in Dem 2.14.00, see chapter 4.2 for mandatory configuration/initialization. GENy (Diag_AsrDem.dll 3.3.0.0) now separates Internal/External Events (Event Destination) from creation of PortInterface (Event Kind). Reworked chapter 8.1.1, 8.2.12.5 and 10.1

Table 1-1 History of the Document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR - Specification of Diagnostics Event Manager (AUTOSAR_SWS_DEM.pdf)	V2.2.1
[2]	AUTOSAR – Specification of Development Error Tracer (AUTOSAR_SWS_DET.pdf)	V2.2.0
[3]	AUTOSAR – Specification of Diagnostic Communication Manager (AUTOSAR_SWS_DCM.pdf)	V3.0.0
[4]	AUTOSAR – Specification of NVRAM Manager (AUTOSAR_SWS_NVRAMManager.pdf)	V2.2.0
[5]	AUTOSAR – Specification of Standard Types (AUTOSAR_SWS_StandardTypes.pdf)	V1.2.0
[6]	AUTOSAR_BasicSoftwareModules.pdf	V1.0.0
[7]	ISO 14229-1:2006 Road vehicles – Unified diagnostic services (UDS) – Part 1: Specification and Requirements	-
[8]	Vector internal PostBuild_GeneralProcedure.pdf	V1.x
[9]	Application Note AN-ISC-8-1118, Vector GmbH MICROSAR BSW Compatibility Check	V1.0

Table 1-2 Reference Documents

Contents

1	Document Information	2
1.1	History	2
1.2	Reference Documents	3
2	Component History	12
3	Introduction	13
3.1	Architecture Overview	13
4	Functional Description	15
4.1	Features	15
4.2	Initialization	16
4.3	States	17
4.4	Main Functions	18
4.5	Event De-bouncing	20
4.5.1	Via Counter Based Algorithm	20
4.5.2	Via Time Based Algorithm	20
4.6	Error Handling	21
4.6.1	Development Error Reporting	21
4.6.1.1	Parameter Checking	28
4.6.2	Production Code Error Reporting	31
5	Integration	32
5.1	Scope of Delivery	32
5.1.1	Static Files	32
5.1.2	Dynamic Files	32
5.2	Include Structure	33
5.3	Compiler Abstraction and Memory Mapping	34
5.4	Critical Sections	35
5.4.1	Startup Phase	35
5.4.2	Call of Callback Functions	36
5.5	Call Context	36
5.6	Scheduling of DEM and DCM	37
5.7	NvRAM Demand	37
6	API Description	39
6.1	Type Definitions	39
6.2	Services Provided by DEM	39

6.2.1	Dem_GetVersionInfo	39
6.2.2	Interface ECU State Manager ⇔ DEM	39
6.2.2.1	Dem_PreInit	39
6.2.2.2	Dem_Init	41
6.2.2.3	Dem_Shutdown	41
6.2.2.4	Dem_InitDiagnosticVariant	42
6.2.3	Interface SW-Components via RTE ⇔ DEM	44
6.2.3.1	Dem_SetEventStatus	44
6.2.3.2	Dem_ResetEventStatus	45
6.2.3.3	Dem_PrestoreFreezeFrame	46
6.2.3.4	Dem_ClearPrestoredFreezeFrame	47
6.2.3.5	Dem_SetOperationCycleState	47
6.2.3.6	Dem_GetEventStatus	49
6.2.3.7	Dem_GetEventFailed	50
6.2.3.8	Dem_GetEventTested	50
6.2.3.9	Dem_GetDTCOfEvent	52
6.2.3.10	Dem_SetValueByOemId	52
6.2.3.11	Dem_SetEnableCondition	52
6.2.3.12	Dem_GetFaultDetectionCounter	53
6.2.3.13	Dem_GetIndicatorStatus	53
6.2.3.14	Dem_GetOccurrenceCounter	55
6.2.4	Interface BSW-Components ⇔ DEM	56
6.2.4.1	Dem_ReportErrorStatus	56
6.2.5	Interface DCM ⇔ DEM	57
6.2.5.1	Access DTCs and Status Information	57
6.2.5.1.1	Dem_SetDTCFilter	58
6.2.5.1.2	Dem_SetDTCFilterForRecords	59
6.2.5.1.3	Dem_SetViewFilter	60
6.2.5.1.4	Dem_GetStatusOfDTC	60
6.2.5.1.5	Dem_GetDTCStatusAvailabilityMask	61
6.2.5.1.6	Dem_GetNumberOfFilteredDTC	61
6.2.5.1.7	Dem_GetNextFilteredDTC	62
6.2.5.1.8	Dem_GetDTCByOccurrenceTime	63
6.2.5.1.9	Dem_GetViewIDOfDTC	64
6.2.5.1.10	Dem_GetNextFilteredRecord	64
6.2.5.1.11	Dem_GetNextFilteredDTCAndFDC	65
6.2.5.1.12	Dem_GetNextFilteredDTCAndSeverity	66
6.2.5.1.13	Dem_GetTranslationType	66
6.2.5.1.14	Dem_GetSeverityOfDTC	67
6.2.5.2	Access Extended Data Records and FreezeFrame Data	68
6.2.5.2.1	Dem_DisableDTCRecordUpdate	68

6.2.5.2.2	Dem_EnableDTCRecordUpdate	68
6.2.5.2.3	Dem_GetDTCOfFreezeFrameRecord	69
6.2.5.2.4	Dem_GetFreezeFrameDataByDTC	70
6.2.5.2.5	Dem_GetFreezeFrameDataIdentifierByDTC	71
6.2.5.2.6	Dem_GetSizeOfFreezeFrame	72
6.2.5.2.7	Dem_GetExtendedDataRecordByDTC	73
6.2.5.2.8	Dem_GetSizeOfExtendedDataRecordByDTC	74
6.2.5.3	Clear DTC Information	75
6.2.5.3.1	Dem_ClearDTC	75
6.2.5.4	Control DTC Storage	76
6.2.5.4.1	Dem_DisableDTCStorage	77
6.2.5.4.2	Dem_EnableDTCStorage	78
6.2.5.4.3	Dem_DisableEventStatusUpdate	79
6.2.5.4.4	Dem_EnableEventStatusUpdate	79
6.3	Services Used by DEM	81
6.4	Callback Functions	81
6.4.1	Dem_NvDataInit	81
6.5	Configurable Interfaces	82
6.5.1	Notifications	82
6.5.1.1	Rte_Call_Dem_<ConfiguredName>_InitMonitorForEvent	83
6.5.1.2	Rte_Call_Dem_<ConfiguredName>_EventStatusChanged	84
6.5.1.3	Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged	86
6.5.2	Callout Functions	86
6.5.2.1	Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier	87
6.5.2.2	Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord	88
6.6	Service Ports	88
6.6.1	Client Server Interface	88
6.6.1.1	Provide Ports on DEM Side	88
6.6.1.1.1	DiagnosticMonitor	89
6.6.1.1.2	OperationCycle	89
6.6.1.1.3	ValueByOemId	89
6.6.1.1.4	EnableCondition	90
6.6.1.1.5	IndicatorStatus	90
6.6.1.2	Require Ports on DEM Side	90
6.6.1.2.1	Notification CallbackInitMonitorForEvent	90
6.6.1.2.2	Notification CallbackInitMonitorForFunction	90
6.6.1.2.3	Notification CallbackEventStatusChange	90
6.6.1.2.4	Notification CallbackDTCStatusChange	90
6.6.1.2.5	Callout Function CallbackGetDataValueByDataID{DataId}	91
6.6.1.2.6	Callout Function CallbackGetExtendedDataRecord{RecordNumber}	91
6.6.1.2.7	Callout Function CallbackGetFaultDetectCounter	91

7	Additional Features.....	92
7.1	Multi-Identity Support.....	92
7.1.1	Functionality.....	92
7.1.2	Single-Identity Mode	92
7.1.2.1	Configuration in CANdela	93
7.1.2.2	Configuration in GENy	93
7.1.3	Vehicle System Group (VSG) Mode	94
7.1.3.1	Configuration in CANdela	94
7.1.3.2	Configuration in GENy	96
8	Configuration.....	97
8.1	Configuration with CANdelaStudio.....	97
8.1.1	DTC Settings	97
8.1.2	Freeze Frame Types (Snapshot Records).....	99
8.1.3	Extended Records	99
8.1.3.1	OccurrenceCounter	99
8.1.4	Pre de-bounce algorithms.....	101
8.2	Configuration with GENy	101
8.2.1	Start the Configuration	102
8.2.2	Import CDD	102
8.2.2.1	Single Identity Mode	102
8.2.2.2	Vehicle System Group (VSG) Mode	103
8.2.2.3	Automatic Configuration via CANdelaStudio Files.....	104
8.2.3	Tree View	105
8.2.4	Main Configuration Window	106
8.2.5	Common	107
8.2.6	Software Component Template.....	107
8.2.7	Postbuild Settings	108
8.2.8	Miscellaneous	108
8.2.9	BSW Error Buffer	109
8.2.10	DTC Groups.....	109
8.2.11	NvRam Block IDs.....	110
8.2.12	Manual Configuration.....	110
8.2.12.1	Operation Cycles Window	111
8.2.12.2	Data Object Window	112
8.2.12.3	Extended Data Records.....	112
8.2.12.4	Indicator Window	113
8.2.12.5	Event Configuration Window	113
8.2.12.5.1	DTC Settings	118
8.2.12.5.2	Event Pre-Debouncing.....	119
8.2.12.5.3	Snapshot Records (FreezeFrames).....	119

8.2.12.5.4	Extended Data Records	120
8.2.12.5.5	Indicators	121
8.2.12.5.6	TriggerOnEvent Callback Functions	121
8.2.12.5.7	TriggerOnDtc Callback Functions	121
8.2.13	Software Component Template.....	121
9	AUTOSAR Standard Compliance.....	122
9.1	Deviations	122
9.1.1	APIs and Features not Supported	122
9.1.2	AUTOSAR Defined APIs that Differ in this Implementation	122
9.1.2.1	Service Dem_ClearDTC	122
9.1.2.2	Service Dem_SetEventStatus.....	122
9.1.2.3	Service Dem_ResetEventStatus.....	122
9.1.2.4	Service Dem_SetOperationCycleState	122
9.1.2.5	Service Dem_ReportErrorStatus.....	122
9.1.2.6	Interface Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged	122
9.1.3	Require Ports not Supported	122
9.2	Additions/ Extensions	123
9.2.1	Development Error Reporting – Include Structure	123
9.2.2	Development Error Reporting – Internal Debug Codes	123
9.2.3	Service Dem_GetOccurrenceCounter	123
9.2.4	Name Description of Configurable Interfaces (Notifications)	123
9.2.5	Interface Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier	123
9.2.6	Port Names Length Limitation.....	123
9.2.7	Version Information.....	123
9.3	Limitations.....	124
9.3.1	Limits Checked During Configuration	124
10	Glossary and Abbreviations.....	125
10.1	Glossary.....	125
10.2	Abbreviations	126
11	Contact.....	127

Illustrations

Figure 3-1	AUTOSAR architecture	13
Figure 3-2	Interfaces to Adjacent Modules of the DEM	14
Figure 4-1	DEM States	17
Figure 4-2	Event De-bouncing via counter based algorithm	20
Figure 4-3	Event De-bouncing via time based algorithm.....	21
Figure 5-1	Include Structure	33
Figure 7-1	DEM single identity mode	93
Figure 7-2	DEM Vehicle System Group (VSG) mode.....	94
Figure 7-3	Defining VSGs in CANdelaStudio	95
Figure 7-4	Assigning VSGs to a DTC in CANdelaStudio.....	95
Figure 8-1	OccurrenceCounter	100
Figure 8-2	This CDD supports the internal Occurrence Counter.....	101
Figure 8-3	This CDD does NOT support the internal Occurrence Counter	101
Figure 8-4	Switching DCM/DEM in GENy to single-identity mode	102
Figure 8-5	Select the CDD and the variant in the CDD	103
Figure 8-6	Switching DCM in GENy to VSG mode.....	103
Figure 8-7	Select the CDD and the variant in the CDD	104
Figure 8-8	Tree View	105
Figure 8-9	Main Configuration Window	106
Figure 8-10	Manual Configuration of the DEM via Main Configuration Window	110
Figure 8-11	Manual Configuration of the DEM via Right-Mouse-Button in Tree View..	111
Figure 8-12	Operation Cycles Window	111
Figure 8-13	Data Object Window	112
Figure 8-14	ExtendedData Records	112
Figure 8-15	Indicator Window.....	113
Figure 8-16	Event Configuration Window	114
Figure 8-17	Configurable Options of an Event	116
Figure 8-18	Create New Snapshot Record	120
Figure 8-19	Snapshot Record Window.....	120

Tables

Table 1-1	History of the Document.....	2
Table 1-2	Reference Documents	3
Table 2-1	Component History	12
Table 4-1	Supported SWS features	15
Table 4-2	Not Supported SWS Features.....	16
Table 4-3	Mapping of Service IDs to Services and Corresponding Errors.....	27
Table 4-4	Mapping of additional Service IDs to Services and Corresponding Errors	27
Table 4-5	Errors Reported to DET	27
Table 4-6	Errors Reported to DET – Internal Debug Codes.....	28
Table 4-7	Development Error Reporting: Assignment of Checks to Services	30
Table 5-1	Static Files of Delivery	32
Table 5-2	Generated (Dynamic) Files of Delivery	33
Table 5-3	Compiler Abstraction and Memory Mapping	34
Table 6-1	Dem_GetVersionInfo.....	39
Table 6-2	Dem_PreInit	40

Table 6-3	Dem_Init.....	41
Table 6-4	Dem_Shutdown.....	42
Table 6-5	Dem_InitDiagnosticVariant.....	44
Table 6-6	Dem_SetEventStatus.....	45
Table 6-7	Dem_ResetEventStatus.....	45
Table 6-8	Dem_PrestoreFreezeFrame.....	46
Table 6-9	Dem_ClearPrestoredFreezeFrame.....	47
Table 6-10	Dem_SetOperationCycleState.....	48
Table 6-11	Dem_GetEventStatus.....	50
Table 6-12	Dem_GetEventFailed.....	50
Table 6-13	Dem_GetEventTested.....	51
Table 6-14	Dem_GetDTCOfEvent.....	52
Table 6-15	Dem_GetFaultDetectionCounter.....	53
Table 6-16	Dem_GetIndicatorStatus.....	54
Table 6-17	Dem_GetOccurrenceCounter.....	55
Table 6-18	Dem_ReportErrorStatus.....	56
Table 6-19	Dem_SetDTCFilter.....	59
Table 6-20	Dem_SetDTCFilterForRecords.....	59
Table 6-21	Dem_GetStatusOfDTC.....	61
Table 6-22	Dem_GetDTCStatusAvailabilityMask.....	61
Table 6-23	Dem_GetNumberOfFilteredDTC.....	62
Table 6-24	Dem_GetNextFilteredDTC.....	62
Table 6-25	Dem_GetDTCByOccurrenceTime.....	63
Table 6-26	Dem_GetViewIDOfDTC.....	64
Table 6-27	Dem_GetNextFilteredRecord.....	65
Table 6-28	Dem_GetNextFilteredDTCAndFDC.....	65
Table 6-29	Dem_GetNextFilteredDTCAndSeverity.....	66
Table 6-30	Dem_GetTranslationType.....	67
Table 6-31	Dem_GetSeverityOfDTC.....	68
Table 6-32	Dem_DisableDTCRecordUpdate.....	68
Table 6-33	Dem_EnableDTCRecordUpdate.....	69
Table 6-34	Dem_GetDTCOfFreezeFrameRecord.....	70
Table 6-35	Dem_GetFreezeFrameDataByDTC.....	71
Table 6-36	Dem_GetFreezeFrameDataIdentifierByDTC.....	72
Table 6-37	Dem_GetSizeOfFreezeFrame.....	73
Table 6-38	Dem_GetExtendedDataRecordByDTC.....	74
Table 6-39	Dem_GetSizeOfExtendedDataRecordByDTC.....	75
Table 6-40	Dem_ClearDTC.....	76
Table 6-41	Dem_DisableDTCStorage.....	77
Table 6-42	Dem_EnableDTCStorage.....	78
Table 6-43	Dem_DisableEventStatusUpdate.....	79
Table 6-44	Dem_EnableEventStatusUpdate.....	80
Table 6-45	Services Used by the DEM.....	81
Table 6-46	Dem_NvDataInit.....	82
Table 6-47	Rte_Call_Dem_<ConfiguredName>_InitMonitorForEvent.....	83
Table 6-48	Rte_Call_Dem_<ConfiguredName>_EventStatusChanged.....	85
Table 6-49	Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged.....	86
Table 6-50	Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier.....	87
Table 6-51	Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord.....	88

Table 6-52	DiagnosticMonitor	89
Table 6-53	OperationCycle	89
Table 6-54	IndicatorStatus	90
Table 6-55	CallbackInitMonitorForEvent	90
Table 6-56	CallbackEventStatusChange	90
Table 6-57	CallbackDTCStatusChange	91
Table 6-58	CallbackGetDataValueByDataID{DataId}	91
Table 6-59	CallbackGetExtendedDataRecord{RecordNumber}	91
Table 8-1	Equivalence of DTC Settings in CANdelaStudio and GENy	98
Table 10-1	Glossary	125
Table 10-2	Abbreviations	126

2 Component History

The component history gives an overview of the important milestones that are supported in the different versions of the component.

Component Version	New Features
[01.xx.yy]	Evaluation version based on AUTOSAR release 2.1 (V2.1.1) of Specification of Diagnostics Event Manager
[02.xx.yy]	Evaluation version based on AUTOSAR release 3 (V2.2.1) of Specification of Diagnostics Event Manager
[02.05.yy]	1 st Release Version

Table 2-1 Component History

3 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module Diagnostic Event Manager (DEM) as specified in [1].

Supported AUTOSAR Release*:	3	
Supported Configuration Variants:	link-time (library)	
Vendor ID:	DEM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	DEM_MODULE_ID	54 decimal (according to ref. [6])

* For the precise AUTOSAR Release 3.x please see the release specific documentation.

The DEM is responsible for processing and storing diagnostic events (both DTCs and events reported by other BSW modules) and associated environmental data.

3.1 Architecture Overview

The following figure shows where the DEM is located in the AUTOSAR architecture.

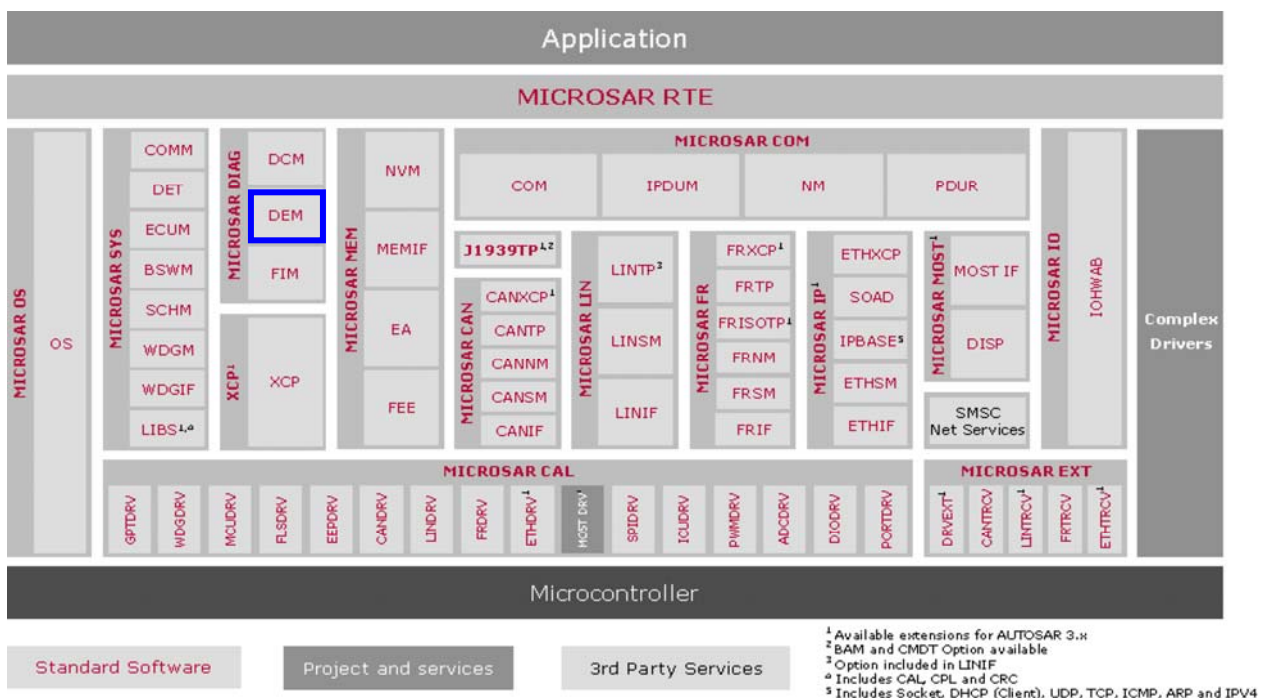


Figure 3-1 AUTOSAR architecture

The next figure shows the interfaces to adjacent modules of the DEM. These interfaces are described in chapter 6.

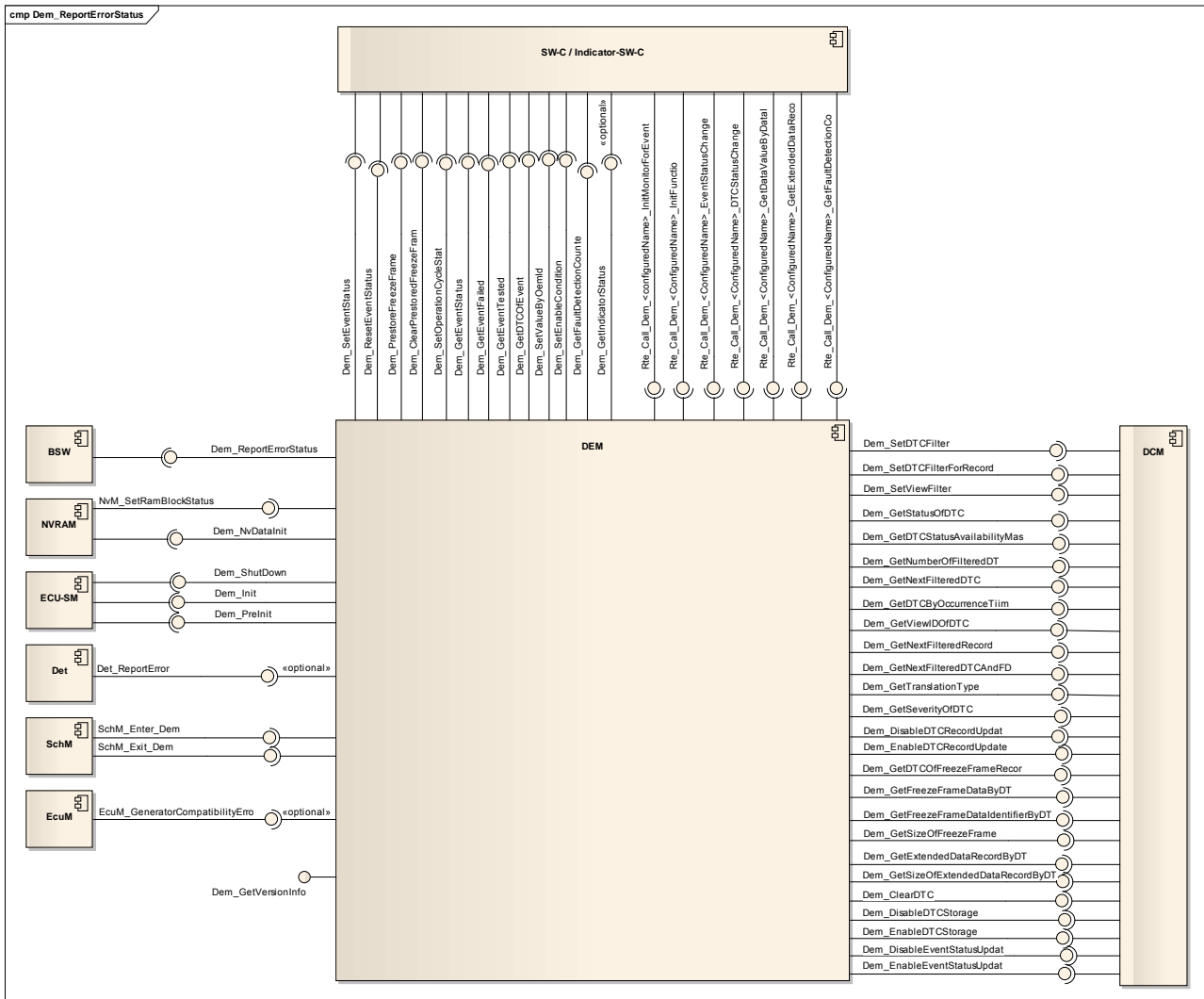


Figure 3-2 Interfaces to Adjacent Modules of the DEM

Applications do not access the services of the BSW modules directly. They use the service ports provided by the BSW modules via the RTE. The service ports provided by the DEM are listed in chapter 6.6 and are defined in [1].

4 Functional Description

4.1 Features

Vector's DEM solution is based on [1].

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features also see chapter 9.

The current implementation offers all basic features of the DEM specification. Some of the optional features (e.g. event combination or OBD) are still not well defined and therefore not supported – refer to the following table for more details.

The following features described in [1] are supported:

Supported Features
All non-optional features described in [1], except features described below

Table 4-1 Supported SWS features

The following features described in [1] are not supported:

Not Supported Features
Event combination The event combination is not supported.
Event mapping The mapping of events to DTCs is "1 to 1" or "1 to 0" and not "n to 1" or "1 to n".
OBD support The OBD support is postponed by AUTOSAR to AUTOSAR phase II.
Mirror / secondary memory support Mirror memory solutions are typically manufacturer specific, for this implementation no support of secondary and/or mirror memory is required.
NvRAM-Manager interface There are pending discussions about the NvRAM manager interface, due to an incomplete SWS specification. Final solution is postponed to AUTOSAR phase II. Vector's solution awaits an updated RAM mirror before <code>Dem_Init()</code> is called and an automatic NvRAM update by the system shut down sequence via <code>NvM_WriteAll()</code> after <code>Dem_Shutdown()</code> call.
Dem_SetValueByOemId This optional API is not supported by Vector specific DEM (it is implemented, but returns always <code>E_NOT_OK</code>).
Dem_SetEnableCondition This optional API is not supported by Vector specific DEM (it is implemented, but returns always <code>E_NOT_OK</code>).

Not Supported Features

CallbackInitMonitorForFunction

To initialize monitors the AUTOSAR API `CallbackInitMonitorForEvent` is used. Therefore the `~ForFunction` callback is not supported by Vector DEM.

ViewId

`Dem_SetViewFilter` is not supported by Vector DEM.

CallbackGetFaultDetectCounter

The `FaultDetectionCounter` is implemented in the DEM. Therefore there is no need to support this API.

Synchronous execution

AUTOSAR recommends a synchronous execution of all event based functions. To keep the call context of these functions, the actions are logged synchronously but are executed cyclically during `Dem_MainFunction()` (see 4.4 for more details).

External event de-bouncing

The external de-bouncing according AUTOSAR can not be implemented in DEM, because the callback to retrieve the fault detection counter from the application is not compatible with the RTE. (see http://www.autosar.org/bugzilla/show_bug.cgi?id=34183)

The counter based de-bouncing can be used instead. If the qualification starts, the fault detection counter has to be update only once with 'prepassed' or 'prefailed'. If the event monitor internal de-bouncing algorithm qualifies the event, it is always possible to pass that result to the DEM with the parameter 'passed' or 'failed'.

Table 4-2 Not Supported SWS Features



Caution

For supporting Vehicle System Group (VSG) Mode (see chapter 7.1 *Multi-Identity Support* and chapter 7.1.3 *Vehicle System Group (VSG) Mode*) together with ECU-C file format to store the configuration, the use of Vector's DCM is mandatory.

4.2 Initialization

The DEM startup behavior conforms to [1].

The function `Dem_PreInit()` shall be used during the startup phase to prepare the internal states of the DEM.¹

The function `Dem_Init()` shall be used during the startup phase of the ECU after the NvRAM Manager has finished the restoration of NvRAM data.

¹ In multiple identity mode (see chapter 7.1) the active configuration must be chosen before calling `Dem_PreInit` by using the additional API `Dem_InitDiagnosticVariant()`. Details in chapter 6.2.2.2 *Dem_Init* and in chapter

**Caution**

If a new pre-compile configuration is used, the NvRAM must be initialized externally e.g. via `Dem_NvDataInit()`.

SW-Components including Monitor Functions are initialized afterwards.

**Caution**

Starting with DEM 2.14.00 you must take special care that the NvRAM block(s) were either correctly restored from a previously stored state or you (resp. the NvM) must call the corresponding initialization function before calling `Dem_Init()`.

Starting the Dem with corrupt or uninitialized data will trigger undefined behaviour.

The Dem will not check any more the block's `NvM_GetErrorStatus()` as this might result in unintentional data clearing when there were NV write problems during ECU sleep/wakeup sequence.

For important information please refer to chapter 6.4.1 *Dem_NvDataInit* and 8.2.11 *NvRam Block IDs*

4.3 States

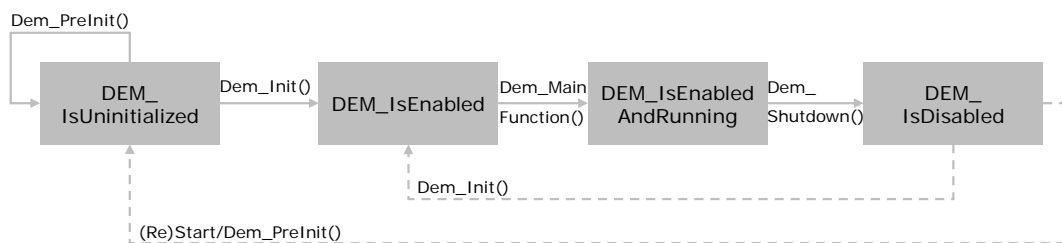


Figure 4-1 DEM States

After the (re)start of the ECU or the call of `Dem_PreInit()` the DEM is in state `DEM_IsUninitialized`. In this state, only BSW errors can be reported via `Dem_ReportErrorStatus()`.

During initialization via `Dem_Init()` the DEM switches to the state `DEM_IsEnabled`. The DTC storage is activated and the first task of `Dem_MainFunction()` is to execute (and empty) the BSW queue.

Thus, normal operation mode (`DEM_IsEnabledAndRunning`) is started, until `Dem_Shutdown()` will finalize all pending operations in the DEM and deactivate the DTC storage (`Dem_IsDisabled`).

4.4 Main Functions

The function `Dem_MainFunction()` is used to process all DEM internal functions. It shall be called periodically, e.g., every 10ms as cyclic task by the AUTOSAR Schedule Manager.

All actions of event based APIs (e.g. `Dem_SetEventStatus()` and `Dem_SetOperationCycleState()`) are logged synchronously to the BSW event queue and are executed during the next `Dem_MainFunction()` call. Thereby the logged sequence will be kept except for first call of the main function. In this case stored operation cycle changes will be brought forward to avoid that possibly reported BSW errors (`Dem_ReportErrorStatus()`) are discarded due their `OperationCycle` being `DEM_CYCLE_STATE_END` (default value after startup) and thus prohibiting the setting of the events.

The number of actions to be queued is limited by the configurable values `Bsw Error Buffer Size` and `Add Error Buffer Size`

The number of actions to be queued is limited by the configurable values `Bsw Error Buffer Size` and `Add Error Buffer Size` (see 8.2.9 for more details).

Functions logged in queue (state: *DEM_IsEnabled*):

`Dem_ReportErrorStatus()`
`Dem_SetEventStatus()`
`Dem_ClearDTC()`
`Dem_EnableDTCStorage()`

Logged in queue for Operation cycles:

`Dem_SetOperationCycleState()`



Info

To notify the application that the queue is full and the API could not be added

- for the APIs `Dem_SetEventStatus()`, `Dem_ResetEventStatus()` and `Dem_SetOperationCycleState()` an additional return value is implemented:
 Without RTE: `DEM_E_QUEUE_OVERFLOW` (0x0F)
 With RTE: `RTE_E_DiagnosticMonitor_DEM_E_QUEUE_OVERFLOW` (15u)
 `RTE_E_OperationCycle_DEM_E_QUEUE_OVERFLOW` (15u)
- the value `DEM_DTC_PENDING` is returned for the APIs `Dem_ClearDTC()` and `Dem_EnableDTCStorage()`. So it is not necessary to implement an additional return value for these APIs which are called by the DCM.

a development error will be reported for the API `Dem_ReportErrorStatus()`:

ErrorId: `DEM_E_QUEUE_OVERFLOW` (0x0F)

**Info**

The return value and the development error DEM_E_QUEUE_OVERFLOW are extensions to AUTOSAR.

4.5 Event De-bouncing

For event de-bouncing the DEM supports two different mechanisms which are described below. . Please refer to chapter 8.1.1 and 8.2.12.5.2 for configuration details.

4.5.1 Via Counter Based Algorithm

Depended on the configured de-bounce step size, the DEM must be triggered actively multiple times by application, until an event will be qualified as passed or failed. The step sizes for count up and count down may have different values. An event will be qualified as 'failed', if the fault detection counter reaches the value 127. If the value -128 will be reached the event will be qualified as 'passed'. Figure 4-2 gives an overview of event de-bouncing by fault detection counter.

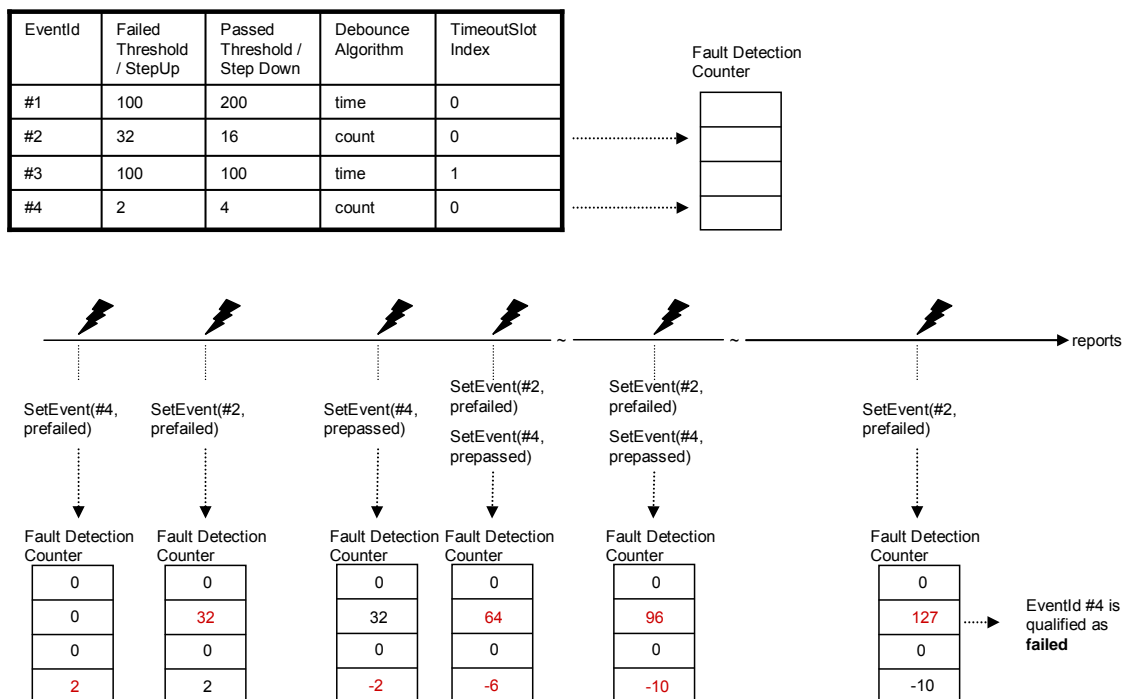


Figure 4-2 Event De-bouncing via counter based algorithm

4.5.2 Via Time Based Algorithm

Time based de-bounced events must be triggered only once by application to set a qualification direction. The event will be qualified after the configured de-bounce time has been elapsed. Multiple triggers for the same event and same qualification direction won't have any effects.

If an event which was reported as 'prefailed' and has not been qualified yet will be reported again as 'prepassed', the timer will be reloaded with the corresponding de-bounce timer value. Hence, once an event was reported the timer can only be stopped with "clear DTC"

or “restart of Operation cycle”. Figure 4-3 gives an overview of the time based de-bounce mechanism.

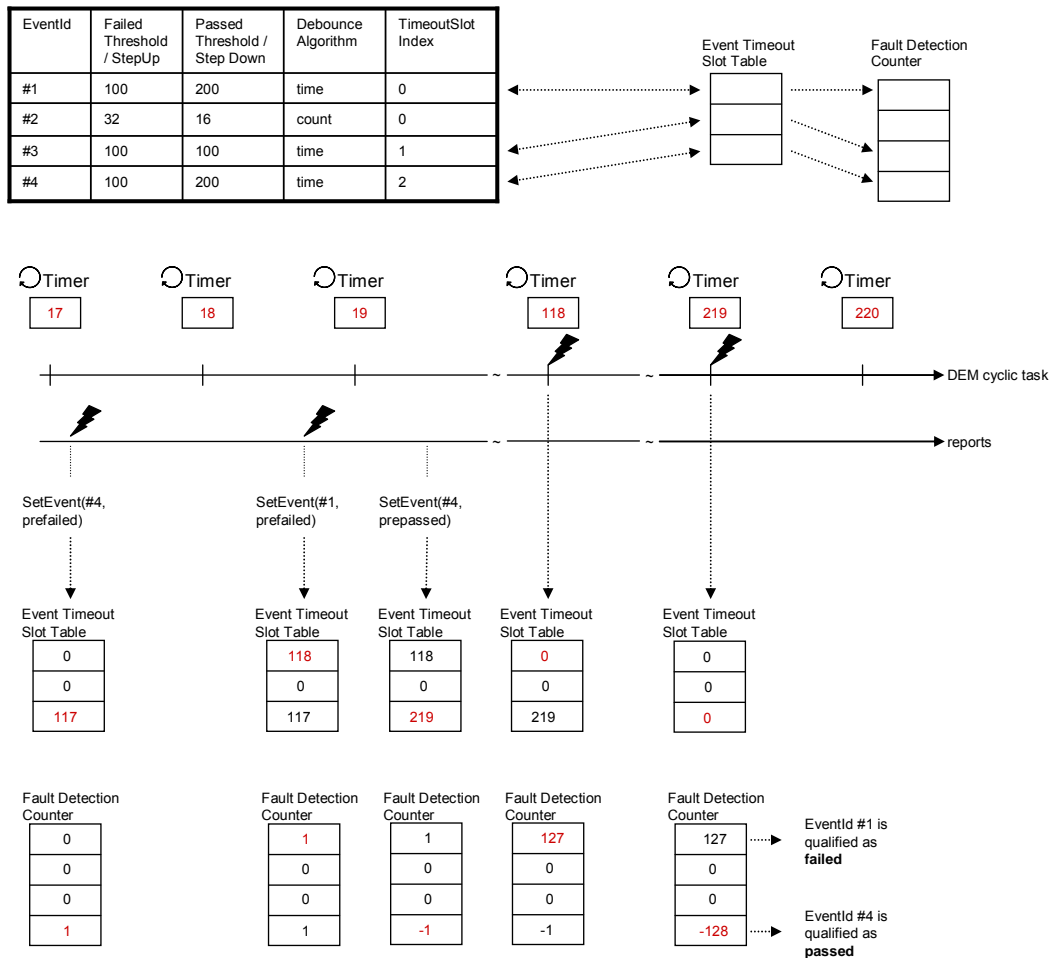


Figure 4-3 Event De-bouncing via time based algorithm

4.6 Error Handling

4.6.1 Development Error Reporting

Development errors are reported to DET using the service `Det_ReportError()`, when the pre-compile parameter `DEM_DEV_ERROR_DETECT` is set to `STD_ON`.

This parameter is preconfigured and automatically generated into `Dem_Cfg.h`. If activated, the file `Dem.c` will include the file `Det.h` to get the necessary prototype for above function (see [2] for more details).

This include is an extension to AUTOSAR's code/header file structure.

The reported `ModuleId` is `DEM_MODULE_ID` (54 dec).

The reported service IDs identify the services which are described in 6.2. The following table presents the service IDs, the related services and corresponding errors:

Service ID	Service	Error Code		Description
0x00	Dem_GetVersionInfo	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter <code>versioninfo</code> invalid
0x01	Dem_PreInit	0x10	DEM_E_PARAM_CONFIG	Vehicle System Group (VSG) mode: no configuration defined at all – given parameter is invalid
0x02	Dem_Init	0x10	DEM_E_PARAM_CONFIG	Invalid config: Dem_PBCfg.c was compiled with different size for type <code>Dem_EventIdType</code> (typedef'd by RTE)
		0x12	DEM_E_PARAM_DATA	Invalid config: Dem_PBCfg.c was generated for a different DEM version
0x03	Dem_Shutdown	-	-	-
0x04	Dem_SetEventStatus	0x12	DEM_E_PARAM_DATA	Invalid <code>EventId</code> or invalid <code>EventStatus</code>
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96	DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: <code>EventId</code> is not active in current configuration
0x05	Dem_ResetEventStatus	0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96	DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: <code>EventId</code> is not active in current configuration
0x06	Dem_PrestoreFreezeFrame	0x10	DEM_E_PARAM_CONFIG	<code>EventId</code> is not configured for prestoreing <code>FreezeFrame</code>
		0x12	DEM_E_PARAM_DATA	Invalid <code>EventId</code>
		0x96	DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: <code>EventId</code> is not active in current configuration
0x07	Dem_ClearPrestoredFreezeFrame	0x10	DEM_E_PARAM_CONFIG	<code>EventId</code> is not configured for prestoreing <code>FreezeFrame</code>
		0x12	DEM_E_PARAM_DATA	Invalid <code>EventId</code>
		0x96	DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: <code>EventId</code> is not active in current configuration

Service ID	Service	Error Code	Description
0x08	Dem_SetOperationCycleState	0x10 DEM_E_PARAM_CONFIG	OperationCycleId is not configured
		0x12 DEM_E_PARAM_DATA	Invalid CycleState
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x0A	Dem_GetEventStatus	0x12 DEM_E_PARAM_DATA	Invalid EventId
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration
0x0B	Dem_GetEventFailed	0x12 DEM_E_PARAM_DATA	Invalid EventId
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration
0x0C	Dem_GetEventTested	0x10 DEM_E_PARAM_CONFIG	Invalid EventId
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration
0x0D	Dem_GetDTCOfEvent	0x10 DEM_E_PARAM_CONFIG	Invalid EventId or DTC number not configured for EventId
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration
0x0E	Dem_GetSeverityOfDTC	0x11 DEM_E_PARAM_ADDRESS	Address for out parameter DTCSeverity invalid
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x0F	Dem_ReportErrorStatus	0x0F DEM_E_QUEUE_OVERFLOW	API service call could not be added to the action queue (overflow)
		0x12 DEM_E_PARAM_DATA	Invalid EventStatus or EventId
		0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration

Service ID	Service	Error Code		Description
0x13	Dem_SetDTCFilter	0x10	DEM_E_PARAM_CONFIG	Invalid argument(s)
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x14	Dem_SetViewFilter	-	-	-
0x15	Dem_GetStatusOfDTC	0x10	DEM_E_PARAM_CONFIG	Invalid DTCOrigin
		0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTCStatus invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x16	Dem_GetDTCStatusAvailabilityMask	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTCStatusMask invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x17	Dem_GetNumberOfFilteredDTC	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter NumberOfFilteredDTC invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x18	Dem_GetNextFilteredDTC	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTC or DTCStatus invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x19	Dem_GetDTCByOccurrenceTime	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTC invalid
		0x12	DEM_E_PARAM_DATA	Invalid DTCRequest or DTCKind
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x1A	Dem_DisableDTCRecordUpdate	0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x1B	Dem_EnableDTCRecordUpdate	0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x1C	Dem_GetDTCOfFreezeFrameRecord	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTC invalid
		0x12	DEM_E_PARAM_DATA	Invalid DTCOrigin or DTCKind
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x1D	Dem_GetFreezeFrameDataByDTC	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DestBuffer or inout parameter BufSize invalid
		0x12	DEM_E_PARAM_DATA	Invalid DTCOrigin or DTCKind
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized

Service ID	Service	Error Code	Description
0x1E	Dem_GetFreezeFrameDataIdentifierByDTC	0x11 DEM_E_PARAM_ADDRESS	Address for out parameter ArraySize or DataId invalid
		0x12 DEM_E_PARAM_DATA	Invalid DTCTOrigin or DTCTKind
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x1F	Dem_GetSizeOfFreezeFrame	0x11 DEM_E_PARAM_ADDRESS	Address for out parameter SizeOfFreezeFrame invalid
		0x12 DEM_E_PARAM_DATA	Invalid DTCTOrigin or DTCTKind
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x20	Dem_GetExtendedDataRecordByDTC	0x11 DEM_E_PARAM_ADDRESS	Address for out parameter DestBuffer or inout parameter BufSize invalid
		0x12 DEM_E_PARAM_DATA	Invalid DTCTOrigin or DTCTKind
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x21	Dem_GetSizeOfExtendedDataRecordByDTC	0x11 DEM_E_PARAM_ADDRESS	Address for out parameter SizeOfExtendedDataRecord invalid
		0x12 DEM_E_PARAM_DATA	Invalid DTCTOrigin or DTCTKind
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x22	Dem_ClearDTC	0x12 DEM_E_PARAM_DATA	Invalid DTCTOrigin or DTCTKind
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x24	Dem_DisableDTCStorage	0x12 DEM_E_PARAM_DATA	Invalid DTCTKind or DTCTGroup
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x25	Dem_EnableDTCStorage	0x12 DEM_E_PARAM_DATA	Invalid DTCTKind or DTCTGroup
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x26	Dem_DisableEventStatusUpdate	0x12 DEM_E_PARAM_DATA	Invalid DTCTKind or DTCTGroup
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x27	Dem_EnableEventStatusUpdate	0x12 DEM_E_PARAM_DATA	Invalid DTCTKind or DTCTGroup
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized

Service ID	Service	Error Code		Description
0x29	Dem_GetIndicatorStatus	0x93	DEM_E_TOO_MANY_INDICATOR_IDS	Too many indicators configured (> 65534)
0x36	Dem_GetViewIDofDTC	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter ViewId invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x37	Callback GetExtendedDataRecord{record-Number}	0x30	DEM_E_NODATAAVAILABLE	Configured function Xxx_DemGetExtendedDataRecord{recordNumber}() returned E_NOT_OK and DEM fills the record with dummy value 0xFF
0x38	Dem_SetValueByOemId	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DataValue invalid
0x39	Dem_SetEnableCondition	-	-	-
0x3A	Dem_GetNextFilteredRecord	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTC or SnapshotRecord invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x3B	Dem_GetNextFilteredDTCAndFDC	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTC or DTCFaultDetectionCounter invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x3C	Dem_GetTranslationType	0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x3D	Dem_GetNextFilteredDTCAndSeverity	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter DTC, DTCStatus, DTCSeverity or DTCFunctionalUnit invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
0x3E	Dem_GetFaultDetectionCounter	0x10	DEM_E_PARAM_CONFIG	Invalid EventId
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized
		0x96	DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration
0x3F	Dem_SetDTCFilterForRecords	0x11	DEM_E_PARAM_ADDRESS	Address for out parameter NumberOfFilteredRecords invalid
		0x20	DEM_E_UNINIT	API service called before the DEM module has been initialized

Service ID	Service	Error Code	Description
0x55	Dem_MainFunction	0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: a module called API Dem_SetEventStatus() or API Dem_ReportErrorStatus() with an EventId, which is not active in current configuration

Table 4-3 Mapping of Service IDs to Services and Corresponding Errors

The reported service IDs identify the services which are described in 6.2. The following table presents the service IDs of APIs not defined by AUTOSAR, the related services and corresponding errors:

Service ID	Service	Error Code	Description
0xE1	Dem_GetOccurrenceCounter	0x10 DEM_E_PARAM_CONFIG	Invalid EventId
		0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0xFE	Dem_InitDiagnosticVariant	0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: API was called twice or API was called after calling Dem_PreInit()

Table 4-4 Mapping of additional Service IDs to Services and Corresponding Errors

The errors reported to DET are described in the following table:

Error Code	Description
0x10 DEM_E_PARAM_CONFIG	API service called with wrong parameter – configuration error
0x11 DEM_E_PARAM_ADDRESS	API service called with invalid address
0x12 DEM_E_PARAM_DATA	API service called with wrong parameter – invalid value
0x13 DEM_E_PARAM_LENGTH	API service called with wrong parameter – invalid length
0x20 DEM_E_UNINIT	API service called before the DEM module has been initialized
0x30 DEM_E_NODATAAVAILABLE	No valid data available by the SW-C
0x96 DEM_E_NOT_IN_ACTIVE_VARIANT	Vehicle System Group (VSG) mode: EventId is not active in current configuration

Table 4-5 Errors Reported to DET

Internal debug codes of the DEM use the range 0x80...0x9F to report errors to the DET:

Error Code		Description
0x82	DEM_E_DTC_NOT_FOUND	No DTC found for <code>EventId</code>
0x83	DEM_E_ERASING_EMPTY_STACK	No entry found in chrono stack / primary memory
0x86	DEM_E_INV_CSTACK_REF	Invalid chrono stack / primary memory reference
0x89	DEM_E_INV_SFN	<code>EventId</code> beyond configured range
0x8F	DEM_E_INV_SNAPSHOT_NUM	Snapshot record number / freeze frame record number beyond configured range
0x90	DEM_E_ZEROPTR_SOURCE	Source address for moving data NULL
0x91	DEM_E_ZEROPTR_TARGET	Target address for moving data NULL
0x92	Dem_E_TOO_MANY_DIDS_FOR_EVENT	Number of DIDs for event exceeds maximum
0x93	DEM_E_TOO_MANY_INDICATOR_IDS	Too many indicators configured
0x95	DEM_E_INV_TIMER_SLOT_VAL	Invalid timer slot value for the timer based de-bounce algorithm

Table 4-6 Errors Reported to DET – Internal Debug Codes

4.6.1.1 Parameter Checking

AUTOSAR requires that API functions check the validity of their parameters. The checks in Table 4-7 are internal parameter checks of the API functions. AUTOSAR standard requires to en-/disable the complete parameter checking via the parameter `DEM_DEV_ERROR_DETECT`.

The following table shows which parameter checks are performed on which services:

Service	Check	Dem_InitializationStatus	EventId	EventStatus	versioninfo	nvBlockStatus	Dem_SizeofEventIdType	Dem_PBConfig.Dem_PBCfgLeader	Dem_PBConfig.Dem_PBCfgTrailer	PrestoreFFBufferOffset	OperationCycleId	CycleState
Dem_GetVersionInfo					■							
Dem_PreInit												
Dem_Init						■	■	■	■			
Dem_Shutdown												
Dem_SetEventStatus		■	■	■								
Dem_ResetEventStatus		■										
Dem_PrestoreFreezeFrame			■							■		
Dem_ClearPrestoredFreezeFrame			■							■		
Dem_SetOperationCycleState		■									■	■
Dem_GetEventStatus		■	■									
Dem_GetEventFailed		■	■									

Service	Check	Dem_InitializationStatus	EventId	EventStatus	versioninfo	nvBlockStatus	Dem_SizeofEventIdType	Dem_PBConfig.Dem_PBCfgLeader	Dem_PBConfig.Dem_PBCfgTrailer	PrestoreFFBufferOffset	OperationCyclcId	CycleState
Dem_GetEventTested		■	■									

Service	Check	Dem_InitializationStatus	EventId	EventStatus	DTCKind	DTCOrigin	DTCSeverity	DTC	DataValue	indicatorRef	FilterWithSeverity	FilterForFaultDetectionCounter
Dem_GetDTCOfEvent		■	■					■				
Dem_SetValueByOemId									■			
Dem_SetEnableCondition												
Dem_GetFaultDetectionCounter		■	■									
Dem_GetIndicatorStatus										■		
Dem_ReportErrorStatus			■	■								
Dem_SetDTCFilter		■			■	■	■				■	■

Service	Check	Dem_InitializationStatus	DTC	DTCKind	DTCOrigin	DTCStatus	DTCRequest	NumberOfFilteredRecords	DTCStatusMask	NumberOfFilteredDTC	ViewId	SnapshotRecord
Dem_SetDTCFilterForRecords		■						■				
Dem_SetViewFilter												
Dem_GetStatusOfDTC		■			■	■						
Dem_GetDTCStatusAvailabilityMask		■							■			
Dem_GetNumberOfFilteredDTC		■								■		
Dem_GetNextFilteredDTC		■	■			■						
Dem_GetDTCByOccurrenceTime		■	■	■			■					
Dem_GetViewIDOfDTC		■									■	
Dem_GetNextFilteredRecord		■	■									■

Service	Check	Dem_InitializationStatus	DTC	DTCKind	DTCOrigin	DTCStatus	DTCFaultDetectionCounter	DTCSeverity	DTCFunctionalUnit	DTCSeverity	DestBuffer	BufSize
Dem_GetNextFilteredDTCAndFDC		■	■				■					
Dem_GetNextFilteredDTCAndSeverity		■	■			■		■	■			
Dem_GetTranslationType		■										
Dem_GetSeverityOfDTC		■								■		
Dem_DisableDTCRecordUpdate		■										
Dem_EnableDTCRecordUpdate		■										
Dem_GetDTCOfFreezeFrameRecord		■	■	■	■							
Dem_GetFreezeFrameDataByDTC		■		■	■						■	■

Service	Check	Dem_InitializationStatus	DTCKind	DTCOrigin	DTCStatus	ArraySize	DataId	SizeOfFreezeFrame	DestBuffer	BufSize	SizeOfExtendedDataRecord	DTCGroup
Dem_GetFreezeFrameDataIdentifierByDTC		■	■	■		■	■					
Dem_GetSizeOfFreezeFrame		■	■	■				■				
Dem_GetExtendedDataRecordByDTC		■	■	■					■	■		
Dem_GetSizeOfExtendedDataRecordByDTC		■	■	■							■	
Dem_ClearDTC		■	■	■								
Dem_DisableDTCStorage		■	■									■
Dem_EnableDTCStorage		■	■									■
Dem_DisableEventStatusUpdate		■	■									■
Dem_EnableEventStatusUpdate		■	■									■

Table 4-7 Development Error Reporting: Assignment of Checks to Services



Caution

Using the DEM APIs with disabled DET and invalid parameters may result in an undefined behavior.

4.6.2 Production Code Error Reporting

DEM does not report any production code related errors.



Info

Production errors in general are errors which shall be saved through the DEM by definition. Errors of DEM itself occurring during normal operation are not saved.

5 Integration

This chapter gives necessary information for the integration of the MICROSAR DEM into an application environment of an ECU.

5.1 Scope of Delivery

The delivery of the DEM contains the files which are described in the chapters 5.1.1 and 5.1.2.

5.1.1 Static Files

File Name	Description
Dem.c	This is the source file of the DEM. It contains the main functionality of the DEM.
Dem.h	This is the header file of the DEM. It provides the DEM API functions to other modules.
Dem_Types.h	This is a header file of the DEM. It contains all DEM data types.

Table 5-1 Static Files of Delivery

5.1.2 Dynamic Files

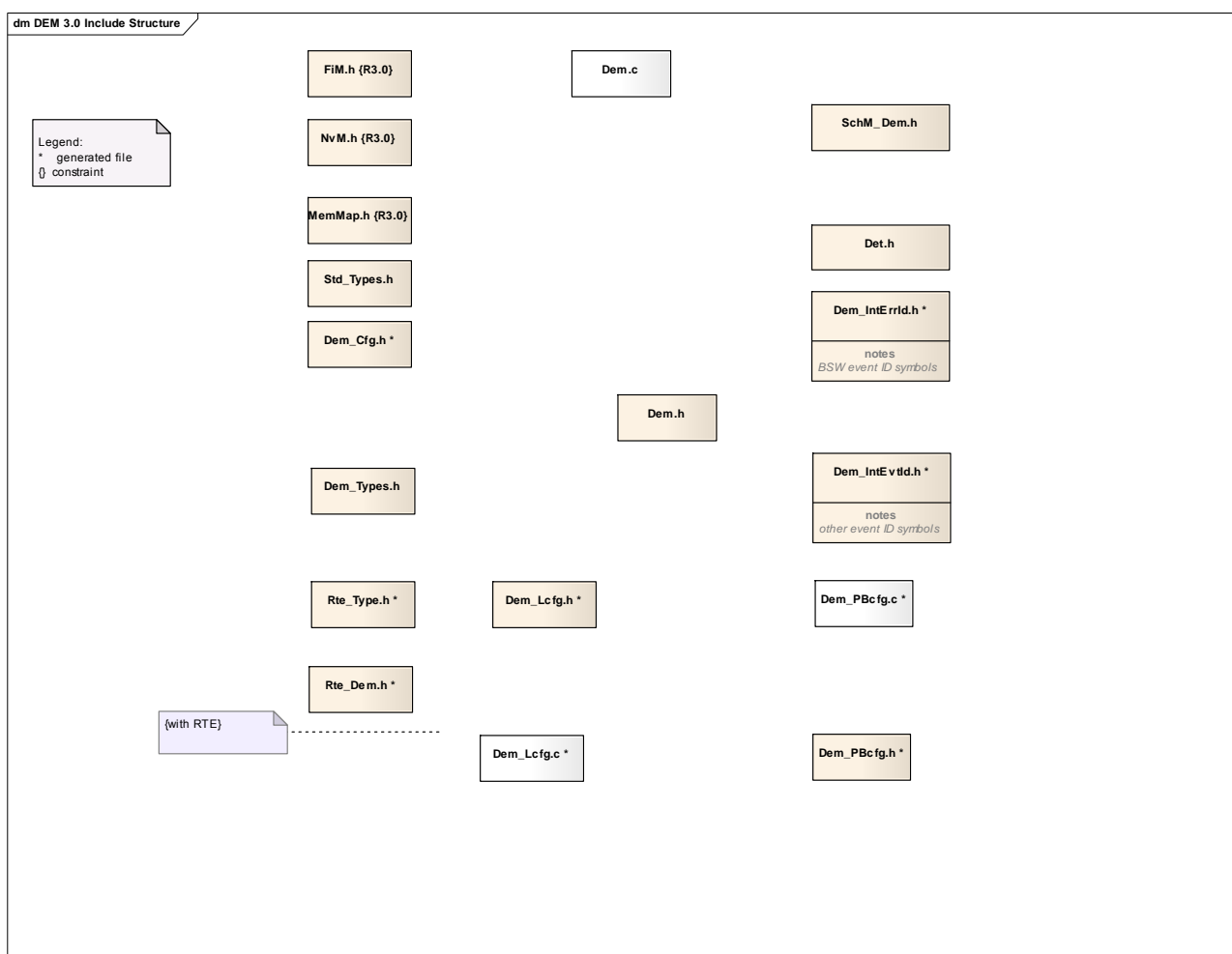
The dynamic files are generated by the configuration tool GENy.

File Name	Description
Dem_PBcfg.c	This is a source file of the generated part of the DEM. It contains the post-build configuration of the DEM.
Dem_PBcfg.h	This is a header file of the generated part of the DEM. It contains the post-build time configurable parameters.
Dem_Lcfg.c	This is a source file of the generated part of the DEM. It contains the link-time configuration of the DEM.
Dem_Lcfg.h	This is a header file of the generated part of the DEM. It contains the link-time configurable parameters.
Dem_Cbk.h	<p>This is a header file of the generated part of the DEM. It contains prototypes of callback functions offered by other components (replacement for DEM's ClientPort when no RTE available).</p> <p>- Only generated when RTE is not part of the configuration -</p>
Dem_swc.arxml	<p>This file is used for the RTE generation. It contains the information to get prototypes of callback functions offered by other components.</p> <p>- Only generated when RTE is part of the configuration -</p>

File Name	Description
Dem_Cfg.h	This is a header file of the generated part of the DEM. It contains the DEM precompiled parameters.
Dem_IntErrId.h	This is a header file of the generated part of the DEM. It contains the BSW event Id symbols.
Dem_IntEvtId.h	This is a header file of the generated part of the DEM. It contains the SW_C event Id symbols.

Table 5-2 Generated (Dynamic) Files of Delivery

5.2 Include Structure



5.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions defined for the DEM and illustrates their assignment among each other.

Compiler Abstraction Definitions	DEM_CODE	DEM_VAR_NOINIT_FAST	DEM_VAR	DEM_VAR_NOINIT	DEM_CONST	DEM_APPL_CONST	DEM_APPL_DATA	DEM_PBCFG
Memory Mapping Sections								
DEM_START_SEC_CODE	■							
DEM_STOP_SEC_CODE								
DEM_START_SEC_CONST_LCFG					■			
DEM_STOP_SEC_CONST_LCFG								
DEM_START_SEC_PBCONST								■
DEM_STOP_SEC_PBCONST								
DEM_START_SEC_CONST_8BIT					■			
DEM_STOP_SEC_CONST_8BIT								
DEM_START_SEC_CONST_32BIT					■			
DEM_STOP_SEC_CONST_32BIT								
DEM_START_SEC_CONST_UNSPECIFIED					■	■		
DEM_STOP_SEC_CONST_UNSPECIFIED								
DEM_START_SEC_VAR_8BIT			■					
DEM_STOP_SEC_VAR_8BIT								
DEM_START_SEC_VAR_FAST_8BIT		■						
DEM_STOP_SEC_VAR_FAST_8BIT								
DEM_START_SEC_VAR_NOINIT_8BIT				■				
DEM_STOP_SEC_VAR_NOINIT_8BIT								
DEM_START_SEC_VAR_NOINIT_16BIT				■				
DEM_STOP_SEC_VAR_NOINIT_16BIT								
DEM_START_SEC_VAR_NOINIT_UNSPECIFIED				■				
DEM_STOP_SEC_VAR_NOINIT_UNSPECIFIED								
DEM_START_SEC_VAR_SAVED_ZONE0_UNSPECIFIED							■	
DEM_STOP_SEC_VAR_SAVED_ZONE0_UNSPECIFIED								

Table 5-3 Compiler Abstraction and Memory Mapping

5.4 Critical Sections

To protect internal data structures against unwanted modification, the DEM uses “Critical Sections” for blocking concurrent access. Typically the function `Dem_EnterCritical()` disables interrupt or the task switching and `Dem_LeaveCritical()` re-enables it again.

Dependent on the customer's requirements, the delivery uses one of the following methods to handle the critical sections (the evaluation bundle has implemented AUTOSAR Schedule Manager). Look in file `Dem.c` section “INCLUDES” for your solution.

- Vector standard library (`VStdLib.h` is included)
- AUTOSAR Schedule Manager (`SchM_Dem.h` is included)
- Disable interrupts in CAN-Driver (`can.h` is included)

There are two issues you should take special care with the critical sections: first the startup phase of the ECU, when the implementation of the `EnterCritical/LeaveCritical` functions must be operable i.e. the component implementing them (e.g. AUTOSAR Schedule Manager) must be already started. Second, when the DEM uses callback functions (e.g. `Rte_Call_Dem_<ConfiguredName>_EventStatusChanged()`), as these callbacks are triggered inside the critical section.

5.4.1 Startup Phase

In AUTOSAR's “STARTUP_1” phase DEM's pre-init function `Dem_PreInit()` is called before starting the OS. Now DEM is able to collect events from other SW-C modules via `Dem_ReportErrorStatus()` and store them in a queue. This queue is protected against concurrent access by using a critical section.

This critical section must be handled specially. A typical implementation for `EnterCritical()` uses methods of the OS (semaphore, mutex, interrupt blocking, ...) but at this time during startup the OS isn't initialized and is not running yet!

Most times, this is no problem as there is no task switch possible and the interrupts are disabled (until in STARTUP_2 phase `Dem_Init()` is called and the queue is switched off), so the function `Dem_ReportErrorStatus()` and therefore the contained critical section can never be interrupted. In this case you could use an empty implementation for the `Enter/LeaveCritical` function.

But your startup may differ, thus this critical section handling is implemented as `Dem_EnterCritical_PreInit()` / `Dem_LeaveCritical_PreInit()` and when using the AUTOSAR Schedule Manager these macros are mapped to the `DEM_EXCLUSIVE_AREA_1`.

All other critical section handling is implemented as `Dem_EnterCritical()` / `Dem_LeaveCritical()` and when using the AUTOSAR Schedule Manager these macros are mapped to the `DEM_EXCLUSIVE_AREA_0`.



Caution

Check your startup sequence and take special care for the correct handling of critical sections between `Dem_PreInit()` and `Dem_Init()`.

5.4.2 Call of Callback Functions

When calling `Dem_SetEventStatus()` the DEM updates the status of the event during the next call of `Dem_MainFunction()`. It uses the configured (callback-) functions to get the environmental data to fill the extended data and snapshot record. Finally, the DEM calls configured (callback-) functions to notify the status change to other components. A similar sequence is executed when events are deleted or a new operation cycle is started – thereby after signalling the status change (as written above) some configured callbacks are used to trigger `InitMonitor` functions.

It is almost always fatal to call these callback functions with disabled interrupts, e.g. inside a critical section. Therefore the critical section is left before the callback and re-entered afterwards. Meanwhile the data is protected against modification by blocking the status update (see API `Dem_DisableEventStatusUpdate()` chapter 6.2.5.4.3).



Caution

Neither call DEM API functions directly nor indirectly from inside callback functions, which are triggered by the DEM itself.

5.5 Call Context

The DEM uses critical sections (see chapter 5.4) to protect against simultaneous modification of internal data structures.

The DEM uses many callback functions of other modules when collecting its data. E.g. for setting an event, it will call (potentially several times – dependent on the configuration) `Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier()` and `Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord()` and `Rte_Call_Dem_<ConfiguredName>_EventStatusChanged()` and afterwards `Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged()`.

All these callback calls will run in the call context of the caller of `Dem_SetEventStatus()`!

If one of the callbacks would directly or indirectly call `Dem_SetEventStatus()` again, we get an recursive call tree, that might cause a deadlock situation – dependent on the implementation of the protection of critical sections.

To protect against this, the action to store data is executed during the next call of `Dem_MainFunction()` as described in chapter 4.4.

To get data consistency of DEM and user data without globally disabling of interrupts, you must

- > not call DEM functions from inside a callback function, which was triggered from DEM,
- > not call different APIs that will access the same `EventId`.

**Caution**

Do link each Diagnostic Monitor only once to one client

You must not call different operations of a Diagnostic Monitor port interface *for the same event* simultaneously.

You must not call the same operation of a Diagnostic Monitor port interface *for the same event* simultaneously.

You must not use any operation of a Diagnostic Monitor port interface for an event that is modified by a BSW module using API `Dem_ReportErrorStatus`.

5.6 Scheduling of DEM and DCM

To get the data for the external tester responses, the DCM accesses APIs of the DEM. To avoid simultaneous data access, the DCM must not interrupt `Dem_MainFunction()`.

**Info**

Possible implementation:

- Configure `Dcm_StateTask()` to the same priority as `Dem_MainFunction()` and make sure, they cannot interrupt each other (e.g. use cooperative tasks or put both functions as sequence in a common SchM task)

Optionally: Configure `Dcm_TimerTask()` with higher priority

The API functions of the interface DCM – DEM must be called always from same context. Do not change between interrupt context / task context ... This constraint will be always fulfilled if you're using DCM from Vector.

5.7 NvRAM Demand

All data which shall be stored in the NvRAM are handled with the object `Dem_NvData` of the struct `Dem_NonVolatileDataType`. To get NvRAM demand (data structure size) of the DEM, temporarily add `sizeof(Dem_NvData)` to the source and read value after compilation (compiler and configuration dependent).

**Info**

There are pending discussions about the NVRAM manager interface, due to an incomplete SWS specification. Final solution is postponed to AUTOSAR phase II.

Vector's solution expects an updated RAM mirror before Dem_Init() is called, see chapter 4.2 and an automatic NVRAM updated after Dem_Shutdown() call² (In Dem_Shutdown() the DEM marks its NvRAM Block IDs as 'Modified' by calling API NvM_SetRamBlockStatus(). The following system shutdown sequence now will save the data by calling NvM_WriteAll().).

² As specified by Dem170 and Dem102

6 API Description

For an interfaces overview see Figure 3-2.

6.1 Type Definitions

The types defined by the DEM are described in [1].

6.2 Services Provided by DEM

The DEM API consists of services, which are realized by function calls.

6.2.1 Dem_GetVersionInfo

Prototype		
<pre>void Dem_GetVersionInfo(Std_VersionInfoType * versioninfo)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x00	Synchronous	Reentrant
Parameter		
versioninfo	Pointer to where to store the version information of this module.	
Return code		
None		
Functional Description		
Returns the version information of this module.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-1 Dem_GetVersionInfo



Info

In the description of this API in [1] the parameter `instanceID` is missing. The definition in [5] includes this parameter. The implementation uses the definition in [5].

6.2.2 Interface ECU State Manager ↔ DEM

6.2.2.1 Dem_PreInit

For a description of the purpose and the arguments of the function, see [1].

For supporting Multi Identity Mode (see chapter 7.1.3 *Vehicle System Group (VSG) Mode*) the API was modified:

Prototype		
Single Identity Mode		
void Dem_PreInit ()		
Multi Identity Mode or GENy setting: “Dem_PreInit() has config parameter”		
void Dem_PreInit (const Dem_ConfigType * ActiveConfig)		
Service ID (hex)	Sync/Async	Reentrancy
0x01	Synchronous	Non Reentrant
Parameter		
ActiveConfig	Pointer to one of the configuration sets declared in Dem_Lcfg.h. (The definition of the configurations is in Dem_Lcfg.c)	
Return code		
-	-	
Functional Description		
<p>If multiple configuration variants are supported, the base configuration variant parameter is expected by this API. The base variant object is declared in the Dem_Lcfg.h file in section “Config sets”. Its actual name depends on your CDD settings.</p> <p>The multi identity (VSG) mode will be active only if the generated switch DEM_SUPPORT_VARIANT_HANDLING in (generated) file Dem_Cfg.h is set to STD_ON.</p> <p>If multiple identity (VSG) mode is active, this API must be called with the generated ConfigSet parameter declared in Dem_Lcfg.h file, e.g.:</p> <pre>Dem_PreInit(&CommonDiagnostics);</pre>		
Particularities and Limitations		
<ul style="list-style-type: none">> To ease the development and permit the switch between single (or no) CDD variant and multiple CDD variant without API change, GENy offers the checkbox “<i>Dem_PreInit() has config parameter</i>”. By activating this checkbox, you will always get the <i>Multi Identity Mode</i> API. When using <i>Single Identity</i> the parameter is mandatory but silently ignored.> You can not change the active configuration any more after calling Dem_PreInit until next ECU power-on boot (or any reset situation). The active configuration consists of the base variant (chosen with Dem_PreInit(...)) and optional, additional variants (chosen prior with Dem_InitDiagnosticVariant(...), see chapter 6.2.2.4).		
Expected Caller Context		
<ul style="list-style-type: none">> Only within disabled global interrupt call context		

Table 6-2 Dem_PreInit

6.2.2.2 Dem_Init

Prototype		
void Dem_Init ()		
Service ID (hex)	Sync/Async	Reentrancy
0x02	Synchronous	Non Reentrant
Parameter		
None		
Return code		
None		
Functional Description		
Initializes this module.		
Particularities and Limitations		
<ul style="list-style-type: none"> > The Dem assumes, that all non-volatile data was restored (or initialized) before calling Dem_Init(). Typically this is automatically done during the startup of the ECU. Please see chapter 4.2 for an overview and chapter 6.4 for correct configuration. > If enabled Dem_Init() will use some patterns ("magic numbers") to evaluate, if the non-volatile data blocks of the Dem are initialized and completely restored. If not, the Dem will trigger the initialization of the buffers itself, see Dem_NvData...() callbacks in chapter 6.4. 		
Expected Caller Context		
<ul style="list-style-type: none"> > Task context 		

Table 6-3 Dem_Init

See [4] for more details of the NvM APIs.

6.2.2.3 Dem_Shutdown

Prototype		
void Dem_Shutdown ()		
Service ID (hex)	Sync/Async	Reentrancy
0x03	Synchronous	Non Reentrant
Parameter		
None		
Return code		
None		
Functional Description		
Shutowns this module.		

Particularities and Limitations
> None
Expected Caller Context
> Task context

Table 6-4 Dem_Shutdown

During shutdown, the DEM calls `NvM_SetRamBlockStatus()` to change the state of the NvRAM BlockIDs which have been modified. The system shutdown sequence will save the labelled data by calling `NvM_WriteAll()`.

See [4] for more details of the NvM APIs. Configuration see chapter 8.2.11.

6.2.2.4 Dem_InitDiagnosticVariant

Multi-Identity Support is a Vector extension for DCM and DEM to support multiple variants in an ECU. On startup, all events of the basic variant are available plus additional the configured events (DTCs) of 0-30 additional variants.

Before calling `Dem_PreInit()`, the additional variants are selected by calling `Dem_InitDiagnosticVariant()`.

Prototype	
Multi Identity Mode or GENy setting: "Dem_PreInit() has config parameter"	
<pre>void Dem_InitDiagnosticVariant (uint32 activeCfg)</pre>	
Parameter	
activeCfg	A bit-mapped value where each set bit represents an active diagnostic configuration.
Return code	
-	-

Functional Description

If a combination of multiple diagnostic configurations is supported, the application can activate, additionally to the base variant, any combination of the other diagnostic configurations that are available.

All configured configuration variant parameter values are located in the Dem_Lcfg.h file in section **"Multiple Variant Initialization"**, with the naming convention:

DEM_CFGVARIANT_<VARIANT_QUALIFIER>. If you use a CDD file for the DEM configuration, then the variant qualifiers are the qualifier of the vehicle system groups defined in the CDD file.

Note:

If the base variant only shall be supported, you can omit the call of this API. DEM always sets the base variant, when calling Dem_PreInit(...) later with argument value 'CommonDiagnostics'.

Usage Example:

```
#include "Dem_Lcfg.h"
```

```
void SysStartupPhase1(void)
{
    /* Compose supported diagnostic configurations */
    uint32 demAddVariantMask = 0L;

    if (<EcuFeatureA_enabled>)
    {
        demAddVariantMask |= DEM_CFGVARIANT_<CDD_VSG_FEATRUREA>;
    }

    if (<EcuFeatureB_enabled>)
    {
        demAddVariantMask |= DEM_CFGVARIANT_<CDD_VSG_FEATRUREB>;
    }

    /* Set additional variants */
    Dem_InitDiagnosticVariant(demAddVariantMask);

    ...
    /* PreInit DEM now with base variant */
    Dem_PreInit(&<DemConfigSet from Dem_Lcfg.h>);
}
```

Particularities and Limitations

- > Must be called prior Dem_PreInit(...) API.
- > Can only be called once. A second call is ignored.
- > You can not change the active configuration any more after calling Dem_PreInit() until next ECU power-on boot (or any reset situation).

With a reset, the following sequence will be executed:

1. Compiler startup code or Dem_InitMemory() is called.
2. Dem_InitDiagnosticVariant(...) is called
3. Dem_PreInit(...) is called

Call context

- > Only within disabled global interrupt call context.

Table 6-5 Dem_InitDiagnosticVariant

6.2.3 Interface SW-Components via RTE ⇔ DEM

6.2.3.1 Dem_SetEventStatus

Prototype

```
Std_ReturnType Dem_SetEventStatus (
    Dem_EventIdType EventId,
    uint8           EventStatus
)
```

Service ID (hex)	Sync/Async	Reentrancy
0x04	Synchronous	Reentrant

Parameter

EventId	<p>Identification of an Event by assigned EventId. The EventId is configured in the DEM.</p> <p>Min.: 1 (0: Indication of no Event)</p> <p>Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)</p>
EventStatus	<p>uint8</p> <p>{DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED, DEM_EVENT_STATUS_PREPASSED, DEM_EVENT_STATUS_PREFAILED [, Custom]}</p>

Return code

Std_ReturnType	<p>E_OK: set of event status was successful</p> <p>E_NOT_OK: set of event status failed</p>
----------------	---

Functional Description

Queue the status of an event.

Particularities and Limitations

> Do not call this API for events, that are set by the BSW too (see chapter 5.5 *Call Context*).

Expected Caller Context

> Task context

Table 6-6 Dem_SetEventStatus

This API accepts internal events as well as external events.



Info

To notify the application whether it was successful to add this API to the action queue (see chapter 4.4) an additional return value is introduced:

Without RTE: DEM_E_QUEUE_OVERFLOW (0x0F)

With RTE: RTE_E_DiagnosticMonitor_E_NOT_OK (15u)

This return value is an extension to AUTOSAR.

6.2.3.2 Dem_ResetEventStatus

Prototype

```
Std_ReturnType Dem_ResetEventStatus (
    Dem_EventIdType EventId
)
```

Service ID (hex)	Sync/Async	Reentrancy
0x05	Synchronous	Reentrant

Parameter

EventId	<p>Identification of an Event by assigned EventId. The EventId is configured in the DEM.</p> <p>Min.: 1 (0: Indication of no Event or Failure)</p> <p>Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)</p>
---------	--

Return code

Std_ReturnType	<p>E_OK: reset of event status was successful</p> <p>E_NOT_OK: reset of event status failed</p>
----------------	---

Functional Description

Resets the Event Status stored in the Event Memory in the DEM.

Particularities and Limitations

> None

Expected Caller Context

> Task context

Table 6-7 Dem_ResetEventStatus

**Info**

To notify the application whether it was successful to add this API to the action queue (see chapter 4.4) an additional return value is introduced:

Without RTE: DEM_E_QUEUE_OVERFLOW (0x0F)

With RTE: RTE_E_DiagnosticMonitor_E_NOT_OK (15u)

This return value is an extension to AUTOSAR.

6.2.3.3 Dem_PrestoreFreezeFrame

Prototype		
Std_ReturnType Dem_PrestoreFreezeFrame (Dem_EventIdType EventId)		
Service ID (hex)	Sync/Async	Reentrancy
0x01	Synchronous	Reentrant
Parameter		
EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)	
Return code		
Std_ReturnType	E_OK PreStoreFreezeFrame was successful E_NOT_OK PreStoreFreezeFrame failed	
Functional Description		
Captures the FreezeFrame data for a specific EventId.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-8 Dem_PrestoreFreezeFrame

6.2.3.4 Dem_ClearPrestoredFreezeFrame

Prototype		
Std_ReturnType Dem_ClearPrestoredFreezeFrame (Dem_EventIdType EventId)		
Service ID (hex)	Sync/Async	Reentrancy
0x07	Synchronous	Reentrant
Parameter		
EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)	
Return code		
Std_ReturnType	E_OK: ClearPreStoreFreezeFrame was successful E_NOT_OK: ClearPreStoreFreezeFrame failed	
Functional Description		
Clears a prestored Freeze Frame		
Particularities and Limitations		
> None		
Expected Caller Context		
>		

Table 6-9 Dem_ClearPrestoredFreezeFrame

6.2.3.5 Dem_SetOperationCycleState

Prototype		
Std_ReturnType Dem_SetOperationCycleState (uint8 OperationCycleId, uint8 CycleState)		
Service ID (hex)	Sync/Async	Reentrancy
0x08	Synchronous	Non Reentrant
Parameter		
OperationCycleId	Identification of operation cycle, like power cycle, driving cycle.	
CycleState	DEM_CYCLE_STATE_END 0x02 End of operation cycle, DEM_CYCLE_STATE_START 0x01 Start of operation cycle	
Return code		
Std_ReturnType	E_OK: set of operation cycle was successful E_NOT_OK: set of operation cycle failed	

Functional Description
Sets an operation cycle state.
Particularities and Limitations
> None
Expected Caller Context
> Task context

Table 6-10 Dem_SetOperationCycleState

**Info**

To notify the application whether it was successful to add this API to the action queue (see chapter 4.4) an additional return value is introduced:

Without RTE: DEM_E_QUEUE_OVERFLOW (0x0F)

With RTE: RTE_E_OperationCycle_E_NOT_OK (15u)

This return value is an extension to AUTOSAR.

6.2.3.6 Dem_GetEventStatus

Prototype	
<pre>Std_ReturnType Dem_GetEventStatus (Dem_EventIdType EventId, uint8 * EventStatusExtended)</pre>	
Service ID (hex)	Sync/Async
0x0a	Synchronous
Reentrancy	
Reentrant	
Parameter	
EventId	<p>Identification of an Event by assigned EventId. The EventId is configured in the DEM.</p> <p>Min.: 1 (0: Indication of no Event)</p> <p>Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)</p>
EventStatusExtended	<p>Bit 0 TestFailed is set to 1 if the last event status update by the function Dem_SetEventStatus(Passed Failed) was called with failed. The status is set to 0 if Dem_SetEventStatus is called with passed, on tester clear command and by API Dem_ResetEventStatus.</p> <p>Bit 0 and 6 is intended to set/reset monitor inhibit or default.</p> <p>Bit 1 TestFailedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed failed) is called with failed this cycle. Intended to be used for defaults reset only at next key on.</p> <p>Bit 2 PendingDTC is set when associated DTC becomes available in Mode07 (currently corresponds to ISO pending bit 2 [9] Intended to be used for the control of IUMPR counters.</p> <p>Bit 3 ConfirmedDTC is set when associated DTC becomes available in Mode03 (currently corresponds to ISO confirmed bit 3 [9] Could be used to set e.g. service request message.</p> <p>Bit 4 TestNotCompletedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus (passed failed) is called after last ClearDTC.</p> <p>Bit 5 testFailedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus is caled with failed this cycle.</p> <p>Bit 6 TestNotCompletedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is used, the differentiation is obsolete).</p> <p>Bit 7 WarningIndicatorRequested reports the status of any warning indicators associated with a particular DTC.</p>
Return code	
Std_ReturnType	<p>E_OK: get of event status was successful</p> <p>E_NOT_OK: get of event status failed</p>
Functional Description	
Gets the extended event status of an event.	
Particularities and Limitations	
> None	

Expected Caller Context

> Task context

Table 6-11 Dem_GetEventStatus

6.2.3.7 Dem_GetEventFailed

Prototype

```
Std_ReturnType Dem_GetEventFailed (
    Dem_EventIdType EventId,
    boolean *       EventFailed
)
```

Service ID (hex)	Sync/Async	Reentrancy
0x0b	Synchronous	Reentrant

Parameter

EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)
EventFailed	TRUE - Last Failed FALSE - not Last Failed

Return code

Std_ReturnType	E_OK: get of "EventFailed" was successful E_NOT_OK: get of "EventFailed" was not successful
----------------	--

Functional Description

Gets the event failed status of an event.

Particularities and Limitations

> None

Expected Caller Context

> Task context

Table 6-12 Dem_GetEventFailed

6.2.3.8 Dem_GetEventTested

Prototype

```
Std_ReturnType Dem_GetEventTested (
    Dem_EventIdType EventId,
    boolean *       EventTested
)
```

Service ID (hex)	Sync/Async	Reentrancy
0x0c	Synchronous	Reentrant

Parameter	
EventId	<p>Identification of an Event by assigned EventId. The EventId is configured in the DEM.</p> <p>Min.: 1 (0: Indication of no Event)</p> <p>Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)</p>
EventTested	<p>TRUE - event tested this cycle</p> <p>FALSE - event not tested this cycle</p>
Return code	
Std_ReturnType	<p>E_OK: get of event state "tested" successful</p> <p>E_NOT_OK: get of event state "tested" failed</p>
Functional Description	
Gets the event tested status of an event.	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-13 Dem_GetEventTested

6.2.3.9 Dem_GetDTCOfEvent

Prototype		
Std_ReturnType Dem_GetDTCofEvent (Dem_EventIdType EventId, uint8 DTCKind, uint32 * DTCofEvent)		
Service ID (hex)	Sync/Async	Reentrancy
0x01	Synchronous	Reentrant
Parameter		
EventId	Identification of an Event by assigned EventId. The EventId is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of EventId's in DEM (Max is either 255 or 65535)	
DTCKind	This parameter defines the requested DTC, either only OBDrelevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
DTCofEvent	Receives the DTC value returned by the function. If the return value of the function is other than E_OK this parameter does not contain valid data.	
Return code		
Std_ReturnType	E_OK: get of DTC was successful DEM_GET_DTCCOFEVENT_WRONG_DTCKIND: 0x04 DTC kind wrong, DEM_GET_DTCCOFEVENT_WRONG_EVENTID: 0x05 Wrong EventId	
Functional Description		
Gets the DTC of an event.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-14 Dem_GetDTCOfEvent

6.2.3.10 Dem_SetValueByOemId

This optional API is not supported by Vector specific DEM (it is implemented, but returns always E_NOT_OK).

6.2.3.11 Dem_SetEnableCondition

This optional API is not supported by Vector specific DEM (it is implemented, but returns always E_NOT_OK).

6.2.3.12 Dem_GetFaultDetectionCounter

Prototype		
Std_ReturnType Dem_GetFaultDetectionCounter (Dem_EventIdType EventId, sint8 * EventIdFaultDetectionCounter)		
Service ID (hex)	Sync/Async	Reentrancy
0x3e	Synchronous	Non Reentrant
Parameter		
EventId	Provide the EventId value the fault detection counter is requested for. If the return value of the function is other than OK this parameter does not contain valid data.	
EventIdFaultDetection-Counter	This parameter receives the fault detection counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data. -128dec...127dec PASSED... FAILED according to ISO 14229-1	
Return code		
Std_ReturnType	E_OK: request of fault detection counter was successful E_NOT_OK: request of fault detection counter failed	
Functional Description		
Gets the fault detection counter of an event.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-15 Dem_GetFaultDetectionCounter

6.2.3.13 Dem_GetIndicatorStatus

Prototype		
<pre>Std_ReturnType Dem_GetIndicatorStatus (uint8 IndicatorId, uint8 * IndicatorStatus)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x29	Synchronous	Non Reentrant
Parameter		
IndicatorId	Number of indicator	

IndicatorStatus	Status of the indicator, like on, off, blinking. DEM_INDICATOR_BLINK_CONT 0x03 Continuous and blinking mode, DEM_INDICATOR_CONTINUOUS 0x01 Continuous on, DEM_INDICATOR_OFF 0x00 Indicator off, DEM_INDICATOR_BLINKING 0x02 blinking mode
Return code	
Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed or is not supported
Functional Description	
Gets the indicator status derived from the event status (Bit TestFailed) and the configured indicator states (see 8.2.12.5.5 to see how to configure indicators). It will be reported as a summary of the single indicator states (usually an indicator is assigned to different events).	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-16 Dem_GetIndicatorStatus

**Info**

When the RTE is part of the software, only valid IndicatorIDs can be used when calling `Dem_GetIndicatorStatus()`.

Without RTE, the function `Dem_GetIndicatorStatus()` will accept all IDs (result: `E_OK`) and returns the indicator status 0 ("not active") not only when no event is "failed", but also if no event was assigned to this IndicatorID. Such "Invalid" IndicatorIDs (result: `E_NOT_OK`) could be detected by expensive search only, as the ID values are post-buildable. This search is not implemented.

6.2.3.14 Dem_GetOccurrenceCounter

Prototype	
<pre>Std_ReturnType Dem_GetOccurrenceCounter (Dem_EventIdType EventId, uint8 * OccurrenceCounterValue)</pre>	
Parameter	
EventId	Provide the EventId value the occurrence counter is requested for. If the return value of the function is other than E_OK this parameter does not contain valid data.
OccurrenceCounterValue	This parameter receives the Occurrence Counter information of the requested EventId. If the return value of the function call is other than E_OK this parameter does not contain valid data.
Return code	
Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Functional Description	
Read the DEM internal implemented OccurrenceCounter of an event. The API shall be used to request the current Occurrence Counter for a given EventId.	
Particularities and Limitations	
> This API is an extension to AUTOSAR.	
Expected Caller Context	
> Task context	

Table 6-17 Dem_GetOccurrenceCounter



Info

This API is an extension to the AUTOSAR description. It is available, when a CDD is used which supports the internally implemented occurrence counter (see 8.1.3.1).

6.2.4 Interface BSW-Components ↔ DEM

6.2.4.1 Dem_ReportErrorStatus

Prototype		
void Dem_ReportErrorStatus (Dem_EventIdType EventId, uint8 EventStatus)		
Service ID (hex)	Sync/Async	Reentrancy
0x01	Synchronous	Reentrant
Parameter		
EventId	Identification of an Event by assigned Event ID. The Event ID is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: Result of configuration of Event IDs in DEM (Max is either 255 or 65535)	
EventStatus	uint8 {DEM_EVENT_STATUS_PASSED, DEM_EVENT_STATUS_FAILED, DEM_EVENT_STATUS_PREPASSED, DEM_EVENT_STATUS_PREFAILED [, Custom]}	
Return code		
None		
Functional Description		
Reports errors to the DEM.		
Particularities and Limitations		
> Do not call this API for events, that are set via RTE/application too (see chapter 5.5 <i>Call Context</i>)		
Expected Caller Context		
> Task context		
> If interrupt context can not be avoided, please ensure, that both critical sections can be called within interrupt context (see chapter 5.4)		

Table 6-18 Dem_ReportErrorStatus

This API accepts internal events as well as external events.



Info

To notify the DET whether it was successful to add this API to the action queue (see chapter 4.4) an additional development error is implemented:

ErrorId: DEM_E_QUEUE_OVERFLOW (0x0F)

This development error is an extension to AUTOSAR.

6.2.5 Interface DCM ⇔ DEM

A further description of the usage of the interface between DCM (diagnostic communication manager) and DEM can be found in chapter “Error Manager Interface” of [3]. Here, especially the handling of FreezeFrame data is described.

6.2.5.1 Access DTCs and Status Information

The following chapter defines the API calls that shall be used to access the number of DTCs as well as the DTCs matching specific filter criteria and the associated status information.

6.2.5.1.1 Dem_SetDTCFilter

Prototype		
<pre>Dem_ReturnSetDTCFilterType Dem_SetDTCFilter (uint8 DTCStatusMask, uint8 DTCKind, uint8 DTCOrigin, uint8 FilterWithSeverity, uint8 DTCSeverity, uint8 FilterForFaultDetectionCounter)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x13	Synchronous	Non Reentrant
Parameter		
DTCStatusMask	According ISO14229-1 StatusOfDTC Values: 0x00: Report all supported DTCs 0x01...0xFF: Match DTCStatusMask as defined in ISO14229-1	
DTCKind	Defines the functional group of DTCs to be reported (e.g. all DTC, OBDrelevant DTC) DEM_DTC_KIND_ALL_DTCS selects all DTCs, DEM_DTC_KIND_EMISSION_REL_DTCS selects OBD-relevant DTCs	
DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.	
FilterWithSeverity	This flag defines whether severity information (ref. to parameter below) shall be used for filtering. This is to allow for coexistence of DTCs with and without severity information. DEM_FILTER_WITH_SEVERITY_YES 0x00 Severity information used, DEM_FILTER_WITH_SEVERITY_NO 0x01 Severity information not used	
DTCSeverity	This parameter contains the DTCSeverityMask according to ISO14229-1. DEM_SEVERITY_CHECK_IMMEDIATELY 0x80 Check immediately, DEM_SEVERITY_NO_SEVERITY 0x00 No severity information available, DEM_SEVERITY_CHECK_AT_NEXT_HALT 0x40 check at next halt, DEM_SEVERITY_MAINTENANCE_ONLY 0x20 maintenance required	

FilterForFaultDetection-Counter	<p>This flag defines whether Fault Detection Counter information shall be used for filtering. This is to allow for coexistence of DTCs with and without Fault Detection Counter information. If Fault Detection Counter information is filter criteria, only those DTCs with a Fault Detection Counter value between 1 and 0x7E shall be reported.</p> <p>Remark: If the event does not use the debouncing inside DEM, then the DEM must request this information via Xxx_DemGetFaultDetectionCounter.</p> <p>DEM_FILTER_FOR_FDC_YES 0x00 Fault Detection Counter information used,</p> <p>DEM_FILTER_FOR_FDC_NO 0x01 Fault Detection Counter information not used</p>
Return code	
Dem_ReturnSetDTCFilterType	Status of the operation of type Dem_ReturnSetDTCFilterType
Functional Description	
Sets the filter mask over all DTCs.	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-19 Dem_SetDTCFilter

6.2.5.1.2 Dem_SetDTCFilterForRecords

Prototype		
Dem_ReturnSetDTCFilterType Dem_SetDTCFilterForRecords (uint16 * NumberOfFilteredRecords)		
Service ID (hex)	Sync/Async	Reentrancy
0x3f	Synchronous	Non Reentrant
Parameter		
NumberOfFilteredRecords	Number of snapshot records currently stored in the event memory.	
Return code		
Dem_ReturnSetDTCFilter-Type	Status of the operation of type Dem_ReturnSetDTCFilterType	
Functional Description		
Sets DTC Filter for records.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-20 Dem_SetDTCFilterForRecords

6.2.5.1.3 Dem_SetViewFilter

This optional API is not supported by Vector specific DEM (it is implemented, but returns always DEM_WRONG_ID).

6.2.5.1.4 Dem_GetStatusOfDTC

Prototype		
<pre>Dem_ReturnGetStatusOfDTCType Dem_GetStatusOfDTC (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin, uint8 * DTCStatus)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x15	Synchronous	Non Reentrant
Parameter		
DTC	For this DTC its status is requested	
DTCKind	<p>This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs</p> <p>DEM_DTC_KIND_ALL_DTCS Select all DTCs</p> <p>DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs</p>	
DTCOrigin	<p>If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.</p> <p>DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory,</p> <p>DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory,</p> <p>DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.</p>	
DTCStatus	<p>This parameter receives the status information of the requested DTC. If the return value of the function call is other than DEM_STATUS_OK this parameter does not contain valid data.</p> <p>0x00...0xFF match DTCStatusMask as defined in ISO14229-1</p>	
Return code		
Dem_ReturnGetStatusOf-DTCType	Status of the operation of type Dem_ReturnGetStatusOfDTCType.	
Functional Description		
Gets the status of a DTC		
Particularities and Limitations		
> None		
Expected Caller Context		

> Task context

Table 6-21 Dem_GetStatusOfDTC

If the DTC is not stored in one of the available event memories the parameter DTCOrigin is neglected e.g. when DTC status is pending.

6.2.5.1.5 Dem_GetDTCStatusAvailabilityMask

Prototype		
Std_ReturnType Dem_GetDTCStatusAvailabilityMask (uint8 * DTCStatusMask)		
Service ID (hex)	Sync/Async	Reentrancy
0x16	Synchronous	Non Reentrant
Parameter		

DTCStatusMask	The value DTCStatu/TT*Q-as/TT the(/T supp(/Torted e Dm[_s)520(TCSt bi)520(s)3(from the
---------------	---

Return code	
Dem_ReturnGetNumberOfFilteredDTCType	Status of the operation to retrieve a number of DTC from the DEM
Functional Description	
Gets the number of a filtered DTC	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-23 Dem_GetNumberOfFilteredDTC

6.2.5.1.7 Dem_GetNextFilteredDTC

Prototype		
Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTC (uint32 * DTC, uint8 * DTCStatus)		
Service ID (hex)	Sync/Async	Reentrancy
0x18	Synchronous	Non Reentrant
Parameter		
DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
DTCStatus	This parameter receives the status information of the requested DTC. If the return value of the function call is other than DEM_FILTERED_OK this parameter does not contain valid data.	
Return code		
Dem_ReturnGetNext-FilteredDTCType	Status of the operation to retrieve a DTC from the DEM.	
Functional Description		
Gets the next filtered DTC.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-24 Dem_GetNextFilteredDTC

6.2.5.1.8 Dem_GetDTCByOccurrenceTime

Prototype		
<pre>Dem_ReturnGetDTCByOccurrenceTimeType Dem_GetDTCByOccurrenceTime (uint8 DTCRequest, uint8 DTCKind, uint32 * DTC)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x19	Synchronous	Non Reentrant
Parameter		
DTCRequest	<p>This parameter defines the request type of the DTC.</p> <p>DEM_FIRST_DET_CONFIRMED_DTC 0x03 first detected confirmed DTC requested,</p> <p>DEM_MOST_RECENT_FAILED_DTC 0x02 most recent failed DTC requested,</p> <p>DEM_MOST_REC_DET_CONFIRMED_DTC 0x04 most recently detected confirmed DTC requested,</p> <p>DEM_FIRST_FAILED_DTC 0x01 first failed DTC requested</p>	
DTCKind	<p>This parameter defines the requested DTC, either only OBD relevant DTCs or all DTCs</p> <p>DEM_DTC_KIND_ALL_DTCS Select all DTCs</p> <p>DEM_DTC_KIND_EMISSION_REL_DTC S Select OBD-relevant DTCs</p>	
DTC	<p>Receives the DTC value returned by the function. If the return value of the function is other than DEM_OCCURR_OK this parameter does not contain valid data.</p>	
Return code		
Dem_ReturnGetDTCBy-OccurrenceTimeType	Status of the operation of type Dem_ReturnGetDTCByOccurrenceTimeType.	
Functional Description		
Gets the DTC by occurrence time.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-25 Dem_GetDTCByOccurrenceTime

6.2.5.1.9 Dem_GetViewIDofDTC

Prototype		
Dem_ReturnGetViewIDofDTCType Dem_GetViewIDofDTC (uint32 DTC, uint8 DTCKind, uint8 * ViewId)		
Service ID (hex)	Sync/Async	Reentrancy
0x36	Synchronous	Non Reentrant
Parameter		
DTC	This parameter defines the requested DTC.	
DTCKind	This parameter defines the requested DTC, either only OBDrelevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
ViewId	The ViewId is a parameter used to select a specific view.	
Return code		
Dem_ReturnGetViewIDofDTCType	Status of the operation of type Dem_ReturnGetViewIDofDTCType.	
Functional Description		
Gets the ViewID of a DTC.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-26 Dem_GetViewIDofDTC

6.2.5.1.10 Dem_GetNextFilteredRecord

Prototype		
<pre>Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredRecord (uint32 * DTC, uint8 * SnapshotRecord)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x3a	Synchronous	Non Reentrant
Parameter		
DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
SnapshotRecord	Snapshot Record Number for the reported DTC.	

Return code	
Dem_ReturnGetNextFilteredDTCType	Status of the operation to retrieve a DTC from the DEM.
Functional Description	
Gets the current DTC and its associated snapshot record numbers from the DEM.	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-27 Dem_GetNextFilteredRecord

6.2.5.1.11 Dem_GetNextFilteredDTCAndFDC

Prototype		
<pre>Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTCAndFDC (uint32 * DTC, sint8 * DTCFaultDetectionCounter)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x3b	Synchronous	Non Reentrant
Parameter		
DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
DTCFaultDetection-Counter	This parameter receives the Fault Detection Counter information of the requested DTC. If the return value of the function call is other than DEM_FILTERED_OK this parameter does not contain valid data. -128dec...127dec PASSED...FAILED according to ISO 14229-1	
Return code		
Dem_ReturnGetNext-FilteredDTCType	Status of the operation to retrieve a DTC from the DEM.	
Functional Description		
Gets the current DTC and its associated Fault Detection Counter (FDC) from the DEM.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-28 Dem_GetNextFilteredDTCAndFDC

6.2.5.1.12 Dem_GetNextFilteredDTCAndSeverity

Prototype		
<pre>Dem_ReturnGetNextFilteredDTCType Dem_GetNextFilteredDTCAndSeverity(uint32 * DTC, uint8 * DTCStatus, uint8 * DTCSeverity, uint8 * DTCFunctionalUnit)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x3d	Synchronous	Non Reentrant
Parameter		
DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
DTCStatus	Receives the status value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
DTCSeverity	Receives the severity value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
DTCFunctionalUnit	Receives the functional unit value returned by the function. If the return value of the function is other than DEM_FILTERED_OK this parameter does not contain valid data.	
Return code		
Dem_ReturnGetNext-FilteredDTCType	Status of the operation to retrieve a DTC from the DEM.	
Functional Description		
Gets the current DTC and its Severity from the DEM.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-29 Dem_GetNextFilteredDTCAndSeverity

6.2.5.1.13 Dem_GetTranslationType

Prototype		
<pre>uint8 Dem_GetTranslationType ()</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x3c	Synchronous	Non Reentrant
Parameter		
None		

Return code	
uint8	<p>The TranslationFormat provides the configured translation formats</p> <p>Bit 0: 2 byte ISO15031-6 DTC</p> <p>Bit 1: 3 byte ISO14229-1 DTC</p> <p>Bit 2: Customer specific DTC</p> <p>Bit 3: SAEJ1939</p> <p>Bit 4: WWH-OBd-format</p> <p>Set the according Bit to '1' means DTC format is supported. Combination of different DTC formats is possible (e.g. ISO 15031-6 and ISO14229-1 is coded by 0x0011b).</p>
Functional Description	
Gets the supported DTC formats of the ECU. The supported formats are configured via DemTypeOfDTCSupported.	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-30 Dem_GetTranslationType

6.2.5.1.14 Dem_GetSeverityOfDTC

Prototype		
Dem_ReturnGetSeverityOfDTCType Dem_GetSeverityOfDTC (uint32 DTC, uint8 * DTCSeverity)		
Service ID (hex)	Sync/Async	Reentrancy
0x0e	Synchronous	Non Reentrant
Parameter		
DTC	The Severity assigned to this DTC should be returned	
DTCSeverity	This parameter contains the DTCSeverityMask according to ISO14229-1. DEM_SEVERITY_CHECK_IMMEDIATELY 0x80 Check immediately, DEM_SEVERITY_NO_SEVERITY 0x00 No severity information available, DEM_SEVERITY_CHECK_AT_NEXT_HALT 0x40 check at next halt, DEM_SEVERITY_MAINTENANCE_ONLY 0x20 maintenance required	
Return code		
Dem_ReturnGetSeverity-OfDTCType	Status of the operation of type Dem_ReturnGetSeverityOfDTCType.	
Functional Description		
Status of the operation of type Dem_ReturnGetSeverityOfDTCType.		

Particularities and Limitations
> None
Expected Caller Context
> Task context

Table 6-31 Dem_GetSeverityOfDTC

6.2.5.2 Access Extended Data Records and FreezeFrame Data

This section defines the API-calls to be used to get access to the environmental data stored with the DTCs in the records of the DEM.

6.2.5.2.1 Dem_DisableDTCRecordUpdate

Prototype		
Std_ReturnType Dem_DisableDTCRecordUpdate (
)		
Service ID (hex)	Sync/Async	Reentrancy
0x1a	Synchronous	Non Reentrant
Parameter		
None		
Return code		
Std_ReturnType	E_OK: Operation was successful	
	E_NOT_OK: Operation failed	
Functional Description		
Disables the DTC record update.		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-32 Dem_DisableDTCRecordUpdate

6.2.5.2.2 Dem_EnableDTCRecordUpdate

Prototype		
Std_ReturnType Dem_EnableDTCRecordUpdate(
)		
Service ID (hex)	Sync/Async	Reentrancy
0x1b	Synchronous	Non Reentrant
Parameter		
None		

Return code	
Std_ReturnType	E_OK: Operation was successful
	E_NOT_OK: Operation failed
Functional Description	
Enables the DTC record update	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-33 Dem_EnableDTCRecordUpdate

6.2.5.2.3 Dem_GetDTCOfFreezeFrameRecord

Prototype		
<pre>Dem_ReturnGetDTCOfFreezeFrameRecordType Dem_GetDTCOfFreezeFrameRecord (uint8 RecordNumber, uint8 DTCOrigin, uint8 DTCKind, uint32 * DTC)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x1c	Synchronous	Non Reentrant
Parameter		
RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.	
DTCOrigin	<p>If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.</p> <p>DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory,</p> <p>DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory,</p> <p>DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory</p> <p>The definition and use of the different memory types is OEM specific.</p>	
DTCKind	<p>This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs</p> <p>DEM_DTC_KIND_ALL_DTCS Select all DTCs</p> <p>DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs</p>	
DTC	Receives the DTC value returned by the function. If the return value of the function is other than DEM_GET_DTCOFFF_OK this parameter does not contain valid data.	

Return code	
Dem_ReturnGetDTCOfFreezeFrameRecordType	Status of the operation of type Dem_ReturnGetDTCOfFreezeFrameRecordType.
Functional Description	
Gets a DTC associated with a FreezeFrame.	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-34 Dem_GetDTCOfFreezeFrameRecord

6.2.5.2.4 Dem_GetFreezeFrameDataByDTC

Prototype		
<pre>Dem_ReturnGetFreezeFrameDataByDTCType Dem_GetFreezeFrameDataByDTC (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin, uint8 RecordNumber, uint16 DataId, uint8 * DestBuffer, uint8 * BufSize)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x1d	Synchronous	Non Reentrant
Parameter		
DTC	This is the DTC the FreezeFrame is assigned to.	
DTCKind	This parameter defines the requested DTC, either only OBD relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.	
RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.	

DataId	This parameter specifies the PID (ISO15031-5) (Mode2 individual parameter or the whole FreezeFrame data set) or data identifier (ISO14229-1) that shall be copied to the destination buffer.
DestBuffer	This parameter contains a byte pointer that points to the buffer to which the FreezeFrame data shall be written.
BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Return code	
Dem_ReturnGetFreezeFrameDataByDTCType	Status of the operation of type Dem_ReturnGetFreezeFrameDataByDTCType.
Functional Description	
Gets a Freeze Frame Data by DTC	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-35 Dem_GetFreezeFrameDataByDTC

6.2.5.2.5 Dem_GetFreezeFrameDataIdentifierByDTC

Prototype		
Dem_ReturnGetFreezeFrameDataIdentifierByDTCType Dem_GetFreezeFrameDataIdentifierByDTC (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin, uint8 RecordNumber, uint8 * ArraySize, const uint16 ** DataId) 		
Service ID (hex)	Sync/Async	Reentrancy
0x1e	Synchronous	Non Reentrant
Parameter		
DTC	This is the DTC the FreezeFrame is assigned to.	
DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	

DTCOrigin	<p>If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.</p> <p>DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory,</p> <p>DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory,</p> <p>DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory</p> <p>The definition and use of the different memory types is OEM specific.</p>
RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.
ArraySize	This parameter specifies the number of data identifiers for the selected RecordNumber.
DataId	Pointer to an array with the supported data identifier for the selected RecordNumber and DTC.
Return code	
Dem_ReturnGetFreezeFrameDataIdentifierByDTCType	Status of the operation of type Dem_ReturnGetFreezeFrameDataIdentifierByDTCType.
Functional Description	
Gets a Freeze Frame Data identifier by DTC	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-36 Dem_GetFreezeFrameDataIdentifierByDTC

6.2.5.2.6 Dem_GetSizeOfFreezeFrame

Prototype		
<pre>Dem_ReturnGetSizeOfFreezeFrameType Dem_GetSizeOfFreezeFrame (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin, uint8 RecordNumber, uint16 * SizeOfFreezeFrame)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
0x1f	Synchronous	Non Reentrant
Parameter		
DTC	This is the DTC the FreezeFrame is assigned to.	

DTCKind	<p>This parameter defines the requested DTC, either only OBDrelevant DTCs or all DTCs</p> <p>DEM_DTC_KIND_ALL_DTCS Select all DTCs</p> <p>DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs</p>
DTCOrigin	<p>If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.</p> <p>DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory,</p> <p>DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory,</p> <p>DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory</p> <p>The definition and use of the different memory types is OEM specific.</p>
RecordNumber	This parameter is a unique identifier for a FreezeFrame record as defined in ISO15031-5 and ISO14229-1.
SizeOfFreezeFrame	Number of bytes in the requested FreezeFrame.
Return code	
Dem_ReturnGetSizeOfFreezeFrameType	Status of the operation of type Dem_ReturnGetSizeOfFreezeFrameType
Functional Description	
Gets the size of a FreezeFrame	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-37 Dem_GetSizeOfFreezeFrame

6.2.5.2.7 Dem_GetExtendedDataRecordByDTC

Prototype		
Dem_ReturnGetExtendedDataRecordByDTCType Dem_GetExtendedDataRecordByDTC (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin, uint8 ExtendedDataNumber, uint8 * DestBuffer, uint8 * BufSize) 		
Service ID (hex)	Sync/Async	Reentrancy
0x20	Synchronous	Non Reentrant
Parameter		
DTC	This is the DTC the 'Extended Data Record' is assigned to.	

DTCKind	<p>This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs</p> <p>DEM_DTC_KIND_ALL_DTCS Select all DTCs</p> <p>DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs</p>
DTCOrigin	<p>If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.</p> <p>DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory,</p> <p>DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory,</p> <p>DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory</p> <p>The definition and use of the different memory types is OEM specific.</p>
ExtendedDataNumber	Identification of requested Extended data record. The requested record is copied to the destination buffer.
DestBuffer	This parameter contains a byte pointer that points to the buffer to which the Extended Data shall be written.
BufSize	When the function is called this parameter contains the maximum number of data bytes that can be written to the buffer. The function returns the actual number of written data bytes in this parameter.
Return code	
Dem_ReturnGetExtendedDataRecordByDTCType	Status of the operation of type Dem_ReturnGetExtendedDataRecordByDTCType
Functional Description	
Gets extended data record by DTC	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-38 Dem_GetExtendedDataRecordByDTC

6.2.5.2.8 Dem_GetSizeOfExtendedDataRecordByDTC

Prototype		
Dem_ReturnGetSizeOfExtendedDataRecordByDTCType Dem_GetSizeOfExtendedDataRecordByDTC (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin, uint8 ExtendedDataNumber, uint16 * SizeOfExtendedDataRecord)		
Service ID (hex)	Sync/Async	Reentrancy
0x21	Synchronous	Non Reentrant

Parameter	
DTC	This is the DTC the 'Extended Data Record' is assigned to.
DTCKind	<p>This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs</p> <p>DEM_DTC_KIND_ALL_DTCS Select all DTCs</p> <p>DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs</p>
DTCOrigin	<p>If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from.</p> <p>DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory,</p> <p>DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information located in the secondary memory,</p> <p>DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory</p> <p>The definition and use of the different memory types is OEM specific.</p>
ExtendedDataNumber	Identification of requested Extended data record. The requested record is copied to the destination buffer.
SizeOfExtendedData-Record	Pointer to Size of the requested data record
Return code	
Dem_ReturnGetSizeOf-ExtendedDataRecordBy-DTCType	Status of the operation of type Dem_ReturnGetSizeOfExtendedDataRecordByDTCType
Functional Description	
Gets the size of an extended data record by DTC	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context	

Table 6-39 Dem_GetSizeOfExtendedDataRecordByDTC

6.2.5.3 Clear DTC Information

The next sections define the usage of the API-calls to delete single DTCs as well as groups of DTCs from the records of the DEM.

6.2.5.3.1 Dem_ClearDTC

Prototype
<pre>Dem_ReturnClearDTCType Dem_ClearDTC (uint32 DTC, uint8 DTCKind, uint8 DTCOrigin)</pre>

Service ID (hex)	Sync/Async	Reentrancy
0x22	Synchronous	Non Reentrant
Parameter		
DTC	Defines the DTC that shall be cleared from the event memory. If the DTC fits to a DTC group number, all DTCs of the group shall be cleared.	
DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
DTCOrigin	If the DEM supports more than one event memory this parameter is used to select the source memory the DTCs shall be read from. DEM_DTC_ORIGIN_MIRROR_MEMORY Event information located in the mirror memory, DEM_DTC_ORIGIN_SECONDARY_MEMORY Event information ocated in the secondary memory, DEM_DTC_ORIGIN_PRIMARY_MEMORY Event information located in the primary memory The definition and use of the different memory types is OEM specific.	
Return code		
Dem_ReturnClearDTC-Type	Status of the operation of type Dem_ReturnClearDTCType.	
Functional Description		
Clears a DTC		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-40 Dem_ClearDTC

When the API is called with the wrong DTCKind, the return value is DEM_CLEAR_WRONG_DTC and not DEM_CLEAR_WRONG_DTCKIND, as DTC and DTCKind don't fit together, and it cannot be determined which of both is incorrect. However, if an invalid DTCKind is used, DEM_CLEAR_WRONG_DTCKIND will be returned.

6.2.5.4 Control DTC Storage

This section defines the API-calls to enable and disable DTC storage in the DEM.

6.2.5.4.1 Dem_DisableDTCStorage

Prototype		
Dem_ReturnControlDTCStorageType Dem_DisabledDTCStorage (uint32 DTCGroup, uint8 DTCKind)		
Service ID (hex)	Sync/Async	Reentrancy
0x24	Synchronous	Non Reentrant
Parameter		
DTCGroup	Defines the group of DTC that shall be disabled to store in event memory. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs	
DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
Return code		
Dem_ReturnControlDTC-StorageType	Status of the operation of type Dem_ReturnControlDTCStorageType.	
Functional Description		
Disables the storage of a DTC group		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-41 Dem_DisableDTCStorage



Info

Internal events can be disabled with the DTCGroup "all".

6.2.5.4.2 Dem_EnableDTCStorage

Prototype		
Dem_ReturnControlDTCStorageType Dem_EnabledDTCStorage (uint32 DTCGroup, uint8 DTCKind)		
Service ID (hex)	Sync/Async	Reentrancy
0x25	Synchronous	Non Reentrant
Parameter		
DTCGroup	Defines the group of DTC that shall be disabled to store in event memory. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs	
DTCKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
Return code		
Dem_ReturnControlDTC-StorageType	Status of the operation of type Dem_ReturnControlDTCStorageType.	
Functional Description		
Enables the storage of a DTC group		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-42 Dem_EnableDTCStorage

6.2.5.4.3 Dem_DisableEventStatusUpdate

Prototype		
Dem_ReturnControlEventUpdateType Dem_DisableEventStatusUpdate (uint32 DTCTGroup, uint8 DTCTKind)		
Service ID (hex)	Sync/Async	Reentrancy
0x26	Synchronous	Non Reentrant
Parameter		
DTCTGroup	Defines the group of DTC that shall be disabled to store in event memory. DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs, DEM_DTC_GROUP_EMISSION_REL_DTCS selects group of OBD-relevant DTCs, DEM_DTC_GROUP_ALL_DTCS selects all DTCs, DEM_DTC_GROUP_CHASSIS_DTCS selects group of chassis DTCs, DEM_DTC_GROUP_NETWORK_COM_DTCS selects group of network communication DTCs, DEM_DTC_GROUP_POWERTRAIN_DTCS selects group of powertrain DTCs	
DTCTKind	This parameter defines the requested DTC, either only OBD-relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS Select OBD-relevant DTCs	
Return code		
Dem_ReturnControlEvent-UpdateType	Status of the operation of type Dem_ReturnControlEventUpdateType	
Functional Description		
Disables the event status update of a DTC group		
Particularities and Limitations		
> None		
Expected Caller Context		
> Task context		

Table 6-43 Dem_DisableEventStatusUpdate

6.2.5.4.4 Dem_EnableEventStatusUpdate

Prototype		
<pre>Dem_ReturnControlEventUpdateType Dem_EnableEventStatusUpdate (uint32 DTCTGroup, uint8 DTCTKind)</pre>		

Service ID (hex)	Sync/Async	Reentrancy
0x27	Synchronous	Non Reentrant
Parameter		

DTCGroup Defines the group of DTC that shall be disabled to store in event memory.
DEM_DTC_GROUP_BODY_DTCS selects group of body DTCs,

6.3 Services Used by DEM

In the following table services provided by other components, which are used by the DEM are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError
EcuM	EcuM_GeneratorCompatibilityError ³
NvM	NvM_SetRamBlockStatus
SchM	SchM_Enter_Dem
SchM	SchM_Exit_Dem

Table 6-45 Services Used by the DEM

6.4 Callback Functions

Callback functions are functions that are implemented by the DEM and that can be invoked by other modules. The prototypes of the callback functions are provided in the header file `Dem.h` by the DEM.

6.4.1 Dem_NvDataInit

Prototype		
<pre>Std_ReturnType Dem_NvDataInit (void)</pre>		
Service ID (hex)	Sync/Async	Reentrancy
none	Synchronous	Reentrant
Parameter		
-		
Return code		
Std_ReturnType	E_OK	to fulfill requirement NVM369 for InitBlockCallbackFunction
Functional Description		
<p>This callback shall be configured in the NvM as callback (NvmBlockDescriptor/NvmInitBlockCallback) to initialize the buffer <code>Dem_NvData</code> (NvmBlockDescriptor/NvmRamBlockDataAddress). The NvM will trigger this callback when either the stored non-volatile data block was never initialized before or when the non-volatile data was corrupted.</p> <p>It may be called directly to re-initialize the non-volatile data of the Dem during the startup phase after the NvM has restored the previous data.</p> <p>The initialization will clear all DTCs and data records from Dem and set all events to status untested (0x50).</p>		

³ See [9]

Particularities and Limitations

- > GENy will automatically configure the NvM in the EcuC file to use that callback:
`/AUTOSAR/NvM/NvmBlockDescriptor/NvmRamBlockDataAddress` is set to `Dem_NvData`
`/AUTOSAR/NvM/NvmBlockDescriptor/NvmInitBlockCallback` is set to `Dem_NvDataInit`
`/AUTOSAR/NvM/NvmBlockDescriptor/NvmSelectBlockForReadall` is set to `true`
- > This function will reset all stored (non volatile) data of the Dem (besides the 'SuppressDTC' data).
- > Do not call this function after Dem was started, as this will corrupt the working data.
Only call this function after `Dem_PreInit()` and before calling `Dem_Init()`.
- > When using this function additional to the NvM callback to reset the (non volatile) data of the Dem, take care that the NvM may restore the data of the Dem inside the startup-sequence, too. To make an impact, in this case you shall call this function after the NvM's restoring and before `Dem_Init()`, otherwise the last wins (and Dem will use NvM's restored data).
- > In `Dem_Init()` the `Dem_NvData` buffer is tested for a specific pattern at the begin and at the end of the buffer. When either the pattern is missing or corrupt, the Dem itself will do a (re-)initialization of the buffer by calling this function.
- > This callback must be called before calling `Dem_Init()` each time when the non-volatile stored data is either uninitialized or corrupt.

Expected Caller Context

- > Must be called in startup phase only (task or interrupt context)

Table 6-46 Dem_NvDataInit

6.5 Configurable Interfaces

6.5.1 Notifications

At its configurable interfaces the DEM defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the DEM but can be performed at configuration time. The function prototypes that can be used for the configuration have to match the appropriate function prototype signatures, which are described in the following sub-chapters.



Info

If following interfaces are used as port interfaces without RTE, the function prefix `Rte_Call_Dem_` will be replaced by the prefix `App1_Dem_`.

6.5.1.1 Rte_Call_Dem_<ConfiguredName>_InitMonitorForEvent

Prototype	
<pre>Std_ReturnType Rte_Call_Dem_<ConfiguredName>_InitMonitorForEvent (Dem_InitMonitorKindType InitMonitorKind)</pre>	
Parameter	
InitMonitorKind	{Clear, Restart} Identification of the type of init function to be called from Monitor Function. DEM_INIT_MONITOR_RESTART (0x02) Monitor Function of the EventId is requested to restart, DEM_INIT_MONITOR_CLEAR (0x01) Monitor Function of the EventId is cleared and all internal values and states are reset.
Return code	
Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Functional Description	
(Re)initializes the Monitor Function of a specific Event.	
Particularities and Limitations	
<ul style="list-style-type: none"> > AUTOSAR requires CBInitEvt_<EventName> as <ConfiguredName>. In the configuration tool GENy the <ConfiguredName> is fully configurable by the user, you are not restricted to this AUTOSAR requirement. For full compliance with non-Vector components, choose the AUTOSAR required name. See “OnScreen Help” window in GENy for hint. 	
Expected Caller Context	
<ul style="list-style-type: none"> > Task context > The specific Event has been cleared or the operation cycle of this Event has been (re)started 	

Table 6-47 Rte_Call_Dem_<ConfiguredName>_InitMonitorForEvent

6.5.1.2 Rte_Call_Dem_<ConfiguredName>_EventStatusChanged

Prototype

```
Std_ReturnType Rte_Call_Dem_<ConfiguredName>_EventStatusChanged (
    Dem_EventStatusExtendedType EventStatusOld,
    Dem_EventStatusExtendedType EventStatusNew
)
```

Parameter

EventStatusOld	<p>Event status before change</p> <p>Bit 0 TestFailed is set to 1 if the last event status update by the function Dem_SetEventStatus(Passed Failed) was called with 'failed'. The status is set to 0 if Dem_SetEventStatus is called with 'passed', on tester clear command and by API Dem_ResetEventStatus.</p> <p>Bits 0 and 6 are intended to set/reset monitor 'inhibit' or 'default'.</p> <p>Bit 1 TestFailedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed failed) is called with 'failed' this cycle. Intended to be used for defaults reset only at next key on.</p> <p>Bit 2 PendingDTC is set when associated DTC becomes available in Mode07 (currently corresponds to ISO pending bit).</p> <p>Intended to be used for the control of IUMPR counters.</p> <p>Bit 3 ConfirmedDTC is set when associated DTC becomes available in Mode03 (currently corresponds to ISO confirmed bit). Could be used to set e.g. service request message.</p> <p>Bit 4 TestNotCompletedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus (passed failed) is called after last ClearDTC.</p> <p>Bit 5 testFailedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus is called with 'failed' this cycle.</p> <p>Bit 6 TestNotCompletedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is used, the differentiation is obsolete).</p> <p>Bit 7 WarningIndicatorRequested reports the status of Bit 0 (TestFailed) when the support of this bit is configured (see 8.2.12.5.1).</p>
----------------	--

EventStatusNew	<p>Event status after change</p> <p>Bit 0 TestFailed is set to 1 if the last event status update by the function Dem_SetEventStatus(Passed Failed) was called with 'failed'. The status is set to 0 if Dem_SetEventStatus is called with 'passed', on tester clear command and by API Dem_ResetEventStatus.</p> <p>Bits 0 and 6 are intended to set/reset monitor 'inhibit' or 'default'.</p> <p>Bit 1 TestFailedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed failed) is called with 'failed' this cycle. Intended to be used for defaults reset only at next key on.</p> <p>Bit 2 PendingDTC is set when associated DTC becomes available in Mode07 (currently corresponds to ISO pending bit).</p> <p>Intended to be used for the control of IUMPR counters.</p> <p>Bit 3 ConfirmedDTC is set when associated DTC becomes available in Mode03 (currently corresponds to ISO confirmed bit). Could be used to set e.g. service request message.</p> <p>Bit 4 TestNotCompletedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus (passed failed) is called after last ClearDTC.</p> <p>Bit 5 testFailedSinceLastClear is set to 0 if at least one time the function Dem_SetEventStatus is called with 'failed' this cycle.</p> <p>Bit 6 TestNotCompletedThisOperationCycle is set if at least one time the function Dem_SetEventStatus (passed failed) is called within this cycle (the usage of different cycles is application-specific, if only one cycle is used, the differentiation is obsolete).</p> <p>Bit 7 WarningIndicatorRequested reports the status of Bit 0 (TestFailed) when the support of this bit is configured (see 8.2.12.5.1).</p>
Return code	
Std_ReturnType	<p>E_OK: Operation was successful</p> <p>E_NOT_OK: Operation failed</p>
Functional Description	
Triggers on changes of the extended Event status	
Particularities and Limitations	
> None	
Expected Caller Context	
> Task context > The extended Event status of the specific Event has been changed	

Table 6-48 Rte_Call_Dem_<ConfiguredName>_EventStatusChanged

6.5.1.3 Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged

Prototype	
<pre>Std_ReturnType Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged (Dem_DTCType DTC, Dem_DTCKindType DTCKind, Dem_DTCStatusMaskType DTCStatusOld, Dem_DTCStatusMaskType DTCStatusNew)</pre>	
Parameter	
DTC	This is the DTC the change trigger is assigned to.
DTCKind	This parameter defines the requested DTC, either only OBD relevant DTCs or all DTCs DEM_DTC_KIND_ALL_DTCS: Select all DTCs DEM_DTC_KIND_EMISSION_REL_DTCS: Select OBD-relevant DTCs
DTCStatusOld	DTC status before change
DTCStatusNew	DTC status after change
Return code	
Std_ReturnType	E_OK: Operation was successful E_NOT_OK: Operation failed
Functional Description	
Triggers on changes of the DTC status (the DTC status reports all supported status bits but the bits disabled by the DTCStatusAvailabilityMask, see 8.1.1).	
Particularities and Limitations	
<p>> AUTOSAR requires CBStatusDTC_<EventName> as <ConfiguredName>. In the configuration tool GENy the <ConfiguredName> is fully configurable by the user, you are not restricted to this AUTOSAR requirement. For full compliance with non-Vector components, choose the AUTOSAR required name. See "OnScreen Help" window in GENy for hint.</p>	
Expected Caller Context	
<p>> Task context</p> <p>> The status of the specific Event (DTC) has been changed</p>	

Table 6-49 Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged



Info

The parameter DTCKind is an extension to the AUTOSAR description. This parameter is needed to get a unique prototype (a DTC is not unambiguous) – see Bugzilla 24330.

6.5.2 Callout Functions

At its configurable interfaces the DEM defines callout functions. The declarations of the callout functions are provided by the BSW module. It is the integrator's task to provide the

corresponding function definitions. The definitions of the callouts can be adjusted to the system's needs.

**Info**

If following interfaces are used as port interfaces without RTE, the function prefix `Rte_Call_Dem_` will be replaced by the prefix `App1_Dem_`.

6.5.2.1 Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier

Prototype

```
Std_ReturnType
Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier (
    Dem_MaxDataValueType Data
)
```

Parameter

Data	Pointer to the buffer in the DEM which must be filled with the requested value.
------	---

Return code

Std_ReturnType	If the data value of the requested data identifier is available and successful supplied as out parameter the API returns E_OK. If there is no data value available for a certain data identifier the API returns E_NOT_OK. In case of E_NOT_OK the DEM fills the missing data with the padding value 0xFF, reports the development error DEM_E_NODATAAVAILABLE to the DET and continues its normal operation.
----------------	---

Functional Description

Gets data value defined by a data identifier.

Particularities and Limitations

- > AUTOSAR requires **CBValByDID_<DataID>** as <ConfiguredName>. In the configuration tool GENy the <ConfiguredName> is fully configurable by the user, you are not restricted to this AUTOSAR requirement. For full compliance with non-Vector components, choose the AUTOSAR required name. See "OnScreen Help" window in GENy for hint.

Expected Caller Context

- > Task context
- > Data values are required for a specific (pre-stored) FreezeFrame

Table 6-50 Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier

**Info**

To get DCM compatible ports, use the option "Use Dcm Compatible Ports" described in chapter 8.2.6.

**Caution**

When DCM compatible ports are used, the runnable may not return neither E_PENDING nor E_NOT_OK.

In case of an error the SWC shall return E_OK and fill the whole data buffer with 0xFF bytes instead of returning E_NOT_OK.

6.5.2.2 Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord

Prototype	
<pre>Std_ReturnType Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord (Dem_MaxExtendedDataRecordType Data)</pre>	
Parameter	
Data	Pointer to the buffer in the DEM which must be filled with the requested value.
Return code	
Std_ReturnType	The return value specifies whether the API call has been successful (Extended Data Record available) or not (no Extended Data Record available). In case of E_NOT_OK the DEM fills the missing Data with the padding value 0xFF, reports the development error DEM_E_NODATAAVAILABLE to the DET and continues its normal operation.
Functional Description	
Gets an extended data record.	
Particularities and Limitations	
<p>> AUTOSAR requires CBExtDataRec_<RecordNumber> as <ConfiguredName>. In the configuration tool GENy the <ConfiguredName> is used as required by AUTOSAR.</p>	
Expected Caller Context	
<p>> Task context</p> <p>> ExtendedDataRecord is required</p>	

Table 6-51 Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord

6.6 Service Ports

6.6.1 Client Server Interface

A client server interface is related to a Provide Port at the server side and a Require Port at the client side.

6.6.1.1 Provide Ports on DEM Side

At the Provide Ports of the DEM the API functions described in 6.2 are available as Runnable Entities. The Runnable Entities are invoked via operations. The mapping from a SWC client call to an operation is performed by the RTE. In this mapping the RTE adds Port Defined Argument Values to the client call of the SWC, if configured.

The following sub-chapters present the Provide Ports defined for the DEM and the operations defined for the Provide Ports, the API functions related to the operations and the Port Defined Argument Values to be added by the RTE.

6.6.1.1.1 DiagnosticMonitor

Operation	API Function	Port Defined Argument Values
SetEventStatus	Dem_SetEventStatus	IN Dem_EventStatusType EventStatus, ERR{E_NOT_OK}
ResetEventStatus	Dem_ResetEventStatus	ERR{E_NOT_OK}
GetEventStatus	Dem_GetEventStatus	OUT Dem_EventStatusExtendedType EventStatusExtended, ERR{E_NOT_OK}
GetEventFailed	Dem_GetEventFailed	OUT Boolean EventFailed, ERR{E_NOT_OK}
GetEventTested	Dem_GetEventTested	OUT Boolean EventTested, ERR{E_NOT_OK}
GetDTCOfEvent	Dem_GetDTCOfEvent	IN Dem_DTCKindType DTCKind, OUT Dem_DTCType DTC, ERR{E_NOT_OK, DEM_GET_DTCCOFEVENT_WRONG_DTCKIND, DEM_GET_DTCCOFEVENT_WRONG_EVENTID}
PrestoreFreezeFrame	Dem_PrestoreFreezeFrame	ERR{E_NOT_OK}
ClearPrestoredFreezeFrame	Dem_ClearPrestoredFreezeFrame	ERR{E_NOT_OK}
GetFaultDetectionCounter	Dem_GetFaultDetectionCounter	OUT Dem_FaultDetectionCounterType EventIdFaultDetectionCounter, ERR{E_NOT_OK}
GetOccurrenceCounter	Dem_GetOccurrenceCounter	OUT UInt8 OccurrenceCounterValue, ERR{E_NOT_OK}

Table 6-52 DiagnosticMonitor

6.6.1.1.2 OperationCycle

Operation	API Function	Port Defined Argument Values
SetOperationCycleState	Dem_SetOperationCycleState	IN Dem_OperationCycleStateType CycleState, ERR{E_NOT_OK}

Table 6-53 OperationCycle

6.6.1.1.3 ValueByOemId

This optional interface is not supported by Vector.

See chapter 6.2.3.10.

6.6.1.1.4 EnableCondition

This optional interface is not supported by Vector.

See chapter 6.2.3.11.

6.6.1.1.5 IndicatorStatus

Operation	API Function	Port Defined Argument Values
GetIndicatorStatus	Dem_GetIndicatorStatus	OUT IndicatorStatusType IndicatorStatus, ERR{E_NOT_OK}

Table 6-54 IndicatorStatus

6.6.1.2 Require Ports on DEM Side

At its Require Ports the DEM calls Operations. These Operations have to be provided by the SWCs by means of Runnable Entities. These Runnable Entities implement the callback functions expected by the DEM.

The following sub-chapters present the Require Ports defined for the DEM, the Operations that are called by the DEM and the related Notifications or Callouts, which are described in chapter 6.5.

6.6.1.2.1 Notification CallbackInitMonitorForEvent

Operation	Notification
InitMonitorForEvent{EventName}	Rte_Call_Dem_<ConfiguredName>_InitMonitorForEvent

Table 6-55 CallbackInitMonitorForEvent

AUTOSAR requires CBInitEvt_<EventName> as <ConfiguredName>. This name is suggested by the configuration tool GENy but fully configurable by the user.

6.6.1.2.2 Notification CallbackInitMonitorForFunction

This callback is not supported by Vector DEM. To initialize monitors use the AUTOSAR API CallbackInitMonitorForEvent instead.

6.6.1.2.3 Notification CallbackEventStatusChange

Operation	Notification
EventStatusChanged	Rte_Call_Dem_<ConfiguredName>_EventStatusChanged

Table 6-56 CallbackEventStatusChange

AUTOSAR requires CBStatusEvt_<EventName> as <ConfiguredName>. This name is suggested by the configuration tool GENy but fully configurable by the user.

6.6.1.2.4 Notification CallbackDTCStatusChange

Operation	Notification
DTCStatusChanged	Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged

Table 6-57 CallbackDTCStatusChange

AUTOSAR requires CBStatusDTC_<EventName> as <ConfiguredName>. This name is suggested by the configuration tool GENy but fully configurable by the user.

6.6.1.2.5 Callout Function CallbackGetDataValueByDataID{DataId}

Operation	Callout Function
GetDataValueByDataID	Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataID

Table 6-58 CallbackGetDataValueByDataID{DataId}

AUTOSAR requires CBValByDID_<DataID> as <ConfiguredName>. This name is suggested by the configuration tool GENy but fully configurable by the user.



Info

To get DCM compatible ports, use the option “Use Dcm Compatible Ports” described in chapter 8.2.6.

6.6.1.2.6 Callout Function CallbackGetExtendedDataRecord{RecordNumber}

Operation	Callout Function
GetExtendedDataRecord	Rte_Call_Dem_<ConfiguredName>_GetExtendedDataRecord

Table 6-59 CallbackGetExtendedDataRecord{RecordNumber}

AUTOSAR requires CBExtDataRec_<RecordNumber> as <ConfiguredName>. This name is used by the configuration tool GENy.

6.6.1.2.7 Callout Function CallbackGetFaultDetectCounter

As debouncing is implemented in the DEM, the SWC does not have to deliver the DTCFaultDetectionCounter via this callback function.

Therefore there is no need to support this API.

7 Additional Features

7.1 Multi-Identity Support

7.1.1 Functionality

Per specification the DEM has a static configuration set – once all events are configured, and the program code is flashed into the ECU there are no more configuration changes possible. This is the so called single identity mode.

The DEM module, developed by Vector, additionally implements a dynamic configuration mode that allows switching among the available configuration sets at run-time (without need to reprogram the ECU). This requires that the superset of all variants must be pre-enabled during the software compilation and link process.

DEM/DCM supports the following kinds of diagnostic configuration set usage:

- > Single Identity – the ECU is programmed with only one configuration set.
- > Multi Identity – the ECU is programmed with multiple configuration sets. At ECU start up there will be only one of them selected as active (exclusive configuration selection).

Multi Identity Mode is not available with this version of the DEM.

- > Vehicle System Group (VSG) – the ECU is programmed with multiple configuration sets, where as one of them is specified as base variant. At ECU start up the base variant will be always active (automatically used in DEM). Optionally the application can select one or more from the available configuration sets to be active too (additive configuration selection).

For more details about the different kinds of the configuration use cases, please refer to the following chapters and to the related chapters of the DCM Technical Reference.

7.1.2 Single-Identity Mode

If the single-identity operation mode has been selected, DEM uses a statically assigned configuration set which is linked at ECU compile and link time.

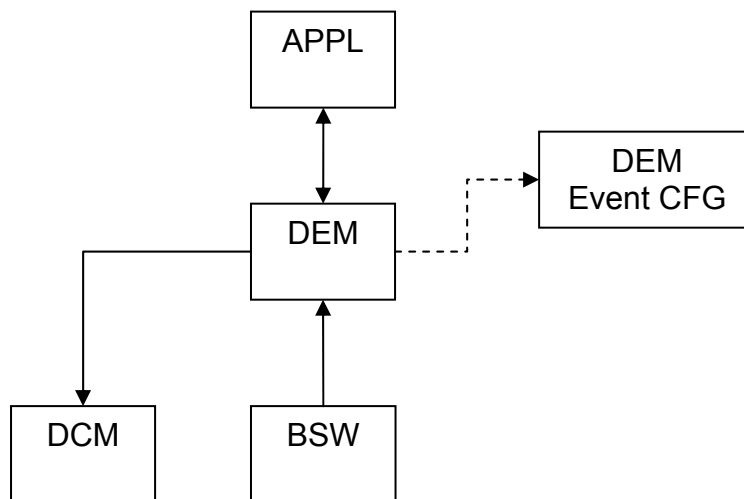


Figure 7-1 DEM single identity mode

In this case DEM has only one diagnostic configuration set.

7.1.2.1 Configuration in CANdela

You just need to prepare the corresponding CDD variant for your ECU configuration in CANdelaStudio as usual.

7.1.2.2 Configuration in GENy

Import the CDD file and the corresponding CDD variant in GENy (refer to chapter 8.2.2.1 Single Identity Mode).

7.1.3 Vehicle System Group (VSG) Mode

In Vehicle System Group (VSG) mode, DEM has the following characteristics:

- > Allows to support multiple diagnostic configuration variants – each variant reflects a VSG from the imported CDD file, and additionally there is a base variant that contains all services that does not belong to any VSG.
- > One or several configuration variants can be simultaneously activated during the ECU initialization. The base variant is always active.

All VSGs are specified by a single variant of a CDD (see chapter 7.1.3.1). The common part of all VSGs will build the *base variant*. The difference of each VSG in relation to the *base variant* will build a *diagnostic configuration variant*. At startup, the ECU will be configured with the *base variant* plus zero or more *diagnostic configuration variants*.

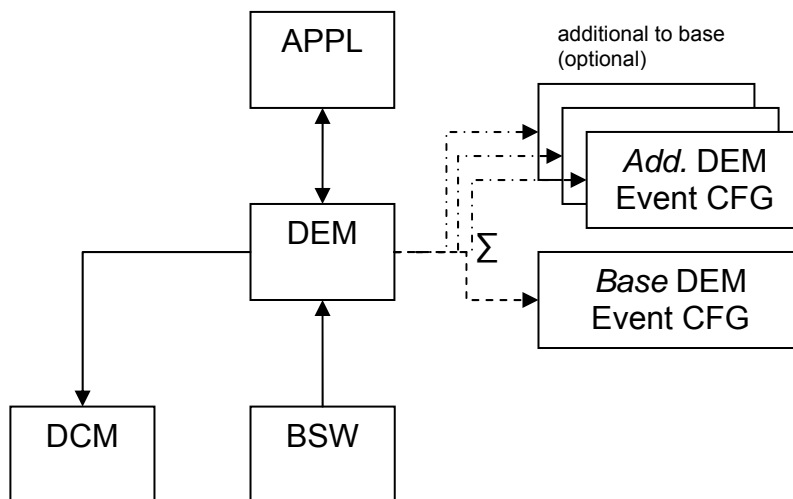


Figure 7-2 DEM Vehicle System Group (VSG) mode

7.1.3.1 Configuration in CANdela

Please follow the steps below on how to configure VSG in CANdelaStudio.

1. Defining all available VSGs for the actual ECU.

In CANdelaStudio, select the Vehicle System Groups view and add all necessary VSGs into the VSG pool.

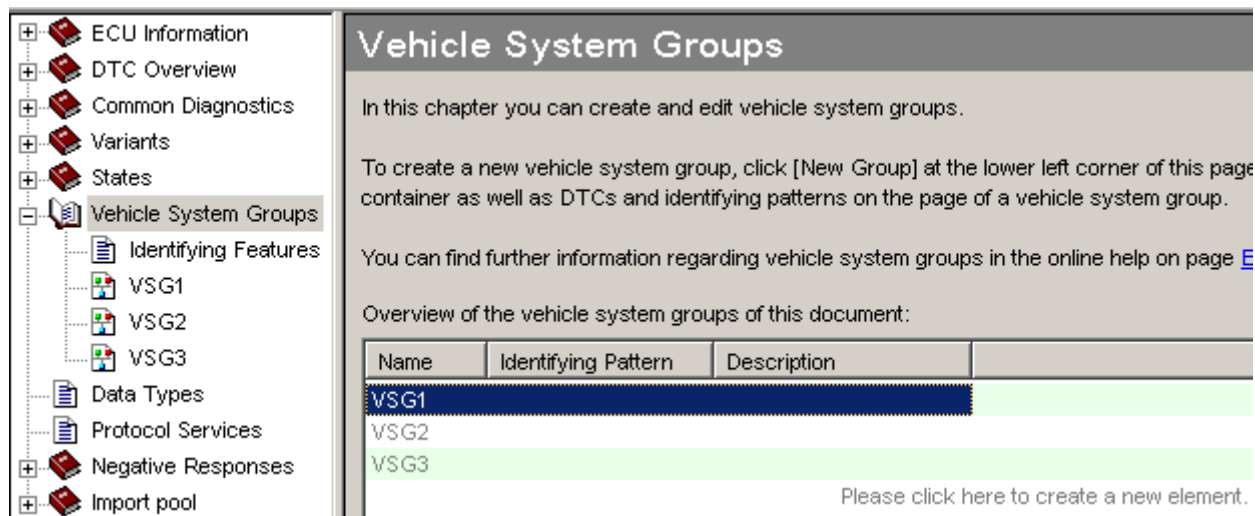


Figure 7-3 Defining VSGs in CANdelaStudio

The name of the created VSGs will be used later by DEM for the diagnostic configuration constants that the DEM application shall use during the configuration activation phase (see chapter 6.2.2.4 *Dem_InitDiagnosticVariant*).

Each DTC can be assigned to one or more VSGs.

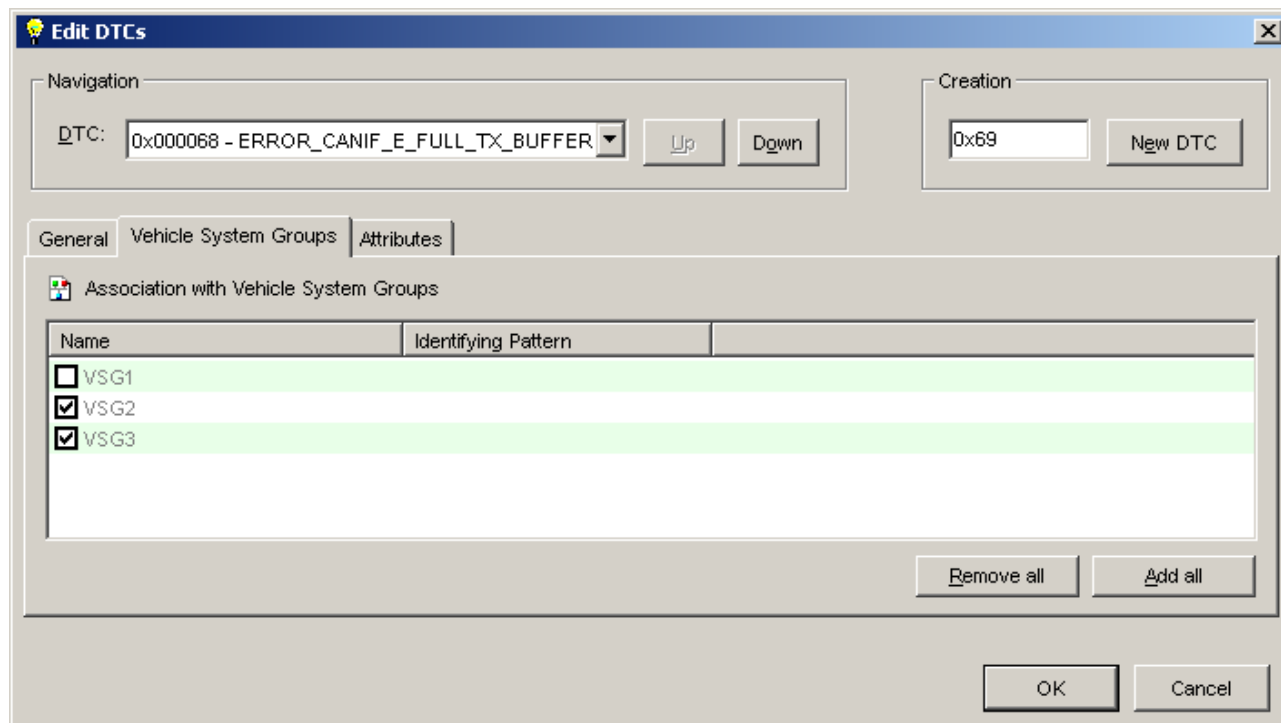


Figure 7-4 Assigning VSGs to a DTC in CANdelaStudio

If a DTC is part of all defined VSGs (or in none of them) it will be put into the *base variant*, otherwise it will be put into the configured additive *diagnostic configuration variant(s)*.

In the example above it will become part of additive variant “VSG2” and of “VSG3” – it is not part of the additive variant “VSG1”.

The DTC is available, when you start-up your ECU in these modes:

- > Base variant + VSG2
- > Base variant + VSG3
- > Base variant + VSG1 + VSG2
- > Base variant + VSG2 + VSG3

It is not available, when you start-up your ECU in that mode:

- > Base variant only
- > Base variant + VSG1

7.1.3.2 Configuration in GENy

Import the CDD file in GENy (refer to chapter 8.2.2.2 *Vehicle System Group (VSG) Mode*).

8 Configuration

GENy is the configuration tool for all MICROSAR components. CANdelaStudio is the expert tool for diagnostic communication managed by the DCM. All diagnostic services used by the DCM can be configured there. The DCM and the DEM work closely together so significant attributes of the DEM can be configured in CANdelaStudio as well as in GENy.

In the DEM the attributes can be configured with the following methods:

- > Configuration in CANdelaStudio, for a detailed description see 8.1
- > Configuration in GENy, for a detailed description see 8.1.3.1



Info

The configuration made in CANdelaStudio can be used for automatic configuration in GENy – see 8.2.2.3.



Caution

The structure of the template (CDDT) and the qualifier of all relevant attributes and parameters is predefined for correct import of CDD files in GENy. Any modification might result in unrecognized values during the import.

The DIAG_ASRDEM_VECTOR 2.08.00 is released for CDD-Template **Vector 2.0.1**. Please check that your CDD founds on the correct CDDT in CANdelaStudio menu “File” > “Properties...” > “Document Info”.

8.1 Configuration with CANdelaStudio

The following attributes can be configured in the diagnostic instance “Fault Memory” for the DEM.

8.1.1 DTC Settings

In the DTCStatusAvailabilityMask table you can configure the reported status bits which are used for all DTCs.

In the DTC Table you can create DTCs and make the needed settings for them. See the following table to find the analogy between the CANdela attributes and the GENy notations (the attributes not used for the configuration are not mentioned here).

DTC Table	
CANdelaStudio	GENy
DTC	DTC
Errortext shortname	Symbolic Name
Operation Cycle	Operation Cycle Ref
Self Healing (Aging)	Healing Allowed
Healing Cycle	Healing Cycle Ref
Healing Cycles	Healing Cycle Counter
Monitor Initialization Function	Init Monitor Name, will be: <code>CBInitEvt_<Symbolic Name></code>
DTC Priority	Event Priority
DTC Severity	DTC Severity
Pre-Debounce Algorithm	Pre Debounce Name
Count-In step size	Count In Step Size
Count-Out step size	Count Out Step Size
Use Jump-Up (Prefailed)	Jump Up
Use Jump-Down (Prepassed)	Jump Down
Set WarningIndicator Bit at fault	DTC Support Warning Indicator
Functional Unit	DTC Functional Unit – fixed to 0xFF
DTC Debounce Failed Timeout Value	Failed Timeout Value [ms]
DTC Debounce Passed Timeout Value	Passed Timeout Value [ms]

Table 8-1 Equivalence of DTC Settings in CANdelaStudio and GENy

**Info**

Several BSW errors are predefined for your project. These errors are typically internal and therefore, no DTC number is configured and no additional data will be stored when the event becomes failed. In GENy, many settings of BSW errors are greyed out and can not be changed.

To change settings of BSW errors, you must configure them via CANdelaStudio. Use the “Symbolic Name” of the BSW error as “Errortext shortname” and make your configurations. During the import of the CDD in GENy (see 8.2.2.3) the settings of that BSW error will be overwritten. To avoid inconsistencies, all imported settings will be write-protected (i.e. greyed out) in GENy.

**Info**

In Vehicle System Group (VSG) mode (see chapter 7.1.3) all DTCs created in GENy are put into the *base variant*.

8.1.2 Freeze Frame Types (Snapshot Records)

Snapshot records can be assigned to the single DTCs. They consist of one or more DIDs (Data Identifiers). To configure the snapshot record data, follow the subsequent description:

- > Ensure that the required DIDs are available in a data identifier diagnostic class e.g. “Read Only Data Identifiers”. Create a new diagnostic instance with the required identifier if necessary:
 - Right-Mouse-Button on “Read Only Data Identifiers”, add new diagnostic instance
 - Configure name and DID
 - Create a new Packet data type for this DID (configure name and size), copy or reuse an existing Packet data type
- > Assign the needed DIDs to the DTCs via Right-Mouse-Button menu “New DID reference” (tab “Snapshot Record” of diagnostic instance “Fault Memory”).

8.1.3 Extended Records

The configured extended data record package is used for all DTCs.

- > Ensure that the data types with the required extended records are available. Otherwise create a new data type “Text Table” or extend an existing one with the missing extended record numbers (e.g. data type “ExtendedDataRecordNumber”).
- > Assign the data type including the required extended data record numbers in tab “ExtendedDataRecordNumber”.

8.1.3.1 OccurrenceCounter

For using the internally implemented occurrence counter your CDD must base on a CDDT that offers this feature. See description below, how to check for the correct CDD.

The extended data record “OccurrenceCounter” shall be predefined in the CDD of your delivery. The record has some constraints:

- it must contain only one data element, and
- this data element must be of Datatype “Occurrence Counter”, which is a 1 Byte unsigned data that was specifically flagged in the CDDT.

The record number for the extended data record can be modified by the user. It defaults to Extended Record Number 0x01.

If this extended data record exists, the internal implementation for the Occurrence Counter will be activated, you will get an API for the BSW modules to read the counter (see

6.2.3.14) and you will get an additional operation for the PortInterface DiagnosticMonitor for access via the RTE (see 6.6.1.1.1).



Info

If this extended record does not exist or above constraints are not fulfilled, the internal implementation of the occurrence counter will be deactivated.

When the “OccurrenceCounter” is missing in your CDD (but supported by the CDDT), please follow the subsequent description to add the counter:

- > Ensure that the datatype “Occurrence Counter” exists. Create a new “raw value” if necessary:
 - Right-Mouse-Button in the list of the Data Types, add “New Data Type” “Raw Value”
 - In tab “General” configure name (“Occurrence Counter”) and qualifier (“OccurrenceCounter”)
 - In tab “id Raw value” configure Length (Byte:Bit) “1:0” and “Encoding” with “Unsigned”
- > Assign this data type to the extended record you want to use as occurrence counter (in diagnostic class “Fault Memory”, tab “ExtendedDataRecord”)

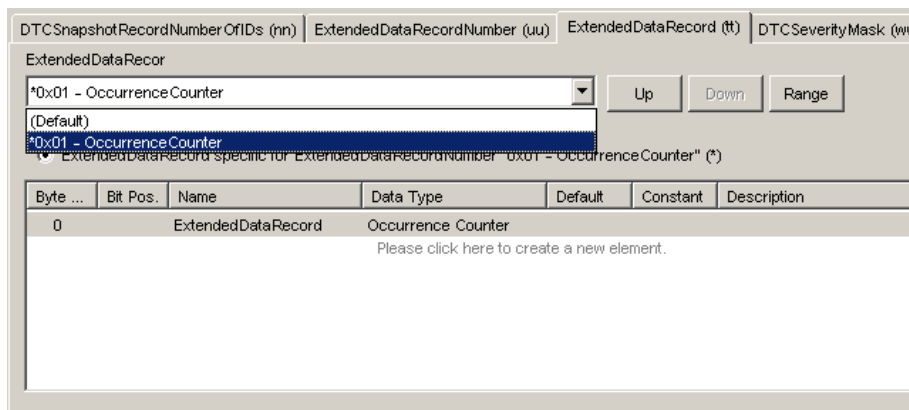


Figure 8-1 OccurrenceCounter

When you followed above sequence but the internal implementation of the occurrence counter is still not available, please prove following, to evaluate if the feature is missing in your CDDT.

In GENy open the DEM's extended data record view:



In the right side of the window, the internal OccurrenceCounter will look like:

Configurable Options	OccurrenceCounter
Remove this extended data record	...
Symbolic Name	OccurrenceCounter
Extended Data Record Number	0x1
Extended Data Record Data Size [bytes]	1

← Correct: No Function here

Figure 8-2 This CDD supports the internal Occurrence Counter

When the CDD/CDDT does **not support** the internal OccurrenceCounter, the window contains an additional row “Get Ext Data Record Fnc”. In this case you must change your CDDT or upgrade your CDD to use the internal counter.

Configurable Options	OccurrenceCounter
Remove this extended data record	...
Symbolic Name	OccurrenceCounter
Extended Data Record Number	0x1
Extended Data Record Data Size [bytes]	1
Get Ext Data Record Fnc	CBExtDataRec_0x01

← Wrong: External Function

Figure 8-3 This CDD does NOT support the internal Occurrence Counter

8.1.4 Pre de-bounce algorithms

The DEM offers the algorithms

- > Counter based
- > Time based

The CDD setting “Application based” is not supported. During import, the counter based algorithm is used instead and GENy will report an error message for the wrong configured DTC.

For implementing an external de-bouncing, choose the counter based algorithm with step size “1”. When you start the application based, external de-bouncing, send the event status PRE_FAILED respectively PRE_PASSED as trigger to the DEM to correctly initialize the FaultDetectionCounter. At end of application based de-bouncing, report the final event status to DEM as PASSED respectively FAILED.

8.2 Configuration with GENy

The DEM is configured with the help of the configuration tool GENy.

8.2.1 Start the Configuration

To configure the DEM select this component in the system configuration window. The DEM should be visible in the tree view.

8.2.2 Import CDD

Independent of the used variant mode (Single Identity or Vehicle System Group (VSG)), the used CDD(s) are configured with component DCM for both, the DEM and the DCM.



Info

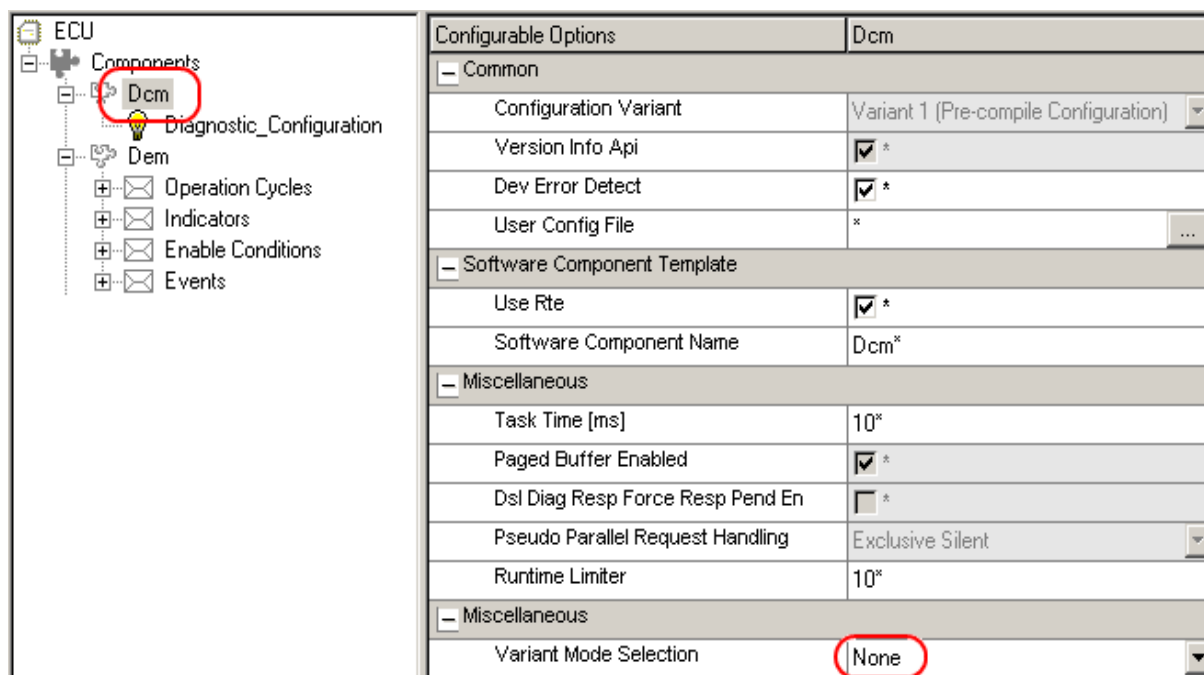
When you switch off the DCM, the virtual component “Diagnostic Configuration” will pop up instead. Use this component to configure the imported CDD files and diagnostic variants:



Note: For supporting Multiple Identity Mode (see chapter 7.1.3 *Vehicle System Group (VSG) Mode*) together with ECU-C file format to store the configuration, the use of Vector’s DCM is mandatory.

8.2.2.1 Single Identity Mode

First you have to put the DCM GENy configurator in single-identity mode:



Configurable Options	Dcm
Common	
Configuration Variant	Variant 1 (Pre-compile Configuration)
Version Info Api	<input checked="" type="checkbox"/> *
Dev Error Detect	<input checked="" type="checkbox"/> *
User Config File	* ...
Software Component Template	
Use Rte	<input checked="" type="checkbox"/> *
Software Component Name	Dcm*
Miscellaneous	
Task Time [ms]	10*
Paged Buffer Enabled	<input checked="" type="checkbox"/> *
Dsl Diag Resp Force Resp Pend En	<input type="checkbox"/> *
Pseudo Parallel Request Handling	Exclusive Silent
Runtime Limiter	10*
Miscellaneous	
Variant Mode Selection	None

Figure 8-4 Switching DCM/DEM in GENy to single-identity mode

Now you have to specify the CDD file and a CDD variant for the first diagnostic configuration.

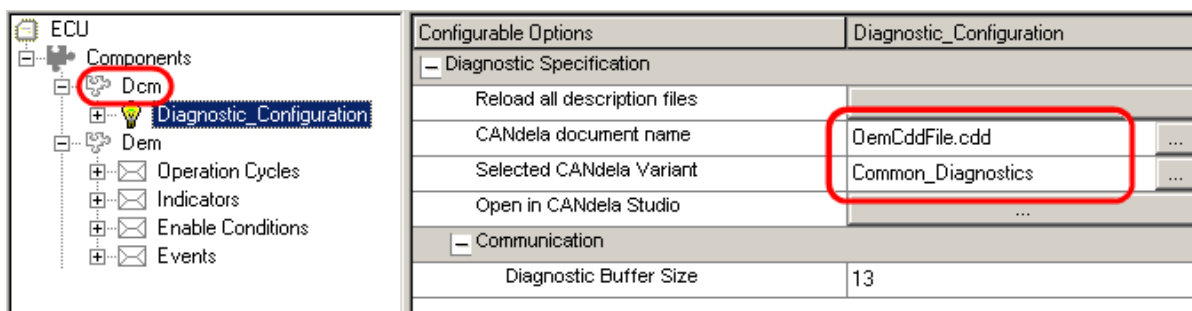


Figure 8-5 Select the CDD and the variant in the CDD

8.2.2.2 Vehicle System Group (VSG) Mode

Here are the steps you have to follow in order to configure multi-identity mode (Vehicle System Group – VSG) in GENy.

1. Select multi-identity mode “VSG Mode” for DCM and DEM

First you have to put the DCM GENy configurator in multi-identity mode:

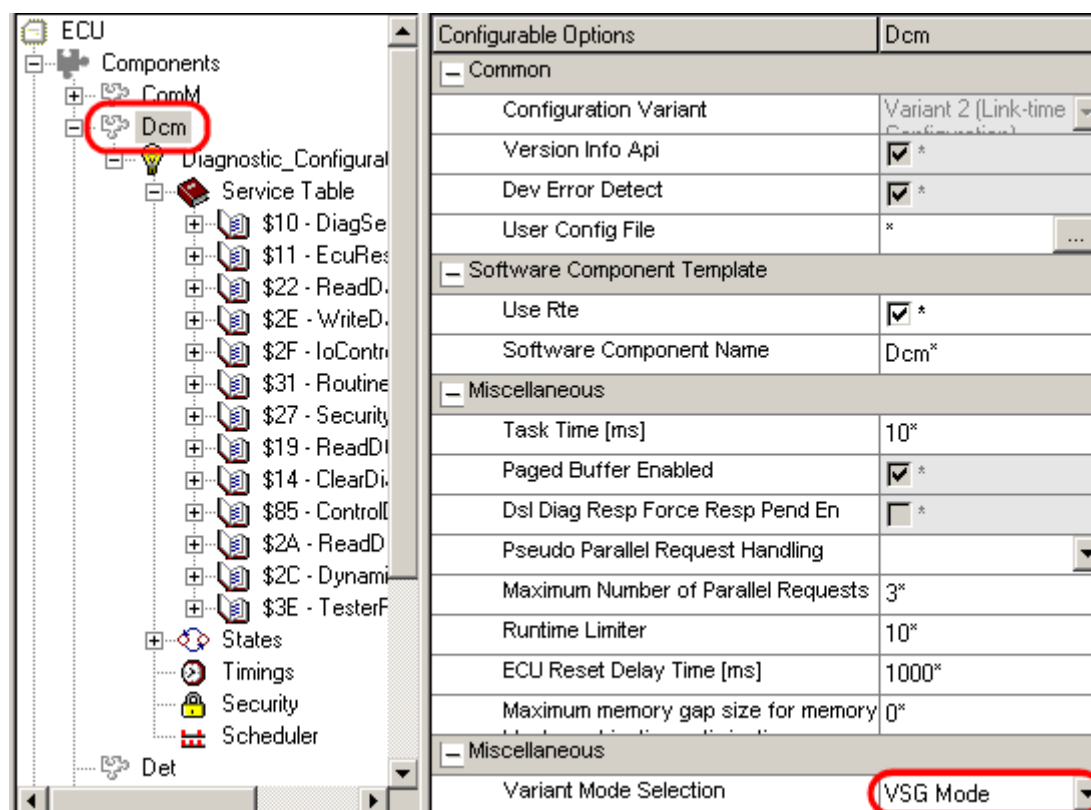


Figure 8-6 Switching DCM in GENy to VSG mode

Now import the CDD file and specify the CDD-variant containing the VSGs in GENy.

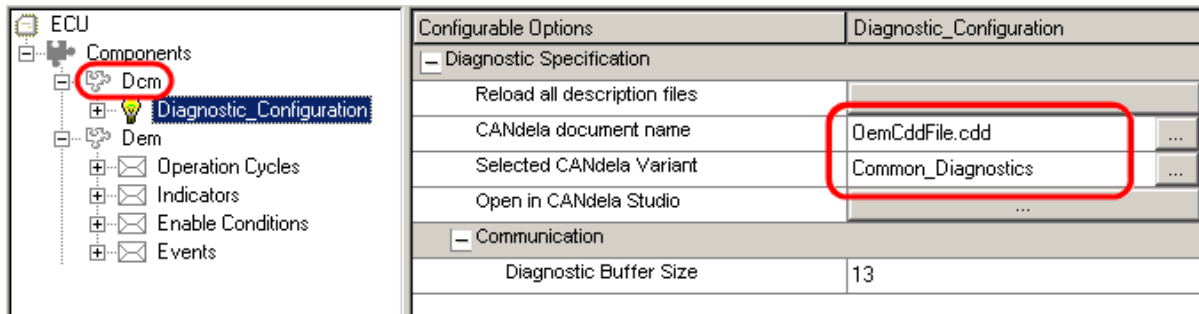


Figure 8-7 Select the CDD and the variant in the CDD

8.2.2.3 Automatic Configuration via CANdelaStudio Files

For automatic configuration via CANdelaStudio file (*.CDD) here some remarks.

8.2.3 Tree View

With the tree view the different tables can be accessed.

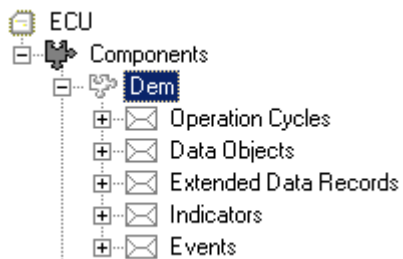


Figure 8-8 Tree View

By pressing the + you will open the tree and see the currently defined objects.

8.2.4 Main Configuration Window

Configurable Options	Dem
Common	
Configuration Variant	Variant 2 (Link-time Configuration)
Version Info Api	<input checked="" type="checkbox"/> *
Dev Error Detect	<input checked="" type="checkbox"/> *
User Config File	* ...
Software Component Template	
Use Rte	<input checked="" type="checkbox"/> *
Software Component Name	Dem*
Use Dcm Compatible Ports	<input type="checkbox"/> *
Miscellaneous	
Dem Version	2.08.00
Task Time [ms]	10*
Support Fim Notification API	<input checked="" type="checkbox"/> *
DTC origin "Primary Memory" supported	<input checked="" type="checkbox"/> *
Status Availability Mask	0xff*
Snapshots Per DTC	5*
First non-permanent snapshot	1*
Max Number Event Entry Prm	8
Type Of DTC Supported	ISO15031_6
Time Based Debouncing Timeout value Data Type	16 Bit
Dem_Prelim() has config parameter	<input checked="" type="checkbox"/>
BSW Error Buffer	
Bsw Error Buffer Size	4*
Add Error Buffer Size	1*
Dtc Groups	
Powertrain DTC Group [DTC#]	0x000001
Chassis DTC Group [DTC#]	0x400000
Body DTC Group [DTC#]	0x800000
Network Communication DTC Group [DTC#]	0xC00000
NvRam Block IDs	
Manually Add an NvRam Block ID	...
Nvram Block Id	Add
Manual Configuration	
Operation Cycles	
Create New Operation Cycle	...
Data Objects	
Create New Data Object	...
Extended Data Records	
Create New Extended Data Record	...
Indicators	
Create New Indicator	...
Events	
Create New Event	...

Figure 8-9 Main Configuration Window

The main configuration window is separated in different sections. For a detailed description see the following chapters.

8.2.5 Common

These switches are informational and show the current settings of the delivered DEM library files.



Caution

The precompiled settings cannot be changed, due to compatibility with the delivered library files.



Info

When you received source files of the DEM, the Development Error Detection ("Dev Error Detect") can be switched off or on. When this option is switched on, development errors are reported to the development error tracer (DET).

8.2.6 Software Component Template

Use RTE: RTE-support can be en-/disabled.



Caution

When you received library files of the DEM, please validate that the size of event ID data type is not different between DEM types and RTE created types when changing this setting.



Caution

When enabling the RTE-support please ensure that there is at least one external event configured. Map the associated port interfaces via DaVinci Network Designer.



Info

The RTE-generated data type of the DEM "Dem_EventIdType" will be removed by the RTE if there is no event with event destination other than internal. That optimization will lead to a compile-error because the DEM needs that data type anyway.

Software Component Name: Used in the service port export as service component template name.



Info

Change this name if you try to import the Dem components of several ECUs and have problems with name clashes.

Use Dcm Compatible Ports:

If this option is enabled, the port interface 'CallbackGetDataValueByDataID' is replaced by a separate port interface CallbackGetDataValue_<DID> per DID.

The operation GetDataValueByDataIdentifier is redefined to be compatible to the DCM counterpart ReadData.

This makes it possible to trigger the same runnable with both operations.



Caution

To work correctly, the runnable may not return neither E_PENDING nor E_NOT_OK and may not depend upon CheckConditionRead being called before the runnable is triggered.

In case of an error the SWC shall return E_OK and fill the whole data buffer with 0xFF bytes instead of returning E_NOT_OK.

If the option is disabled, the ports are generated as defined by AUTOSAR.



Info

This option has no effect if the RTE support is disabled.

8.2.7 Postbuild Settings

Postbuild Start Address: Enter the start address of your post-build configurable data of the DEM.

See [8] for more information.



Info

Configuration variant post-build is not supported (see chapter 3 for supported configuration variant(s)).

8.2.8 Miscellaneous

Dem Version: Version of the DEM embedded source.

Task Time [ms]: Enter the time period for the scheduling of the Dem_MainFunction [ms].



Info

This specified time must be provided by the ECU manager.

Support Fim Notification API: As the FIM is part of the AUTOSAR system, the FIM will be notified of event changes.

DTC origin “Primary Memory” supported: If enabled, this origin is available at the ECU.

Status Availability Mask: The status availability mask as configured in the CDD

Snapshots Per DTC:

8.2.11 NvRam Block IDs

The NvM-API mentioned in 6.2.2.3 (`NvM_SetRamBlockStatus()`) accesses the NvRAM blocks configured in the NvRAM manager.

Manually add an NvRAM BlockID: This option is to configure a **non-existing** NvRAM BlockID for use with the DEM.

Pressing this button will only create a symbolic name for an NvRAM Block in the NvM configuration, so you can choose that block when selecting the NvRAM Block ID in the next settings.

This NvRAM Block ID is only used as a placeholder for GENy configuration and will **not** be exported to an EcuC file when it is created here.



Caution

Please ensure that the NvRAM BlockID is completely configured in the NvRAM manager later, if you choose that ID in the following settings!

NvRAM Block ID / Core NvRam Block: The drop down list contains all available (configured) NvM Block IDs. Select one for the DEM to store its non-volatile data at.

GENy will configure the following attributes in the `NvM/NvmBlockDescriptor` in the EcuC file:

`NvmRamBlockDataAddress` is set to `Dem_NvData`

`NvmInitBlockCallback` is set to `Dem_NvDataInit` (see ch. 6.4.1)

`NvmSelectBlockForReadall` is set to `true`.

If the chosen NvRAM Block ID was placeholder only (“manually added”), it now becomes a **partly configured** NvRAM block after saving the EcuC file. Don’t forget to configure the leftover parts!

8.2.12 Manual Configuration

The different configuration elements can either be added in the main configuration window

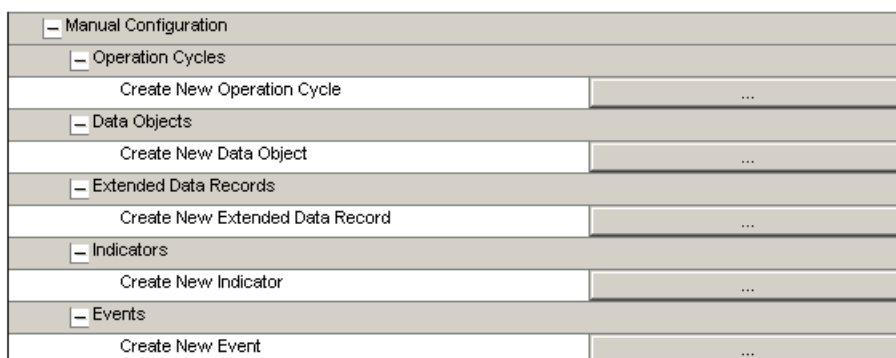


Figure 8-10 Manual Configuration of the DEM via Main Configuration Window

or by using Right-Mouse-Button in the left side tree view.

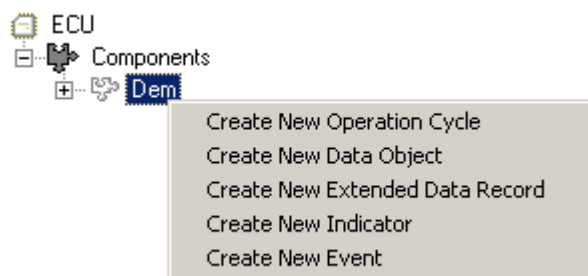


Figure 8-11 Manual Configuration of the DEM via Right-Mouse-Button in Tree View

Create New Operation Cycle: A new Operation Cycle can be added.

Create New Data Object: A new Data Object can be added.

Create New Extended Data Record: A new Extended Data Record can be added.

Create New Indicator: A new Indicator can be added.

Create New Event: A new Event can be added.



Info

For configuring an event, you need an operation cycle and indicator(s) if required. To accelerate the configuration, please check that the necessary objects do exist before creating a new event.



Info

With Vehicle System Group (VSG) mode (see chapter 7.1.3) each event manually created in GENy is put into the *base variant*.

8.2.12.1 Operation Cycles Window

AUTOSAR predefines 4 Operation Cycles which are preconfigured. You can add your own cycles here by using the Right-Mouse-Button in the tree view or by pressing the “Add Operation Cycle” button in the main configuration window.

	Operation Cycle	Remove this operation cycle	Symbolic Name
DEM_IGNITION	Ignition ON / OFF Cycle		
DEM_OBD_DCY	OBD Driving Cycle		
DEM_POWER	Power ON / OFF Cycle		
DEM_WARMUP	OBD OBD Warm up Cycle		
DEM_OP_CYCLE_NEW4	Ignition ON / OFF Cycle	...	DEM_OP_CYCLE_NEW4

Figure 8-12 Operation Cycles Window

The set “Symbolic Name” for this operation cycle can be used to identify this operation cycle in a more human readable way. In the generated source there will be created a `#define` using this name.

**Info**

The AUTOSAR configuration model does not permit the definition of new, independent cycles. You can only create new labels (symbolic names) for existing cycles.

8.2.12.2 Data Object Window

A data object describes one DataIdentifier – including its symbolic name, Data ID and Data Size. For each data object a separate port interface can be created by adding individual Port Prototype Name. If the same Port Prototype Name is used a common port interface will be used.

	Remove this DataID reference	Symbolic Name	Data ID	Freeze Frame Id Data Size [bytes]	Get Data Val By Data Id List Fnc
DATA0	...	DATA0	0x1010	3"	CBValByDID_DATA0

Figure 8-13 Data Object Window

**Info**

To get DCM compatible ports, use the option "Use Dcm Compatible Ports" described in chapter 8.2.6.

8.2.12.3 Extended Data Records

The following ExtendedData Record can be imported via CDD (configuration see 8.1.3.1):

	Remove this extended data record	Symbolic Name	Extended Data Record Number	Extended Data Record Data Size [bytes]	Get Ext Data Record Fnc
OccurrenceCounter	...	OccurrenceCounter	0x1	1	
RECORD55	...	RECORD55	0x55	4"	CBExtDataRec_0x55

Figure 8-14 ExtendedData Records

OccurrenceCounter:

This ExtendedData Record will be available when a CDD configured as described in chapter 8.1.3.1 has been imported. It has a fixed size of 1 Byte (unsigned). The record number is configurable in the CDD. You can neither change the size and the name of this ExtendedData Record nor remove it from the configuration.

For this ExtendedData Record, an internal function of the DEM exists and therefore, no port prototype must be configured.

Optional: further, user defined Data Records:

After adding a new record you must define the (1 Byte) record number and the size of this record. The data values for ExtendedData Records are provided by the SWC. Therefore the callback function that is written into Port Prototype Name field will be called, if it is necessary to update the Extended Data Record of any event.

The suggestion of GENy for the function name (Get Ext Data Record Fnc) is CBExtDataRec_<RecordNumber>. In spite of AUTOSAR configuration, which limits the

name of the port interface to CBExtDataRec_<RecordNumber>, you can set the full name of the function here.



Info

The user defined extended records will be updated each time the event gets failed (status bit “testFailed” changes from 0 to 1), when all storage conditions are fulfilled.



Caution

The biggest extended data record must fit in the configured page buffer of the DCM. See technical reference of the DCM for more details.

8.2.12.4 Indicator Window

AUTOSAR does not have any predefined Indicators and if your ECU doesn't need them, you needn't define any.

To define the first Indicator you can either press in the main configuration window the button “Add Indicator” or use in the left side tree view the Right-Mouse-Button option “Add New Indicator”.

After defining the first Indicator you can also use Right-Mouse-Button in the Indicator window to add further Indicators:

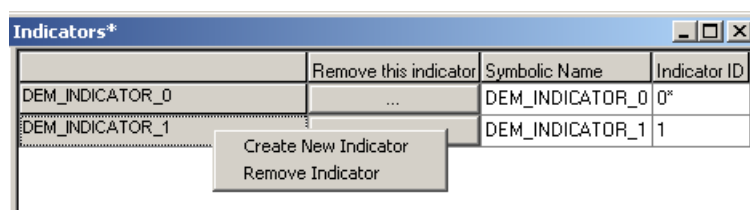


Figure 8-15 Indicator Window

Enter a symbolic name and assign an IndicatorID.

The set “Symbolic Name” of this Indicator can be used to identify it in a more human readable way. In the generated source there will be created a #define using this name.

8.2.12.5 Event Configuration Window

When starting with a new configuration, typically there exist some preset events. These are required by other AUTOSAR SW-C modules and are typically configured as “Internal Event”: They have no DTC number assigned thus cannot be requested externally by diagnostic services. They don't store environmental data with them (as such could not be read via diagnostics) but they have an event status that can be used by FIM.

Such predefined internal events have greyed out the “Symbolic Name”; they cannot be removed from the configuration.



Info

Starting with Dem version 2.14.00, the interpretation of the “Symbolic Name” was modified to act closer to the AUTOSAR requirements:

When creating #defines for EventID and IndicatorID, the Dem will now add prefix Dem_ to the configured Symbolic Names to avoid namespace clashes. See your release notes for more details.

Example: the symbolic name for the event CAN_E_TIMEOUT will now result in file Dem_IntErrId.h to a line like that:

```
#define Dem_CAN_E_TIMEOUT          42
```

	Remove this event	Event Kind	Symbolic Name	Operation Cycle
CAN_E_TIMEOUT	...	BSW	CAN_E_TIMEOUT	DEM_IGNITION
CANIF_E_FULL_TX_BUFFER	...	BSW	CANIF_E_FULL_TX_BUFFER	DEM_IGNITION
CANIF_E_STOPPED	...	BSW	CANIF_E_STOPPED	DEM_IGNITION
CANIF_E_INVALID_RXPDUID	...	BSW	CANIF_E_INVALID_RXPDUID	DEM_IGNITION
CANIF_E_INVALID_TXPDUID	...	BSW	CANIF_E_INVALID_TXPDUID	DEM_IGNITION
PDUR_E_INIT_FAILEDCAN	...	BSW	PDUR_E_INIT_FAILEDCAN	DEM_IGNITION
CANTP_E_OPER_NOT_SUPPORTED	...	BSW	CANTP_E_OPER_NOT_SUPPORTED	DEM_IGNITION
NVM_E_REQ_FAILED	...	BSW	NVM_E_REQ_FAILED	DEM_IGNITION
NVM_E_INTEGRITY_FAILED	...	BSW	NVM_E_INTEGRITY_FAILED	DEM_IGNITION
ECUM_E_ALL_RUN_REQUESTS_KIL	...	BSW	ECUM_E_ALL_RUN_REQUESTS_KILLED	DEM_IGNITION
ECUM_E_RAM_CHECKED_FAILED	...	BSW	ECUM_E_RAM_CHECKED_FAILED	DEM_IGNITION
WDGM_E_SET_MODE	...	BSW	WDGM_E_SET_MODE	DEM_IGNITION
WDGM_E_ALIVE_SUPERVISION	...	BSW	WDGM_E_ALIVE_SUPERVISION	DEM_IGNITION
WDG_E_MODE_SWITCH_FAILED	...	BSW	WDG_E_MODE_SWITCH_FAILED	DEM_IGNITION
MCU_E_CLOCK_FAILURE	...	BSW	MCU_E_CLOCK_FAILURE	DEM_IGNITION
EEP_E_COM_FAILURE	...	BSW	EEP_E_COM_FAILURE	DEM_IGNITION

Create New Event

Figure 8-16 Event Configuration Window

To define your own event, use one of the 3 choices: press in the main configuration window the button “Add Event”, use in the left side tree view the Right-Mouse-Button option “Add New Event” or (as seen above

**Info**

For configuring an event, you need an operation cycle and indicator(s) if required. To accelerate the configuration, please check that the necessary objects mentioned above do exist before creating a new event.

**Caution**

The DEM requires at least one external event (assign a DTC Number to at least one event). Otherwise the RTE will not generate the `typedef Dem_EventIdType` and the compilation will not be possible.

Configurable Options		DEM_my_first_event
Remove this event	...	
Event Kind	SWC	
Symbolic Name	DEM_my_first_event	
Operation Cycle Ref	DEM_IGNITION	
Healing Allowed	<input checked="" type="checkbox"/>	
Healing Cycle Ref	DEM_IGNITION	
Healing Cycle Counter	10	
Init Monitor Name	DEM_InitMonitor_0x123456	
Event Priority	0*	
Event Destination	PRIMARY_MEMORY	
DTC Settings		
DTC	0x123456	
DTC Group	POWERTRAIN	
DTC Severity	CHECK_AT_NEXT_HALT	
DTC Kind	NORMAL	
DTC Support Warning Indicator	<input type="checkbox"/> *	
DTC Functional Unit	255	
Event Pre-Debouncing		
Pre Debounce Algorithm Class	Counter based	
Count In Step Size	1*	
Count Out Step Size	1*	
Jump Up	<input checked="" type="checkbox"/> *	
Jump Down	<input type="checkbox"/> *	
Snapshot Records		
FF Prestorage Supported	<input type="checkbox"/> *	
Create New Snapshot Record	...	
Freeze Frame Class Ref	Add	
Extended Data Records		
Extended Data Ref		Add
Extended Data Ref	Delete	OccurrenceCounter
Indicators		
Indicator Attribute		Add
IndicatorConfig		Delete
Linked Indicator	DEM_INDICATOR_0	
Indicator Behavior	Blinking	
TriggerOnEvent Callback Functions		
Trigger On Event Status Target Ref		Add
Trigger On Event Status Target Ref	Delete	CBStatusEvt_DEM_my_first_event
TriggerOnDtc Callback Functions		
Trigger On DTC Status Target Ref		Add
Trigger On DTC Status Target Ref	Delete	CBStatusDTC_DEM_my_first_event

Figure 8-17 Configurable Options of an Event

All event specific parameters have to be adapted in this window. Preconfigured options may not be changed.

Event Kind:

Define the origin of the event (i.e. who is setting/modifying the event data) SWC (SW-Component) or BSW (Basic software).

Events of Kind SWC will offer a Port Interface “Diagnostic Monitor” to set and read the event data via RTE. Such events must be set via RTE/Dem_SetEventStatus only.

Events of Kind BSW will be queued in the Dem and worked in Dem's cyclic task to get a defined call context (for notifications and callouts). Such events must be set via Dem_ReportErrorStatus only.



Caution

Either modify an event only through a monitor on application level (i.e. above the RTE) and select "Event Kind: SWC" here, or through a monitor on basic software level and select "Event Kind: BSW" here. Prevent modification from both sides for an event – else you may run into data corruption and undefined behavior resulting from infrequent simultaneous access.

Do not select "Event Kind: SWC" for BSW events and do not manually add a PortInterface (e.g. in DaVinci) for BSW events, for example to be able to use GetEventStatus via RTE: you will sporadically get incomplete event status when reading these events and you will sporadically corrupt DEM internal data when writing such events which can trigger undefined behavior. See chapter 5.5 *Call Context* for some background info.

Symbolic Name:

The names can only be changed with own added events. The names of pre-added events cannot be changed.

Operation Cycle Ref:

An operation cycle has to be added to an event.



Caution

Only if the operation cycle is started, the status of the event can be changed.

Healing Allowed:

Use this configuration switch to choose, if it will be possible to heal this event or not.

Healing Cycle Ref:

The cycle used for healing can be set here. Usually the "Operation Cycle" is chosen.

Healing Cycle Counter:

If an event can be healed, the necessary count of healing cycles until the event is healed can be adapted here.



Info

The healing cycles are only counted, when the event is tested 'passed' during that operation cycle.

Init Monitor Name:

This function will be called to inform the monitor function if the operation cycle for this event is (re-)started (DEM_INIT_MONITOR_RESTART) or if the event is cleared from the event memory (DEM_INIT_MONITOR_CLEAR).

In spite of AUTOSAR configuration, which limits the name of the port interface to `CBInitEvt_<EventName>`, you can set the full name of the function here.

When you are importing from CDD, the default name will be `CBInitEvt_<Symbolic Name>`. You may change the name after the import.

Event Priority:

The priority value influences the displacement strategy if the storage buffer is full. The event with the lowest priority will be discarded, when the new event has a higher priority.



Info

Different DTCs with equal priority value are permitted. The priorities are limited to range 0...255 as required by AUTOSAR.

Event Destination:

The event destination is currently restricted to the primary memory, because the other memory locations are not supported.



Info

If an event has **no DTC number** assigned, it becomes an **INTERNAL EVENT** which cannot be accessed via diagnostic services and which does not store environmental data! This is the default for all BSW errors.

INTERNAL EVENTS do have an event status, which is used by the FIM.

8.2.12.5.1 DTC Settings

DTC:

This number is the external representation of the event.



Info

If an event has **no DTC number** assigned, it becomes an **INTERNAL EVENT** which cannot be accessed via diagnostic services and which does not store environmental data! This is the default for all BSW errors.

INTERNAL EVENTS do have an event status, which is used by FIM.



Info

Each event can get just one DTC number. Event - DTC is a 1:1 correlation (or 1:0 for internal events).

DTC Group:

The DTC group will automatically be assigned to an event, based on the configured DTC number and the available DTC groups.

DTC Severity:

The severity can be selected here.

DTC Kind:

Until OBD is supported, the DTC kind is fixed to 'NORMAL' DTCs.

DTC Support Warning Indicator:

If this parameter is enabled Bit 7 'WarningIndicatorRequested' of the DTC status will be set when the DTC is failed (Bit 0 'TestFailed' is set).

DTC Functional Unit:

This parameter is necessary for the report of severity information. It identifies the corresponding basic vehicle / system function which reports the DTC.

8.2.12.5.2 Event Pre-Debouncing

Pre Debounce Algorithm Class:

The current DEM version supports the 'counter based' and 'timer based' de-bounce algorithm.

Count In Step Size (counter based):

This value represents the step size by which the DTC fault detection counter is increased for each 'prefailed' call. Reaching value +127 will make an event become 'failed'.

Count Out Step Size (counter based):

This value represents the step size by which the DTC fault detection counter is decreased for each 'prepassed' call. Reaching value -128 will make an event become 'passed'.

Failed Timeout Value (timer based):

This value represents the time which has to elapse until an event which was reported with 'prefailed' become 'failed'.

Passed Timeout Value (timer based):

This value represents the time which has to elapse until an event which was reported with 'prepassed' become 'passed'.

Jump Up (counter based):

If the direction of the DTC fault detection counting will change from decrementing to incrementing and the value is below zero, a jump to zero will be made before incrementing.

Jump Down (counter based):

If the direction of the DTC fault detection counting will change from incrementing to decrementing and the value is above zero, a jump to zero will be made before decrementing.

**Info**

AUTOSAR forbids using Jump-Down if Jump-Up was deactivated. Rationale in [1]:
"‘passed’ results could be faster obtained than ‘failed’ results. This is critical from legal point of view."

8.2.12.5.3 Snapshot Records (FreezeFrames)

FF Prestorage Supported:

If this parameter is enabled, the prestorage of snapshot records is supported by the assigned event.

Create New Snapshot Record:

To create a new snapshot record, click the button in the event configuration window (see Figure 8-17) or use the right-mouse-button option in the tree window.

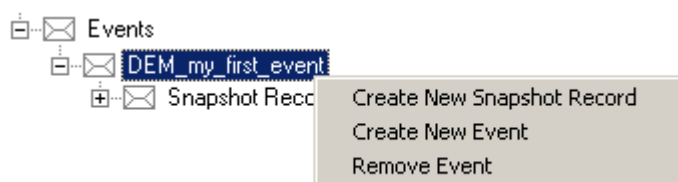


Figure 8-18 Create New Snapshot Record



Info

For defining a snapshot record, you need data objects. To accelerate the configuration, please check that the necessary data objects do exist before creating a new snapshot record.

Each snapshot record includes a list of data objects. You can use one data object in several snapshot records and each snapshot record may have any number of data objects.

	Remove this snapshot record	Symbolic Name	Data Objects	
DEM_FFTYPE_NEW3	...	DEM_FFTYPE_NEW3	Add	Delete
			Data Object	
			DATA0	

Figure 8-19 Snapshot Record Window



Info

All configured data objects of one event will be reported in one snapshot record. The snapshot record number of each stored snapshot record will be incremented per event (first confirmed snapshot record: 0x01, second confirmed snapshot record: 0x02,...).



Caution

The complete snapshot record must fit in the configured page buffer of the DCM. See technical reference of the DCM for more details.

8.2.12.5.4 Extended Data Records

All (external) events will use the identical set of Extended Data Records imported via CANdelaStudio CDD file. To modify the set please refer chapter 8.2.12.3 *Extended Data Records* or 8.1.3 *Extended Records* via CANdela.

8.2.12.5.5 Indicators

Linked Indicator:

If the event has status 'failed' (Bit 0 'TestFailed' is set), this can be shown by one (or more) indicator(s). Choose one of the indicators created before from the drop down box.

Indicator Behavior:

Select to the required behaviour 'active' or 'blinking' for each linked indicator.

8.2.12.5.6 TriggerOnEvent Callback Functions

If the event status is changing one or multiple trigger functions can be called. The event status reports all supported status bits.

The suggestion of GENy for the function name (Port Prototype Name) is `CBStatusEvt_<EventSymbolicName>`. In spite of AUTOSAR configuration, which limits the name of the port interface to `CBStatusEvt_<EventName>`, you can set the full name of the function here.

8.2.12.5.7 TriggerOnDtc Callback Functions

If the DTC status is changing one or multiple trigger functions can be called. The DTC status reports all supported status bits but the bits disabled by the `DTCStatusAvailabilityMask` (see 8.1.1).

The suggestion of GENy for the function name (Port Prototype Name) is `CBStatusDTC_<EventSymbolicName>`. In spite of AUTOSAR configuration, which limits the name of the port interface to `CBStatusDTC_<EventName>`, you can set the full name of the function here.



Info

The `TriggerOnDtc` callback functions can only be configured for events with a DTC Number.

8.2.13 Software Component Template

When the RTE is part of the configuration, GENy generates during each generation process a software component template. The name of the software component can be specified in GENy with the configuration parameter "Software Component Name". The name of the XML file consists of the software components name with the postfix "_swc" and the file extension ".arxml" (`<SoftwareComponentName>_swc.arxml`). The default is "Dem_swc.arxml".

The software component template contains the AUTOSAR service port description for the ports provided by DEM and can be used by the RTE to provide these ports to the application.

9 AUTOSAR Standard Compliance

9.1 Deviations

9.1.1 APIs and Features not Supported

See chapter 4.1.

9.1.2 AUTOSAR Defined APIs that Differ in this Implementation

9.1.2.1 Service Dem_ClearDTC

When the API is called with the wrong DTCKind, the return value is DEM_CLEAR_WRONG_DTC.

See chapter 6.2.5.3.1

9.1.2.2 Service Dem_SetEventStatus

The return value DEM_E_QUEUE_OVERFLOW has been defined additionally to the return values described in [1].

See chapter 6.2.3.1.

9.1.2.3 Service Dem_ResetEventStatus

The return value DEM_E_QUEUE_OVERFLOW has been defined additionally to the return values described in [1].

See chapter 6.2.3.2.

9.1.2.4 Service Dem_SetOperationCycleState

The return value DEM_E_QUEUE_OVERFLOW has been defined additionally to the return values described in [1].

See chapter 6.2.3.5.

9.1.2.5 Service Dem_ReportErrorStatus

The development error value DEM_E_QUEUE_OVERFLOW has been defined additionally to the development error values described in [1].

See chapter 6.2.4.1.

9.1.2.6 Interface Rte_Call_Dem_<ConfiguredName>_DTCStatusChanged

The parameter DTCKind has been defined additionally to the parameters described in [1].

See chapter 6.5.1.3.

9.1.3 Require Ports not Supported

CallbackInitMonitorForFunction and CallbackGetFaultDetectCounter – see chapter 6.6.1.2.

9.2 Additions/ Extensions

9.2.1 Development Error Reporting – Include Structure

The include structure described by AUTOSAR is extended, when the development error reporting is activated.

See chapter 4.6.1.

9.2.2 Development Error Reporting – Internal Debug Codes

The following error codes are reported additionally to the errors defined in [1]:
Internal debug codes – see chapter 4.6.1.

9.2.3 Service Dem_GetOccurrenceCounter

The API `Dem_GetOccurrenceCounter()` is an extension to AUTOSAR. It can be used to request the current occurrence counter for a given event.

See chapter 6.2.3.146.2.3.14.

9.2.4 Name Description of Configurable Interfaces (Notifications)

AUTOSAR requires specific names for the notifications. In GENy these names are suggested, but can fully be set by the user.

See chapter 8.2.12.

9.2.5 Interface Rte_Call_Dem_<ConfiguredName>_GetDataValueByDataIdentifier

It is possible to use DCM compatible port interfaces. This is an extension to AUTOSAR.

See chapter 8.2.6.

9.2.6 Port Names Length Limitation

AUTOSAR limits the length of port names to 31 characters. In contrast naming conventions require port names longer than 31 characters.

See Bugzilla 16927.

9.2.7 Version Information

The return parameters for the `Dem_GetVersionInfo()` API are defined unequally. The implemented version is described in chapter 6.2.1.

9.3 Limitations

9.3.1 Limits Checked During Configuration

See not supported features in chapter 4.1:

- event combination is not supported
- the mapping of events to DTCs is “1 to 1” or “1 to 0” and not “n to 1” or “1 to n”
- until OBD is supported, the DTC kind is fixed to ‘NORMAL’ DTCs (see chapter 8.2.12.5.1)
- the event destinations are currently restricted to ‘PRIMARY_MEMORY’ or ‘INTERNAL’, because the other memory locations are not supported (see chapter 8.2.12.5)
- the pre-debounce algorithm is fixed to ‘Counter based’ for all DTCs (see chapter 8.2.12.5.2)

10 Glossary and Abbreviations

10.1 Glossary

Term	Description
Callback Function	Callback functions are implemented by the Dem and can be invoked by other modules.
Callout Function	At its configurable interfaces the DEM defines callout functions to ask for data values. The declarations of the callout functions are provided by the BSW module. It is the integrator's task to provide the corresponding function definitions.
Notification	At its configurable interfaces the Dem defines notifications that can be mapped to callback functions provided by other modules. The mapping is not statically defined by the Dem but can be performed at configuration time.
Event Kind "BSW"	A basic software component will set the event via API call Dem_ReportErrorStatus. To achieve data consistency, the Dem will queue such triggers and work them in its cyclic task. Never try to set such DTCs via API Dem_SetEventStatus – you may get undefined behavior. Do not manually add a Port Interface "Diagnostic Monitor" to such events.
Event Kind "SWC"	A software component will set the event via RTE (and therefore API Dem_SetEventStatus). GENy will generate a Port Interface "Diagnostic Monitor" (see chapter 6.6.1.1.1) for events having Event Kind "SWC" to implement that use case, and to allow the read access to the event's status.
External Event	<p>External events have a DTC Number assigned, so they can be read via diagnostic request. Therefore each external event must have configured the mandatory Extended Data Records.</p> <p>The data records are stored in the Primary Memory for the ChronoStack (and in Mirror/Secondary Memory for the Historical Stack).</p>
Internal Event	<p>Internal events do not have a DTC Number assigned, so they can not be read via diagnostic request and thus are invisible for an external tester.</p> <p>The Dem offers no API resp. Port Interface to read Extended Data Records for other software inside the ECU, so internal events do not need to store such data.</p> <p>Therefore internal will only store the DTC status but no other data.</p>
CANdelaStudio	Tool to compile, edit and display diagnostic data descriptions for control units in the automotive area.
GENy	Generation tool for CANbedded and MICROSAR components.

Table 10-1 Glossary

10.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software
CDD	CANdelia Diagnostic Data
DCM	Diagnostic Communication Manager
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DID	Data Identifier
DTC	Diagnostic Trouble Code
ECU	Electronic Control Unit
ECU-SM	ECU State Manager
FIM	Function Inhibition Manager
HIS	Hersteller Initiative Software
IUMPR	In Use Monitoring Performance Ratio
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
NvRAM	Non-volatile RAM
OBD	Onboard Diagnostics
OS	Operating System
RTE	Runtime Environment
SWC	Software Component
SWS	Software Specification

Table 10-2 Abbreviations

11 Contact

Visit our website for more information on

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

www.vector.com