VECTOR

MICROSAR FiM

Technical Reference

Function Inhibition Manager

Version 5.2.0

Authors	Thomas Necker
Status	Released

Document Information

History

Author	Date	Version	Remarks
Thomas Necker	2012-10-19	1.0.0	first version of FiM according AR4
Thomas Necker	2013-03-15	1.1.0	added calibration section, added OBD support, removed restriction regarding cyclic event evaluation
Thomas Necker	2013-06-28	1.2.0	changed include structure
Thomas Necker	2013-10-18	2.0.0	added info to FiM_DemTriggerOnEventStatus
Thomas Necker	2014-03-07	2.1.0	added Post-Build Loadable description, some smaller changes
Thomas Necker	2014-10-31	3.0.0	described format of version info numbers added section for integration in AR 3 stack
Thomas Necker	2015-03-20	3.1.0	described new 3.1.0 features
Thomas Necker	2016-01-08	4.0.0	removed calibration section and descriptions related to cyclic event evaluation
Thomas Necker	2016-11-18	4.2.0	added …VAR_INIT… for compiler abstraction / memory mapping
Thomas Necker	2017-10-27	5.0.0	rework for AR4.3
Thomas Necker	2017-11-24	5.1.0	small changes
Thomas Necker	2018-02-16	5.1.1	additional information regarding multi-core usage
Thomas Necker	2018-03-16	5.2.0	added description for calibration using vPblCalib

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_FIM.pdf	4.3.0
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	3.2.0
[3]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	1.6.0
[4]	Vector	MICROSAR Diagnostic Event Manager (DEM) for OBD, Technical Reference Addendum	see delivery
[5]	Vector	TechnicalReference_PostBuildLoadable.pdf	see delivery
[6]	Vector	TechnicalReference_vPblCalib.pdf	see delivery

Scope of the Document

This technical reference describes the general use of the Function Inhibition Manager Basic Software Module.

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Caution
This symbol calls your attention to warnings.

Contents

1 Component History 7

2 Introduction 8
2.1 Architecture Overview 9

3 Functional Description 11
3.1 Features 11
3.2 Major Changes in AUTOSAR 4.3 version of FiM 12
3.2.1 Usage of Monitor Status 12
3.2.2 FiM_GetPendingStatus 12
3.2.3 Initialization 13
3.2.4 Runtime 13
3.3 States 14
3.4 Main Functions 14
3.5 Error Handling 14
3.5.1 Development Error Reporting 14
3.5.2 Production Code Error Reporting 15

4 Integration 16
4.1 Scope of Delivery 16
4.1.1 Static Files 16
4.1.2 Dynamic Files 16
4.2 Include Structure 17
4.3 Compiler Abstraction and Memory Mapping 18
4.4 Critical Sections 19
4.5 Usage with Vector's Multi-Core/Multi-Partition Diagnostic Event Manager (DEM) 20
4.5.1 Usage in Multi-Partition Systems 21
4.5.2 Usage in Multi-Core Systems 21
4.6 Integration into AUTOSAR 3 Stack 22
4.6.1 RTE 22
4.6.2 SchM 22
4.6.2.1 Geny 22
4.6.2.2 DaVinci Configurator Pro 5 22

5 API Description 23
5.1 Type Definitions 23
5.2 Services provided by FiM 24
5.2.1 FiM_Init() 24

5.2.2 FiM_DemInit() ... 25
5.2.3 FiM_DemTriggerOnMonitorStatus() ... 26
5.2.4 FiM_DemTriggerOnEventStatus() ... 27
5.2.5 FiM_GetFunctionPermission() ... 28
5.2.6 FiM_GetFunctionPendingStatus() ... 29
5.2.7 FiM_GetVersionInfo() ... 30
5.2.8 FiM_InitMemory() ... 30
5.2.9 FiM_MainFunction() ... 31
5.3 Services used by FiM ... 32
5.3.1 EcuM_BswErrorHook() ... 32
5.4 Service Ports ... 33
5.4.1 Client Server Interface ... 33
5.4.1.1 Provide Ports on FiM Side ... 33
5.4.1.1.1 FunctionInhibition ... 33
5.4.1.2 Require Ports on FiM Side ... 33

6 Configuration ... 34
6.1 Configuration Variants ... 34
6.2 Configurable Attributes ... 34
6.2.1 Inhibition Configuration Codes ... 35
6.3 Measurement and Calibration ... 36
6.3.1 Measurable Objects ... 36
6.3.2 Calibration ... 36
6.4 Post-Build support ... 37
6.4.1 Post-Build Loadable ... 37
6.4.1.1 Initialization ... 37
6.4.1.2 Configuration of Post-Build Loadable ... 38

7 AUTOSAR Standard Compliance ... 39
7.1 Deviations ... 39
7.2 Additions/ Extensions ... 39
7.3 Limitations ... 39

8 Glossary and Abbreviations ... 40
8.1 Glossary ... 40
8.2 Abbreviations ... 40

9 Contact ... 41

Illustrations

Figure 2-1 AUTOSAR 4.2 Architecture Overview 9
Figure 2-2 Interfaces to adjacent modules of the FiM 10
Figure 3-1 Initialization sequence of the FiM 13
Figure 4-1 Include structure 17
Figure 4-2 Using FiM in a multi-core/multi-partition system 20

Tables

Table 1-1 Component history 7
Table 3-1 Supported AUTOSAR standard conform features 11
Table 3-2 Not supported AUTOSAR standard conform features 11
Table 3-3 Features provided beyond the AUTOSAR standard 12
Table 3-4 Service IDs 14
Table 3-5 Errors reported to DET 15
Table 4-1 Static files 16
Table 4-2 Generated files 16
Table 4-3 Compiler abstraction and memory mapping 18
Table 4-4 Rte_FiM_Type.h 22
Table 4-5 User Configuration File 22
Table 5-1 Type definitions 23
Table 5-2 FiM_Init() 24
Table 5-3 FiM_DemInit() 25
Table 5-4 FiM_DemTriggerOnMonitorStatus() 26
Table 5-5 FiM_DemTriggerOnEventStatus() 27
Table 5-6 FiM_GetFunctionPermission() 28
Table 5-7 FiM_GetFunctionPendingStatus() 29
Table 5-8 FiM_GetVersionInfo() 30
Table 5-9 FiM_InitMemory() 30
Table 5-10 FiM_MainFunction() 31
Table 5-11 Services used by the FiM 32
Table 5-12 FunctionInhibition 33
Table 6-1 Configurable Parameters 34
Table 6-2 Inhibition Configuration Codes 35
Table 6-3 Error Codes possible during Post-Build initialization failure 37
Table 8-1 Glossary 40
Table 8-2 Abbreviations 40

1 Component History

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

Component Version	New Features
1.0.0	first version of FiM according AR4
1.1.0	added support for ▶ calibration ▶ OBD ▶ cyclic event evaluation
1.2.0	optionally works without RTE
2.0.0	additional search algorithms for FiM_DemTriggerOnEventStatus
2.1.0	▶ added support for Post-Build Loadable ▶ optimizations for inhibition configuration
2.2.0	▶ no features in embedded software
3.0.0	▶ for new release of DaVinci Configurator Pro 5 ▶ slight differences in DET / return code behavior
3.1.0	▶ FiM can now handle SetEventAvailable feature of DEM ▶ split of Development Error Reporting from Development Error Detection
4.0.0	▶ for new release of DaVinci Configurator Pro 5 ▶ removed support for calibration and cyclic event evaluation
4.1.0	▶ internal changes
4.2.0	▶ SafeBSW
5.0.0	▶ rework for AR4.3 (development status)
5.1.0	▶ release of FiM
5.2.0	▶ support of vPblCalib

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module FiM as specified in [1].

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile, post-build	
Vendor ID:	FiM_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	FiM_MODULE_ID	11 decimal (according to ref. [3])

* For the precise AUTOSAR Release 4.x please see the release specific documentation.

The Function Inhibition Manager is responsible for providing a control mechanism for software components and the functionality therein. In this context, functionality can comprise one, several or parts of runnable entities with the same set of permission / inhibit conditions.

2.1 Architecture Overview

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

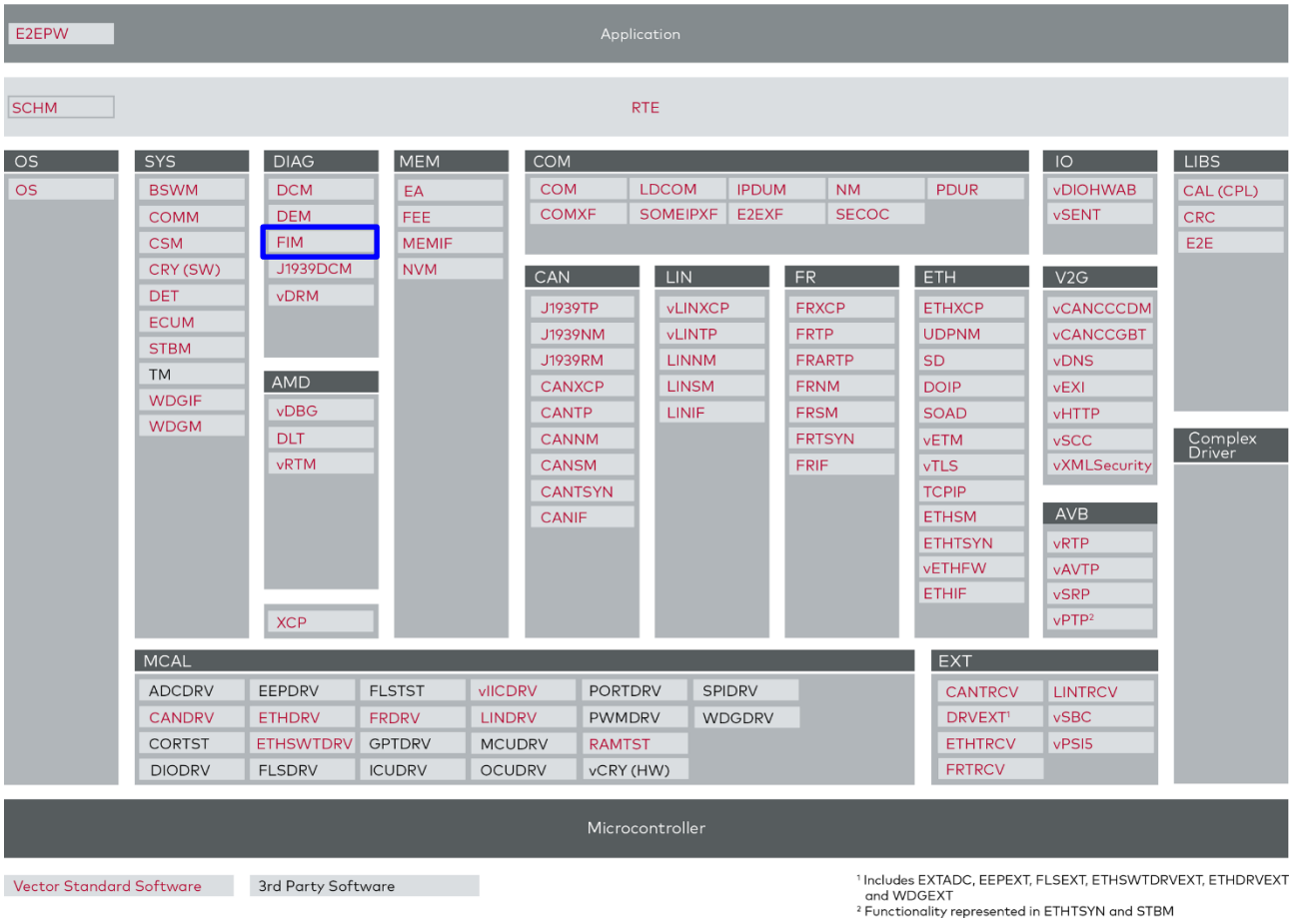

Figure 2-1 AUTOSAR 4.2 Architecture Overview

The next figure shows the interfaces to adjacent modules of the FiM. These interfaces are described in chapter 5.

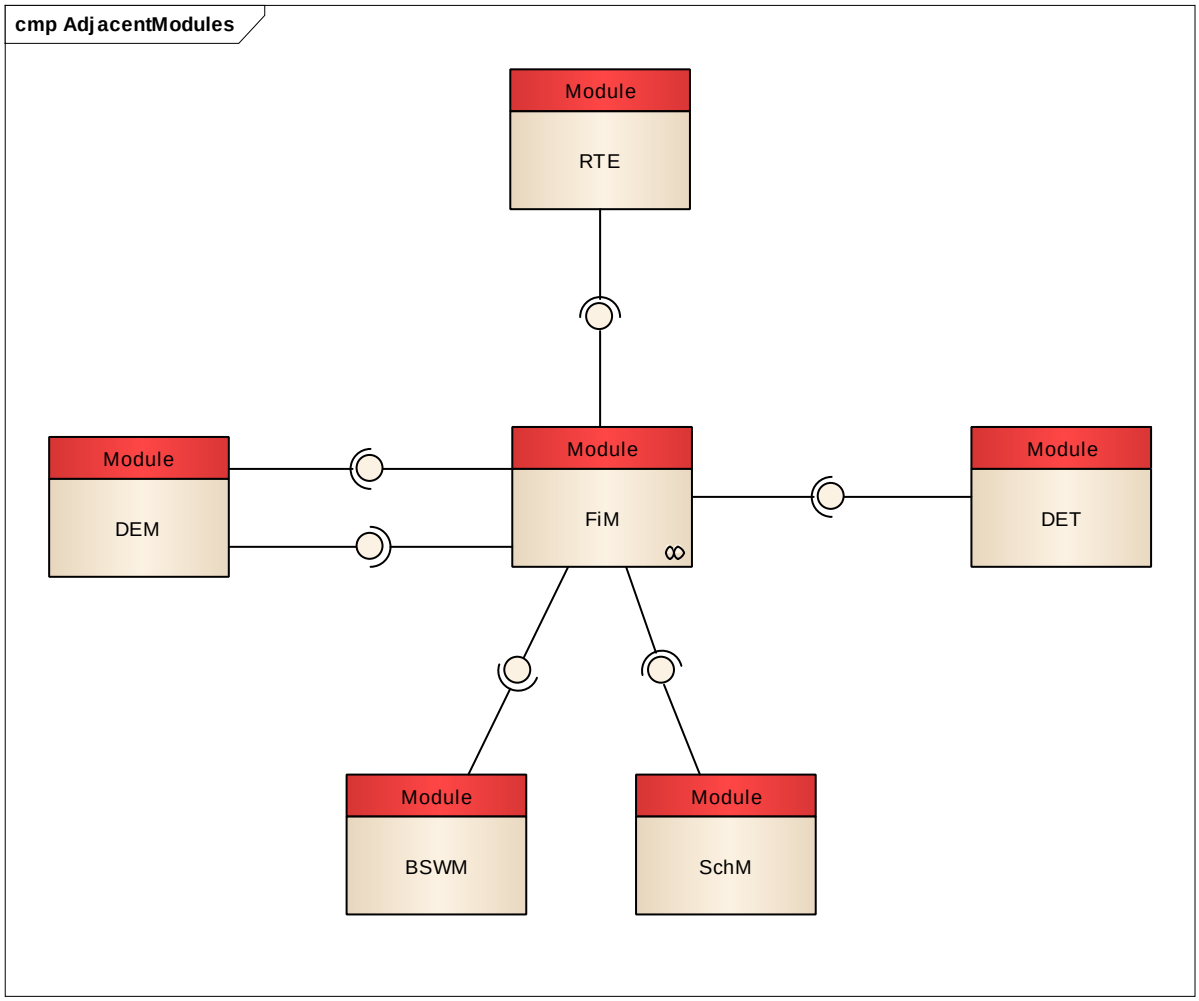

Figure 2-2 Interfaces to adjacent modules of the FiM

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 FiM are listed in chapter 5.4 and are defined in [1].

3 Functional Description

3.1 Features

The features in the following tables cover the complete functionality specified for the FiM.

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further FiM functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Get function permission by FID
Use DEM monitor status as inhibition condition
DEM notify FIM on monitor status change
Pre-compile configuration
AUTOSAR service component template generation
Development error detection over DET
Post-Build Loadable (certain configurations, see section 6.4.1)
Status variable tracking via measurement
Module individual post-build loadable update

Table 3-1 Supported AUTOSAR standard conform features

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

Not Supported AUTOSAR Standard Conform Features
Cyclic evaluation of DEM Events
Link time configuration
Summarized events
Monitored Components
Direct Configuration via calibration tool
Set function availability
Debugging
MICROSAR Identity Manager using Post-Build Selectable

Table 3-2 Not supported AUTOSAR standard conform features

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond the AUTOSAR Standard
FiM_InitMemory, see section 5.2.8
FiM_GetFunctionPendingStatus, see section 5.2.6
FiM_DemTriggerOnEventStatus, see section 5.2.4
Usage without RTE
Configuration via calibration tool using vPbICalib (see section 6.3.2)

Table 3-3 Features provided beyond the AUTOSAR standard

3.2 Major Changes in AUTOSAR 4.3 version of FiM

This section provides an overview of major changes in this 5.x version of the FiM compared to the previous 4.x version.

Note
This is only an overview and may not be complete. See remainder of Technical Reference for details.

3.2.1 Usage of Monitor Status

FiM now uses DEM's AUTOSAR 4.3 monitor status bits instead of UDS status bits for calculating FID states. Monitor status changes are communicated to FiM using the new API FiM_DemTriggerOnMonitorStatus while event status changes were communicated via FiM_DemTriggerOnEventStatus.

FiM_DemTriggerOnEventStatus is still available for OBD use cases where changes of the Pending status bits need to be passed from DEM to FiM.

Note
FiM is prepared to work with the multicore satellites of Vector's AUTOSAR 4.3 DEM.

However, currently DEM (satellite) needs to reside on the same core/partition as the FiM.

3.2.2 FiM_GetPendingStatus

FIDs that should be blocked depending on an event's pending status now need to be configured separately. Previously, the pending status of any event that was connected via a normal Inhibition configuration was considered.

3.2.3 Initialization

FiM_Init now does only a pre-initialization of the module. FiM is not ready for operation afterwards. DEM needs to call FiM_DemInit to complete FiM's initialization. The new initialization sequence is shown in Figure 3-1.

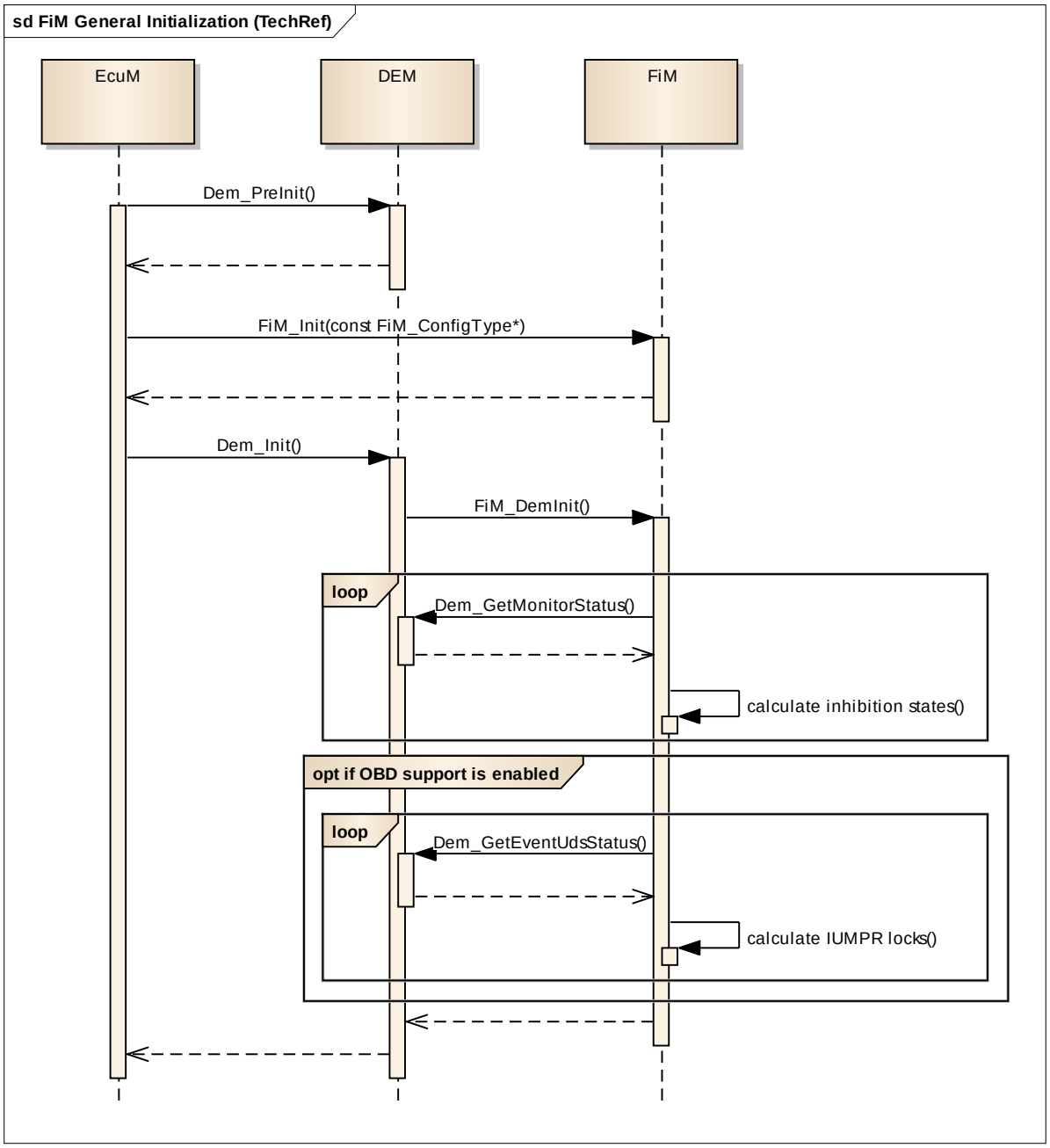

Figure 3-1 Initialization sequence of the FiM

3.2.4 Runtime

The calculation of FID states previously was mainly done in the context of DEM's event status update trigger (FiM_DemTriggerOnEventStatus). While this is still true for updates caused by pending status changes, updates caused by monitor status changes are now only handled partially during runtime of FiM_DemTriggerOnMonitorStatus. The rest is done during the call of FiM_GetFunctionPermission.

3.3 States

The FiM has the following internal states:

- inhibition state (one per event/inhibition combination)
- function pending state (one per FID that is connected to pending configuration if OBD is enabled)
- initialization state of FiM

3.4 Main Functions

The main function is empty and does not need to be scheduled.

3.5 Error Handling

3.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `FIM_DEV_ERROR_DETECT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported FiM ID is 11 decimal.

The reported service IDs identify the services which are described in section 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x00	FiM_Init
0x01	FiM_GetFunctionPermission
0x02	FiM_DemTriggerOnMonitorStatus
0x03	FiM_DemInit
0x04	FiM_GetVersionInfo
0x05	FiM_MainFunction
0x80	FiM_GetFunctionPendingStatus
0x82	FiM_DemTriggerOnEventStatus

Table 3-4 Service IDs

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

Error Code		Description
0x01	FIM_E_UNINIT	API function is called before the FIM module has been fully initialized
0x02	FIM_E_FID_OUT_OF_RANGE	FiM_GetFunctionPermission/ FiM_GetFunctionPendingStatus is called with wrong FID
0x03	FIM_E_EVENTID_OUT_OF_RANGE	Dem calls FiM with invalid/unconfigured EventId
0x04	FIM_E_PARAM_POINTER	API function is invoked with NULL Pointer
0x05	FIM_E_INIT_FAILED	initialization of FiM could not be completed, i.e. FiM_Init or FiM_DemInit failed (e.g. invalid configuration set selection)

Table 3-5 Errors reported to DET

Note
FIM_E_EVENTID_OUT_OF_RANGE (DEM calls FIM with invalid/unconfigured EventId according to [1]) is not used by FiM.

3.5.2 Production Code Error Reporting

Production code related errors are not defined for the FiM.

4 Integration

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

4.1 Scope of Delivery

The delivery of the FiM contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Source Code Delivery	Object Code Delivery	Description
FiM.c	■	n/a	Main source code file
FiM.h	■	n/a	Main header file
FiM_Types.h	■	n/a	Header file containing FiM types

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool Configurator 5.

File Name	Description
Fim_Cfg.h	This header file contains the configuration switches of the FiM
Fim_Lcfg.h	This header file provides forward declarations for the configuration values/tables of the FiM
Fim_Lcfg.c	This source file contains configuration values/tables of the FiM
Fim_PBcfg.h	This header file provides forward declarations for the configuration values/tables of the FiM
Fim_PBcfg.c	This source file contains post-buildable configuration values/tables of the FiM. For easier handling, this file is created in pre-compile configurations as well. If your build environment produces error messages due to this file not defining any symbols, please exclude it from the build.
FiM_swc.arxml	This AUTOSAR xml file is used for the configuration of the RTE. It contains the information to get prototypes of callback functions offered by other components.

Table 4-2 Generated files

4.2 Include Structure

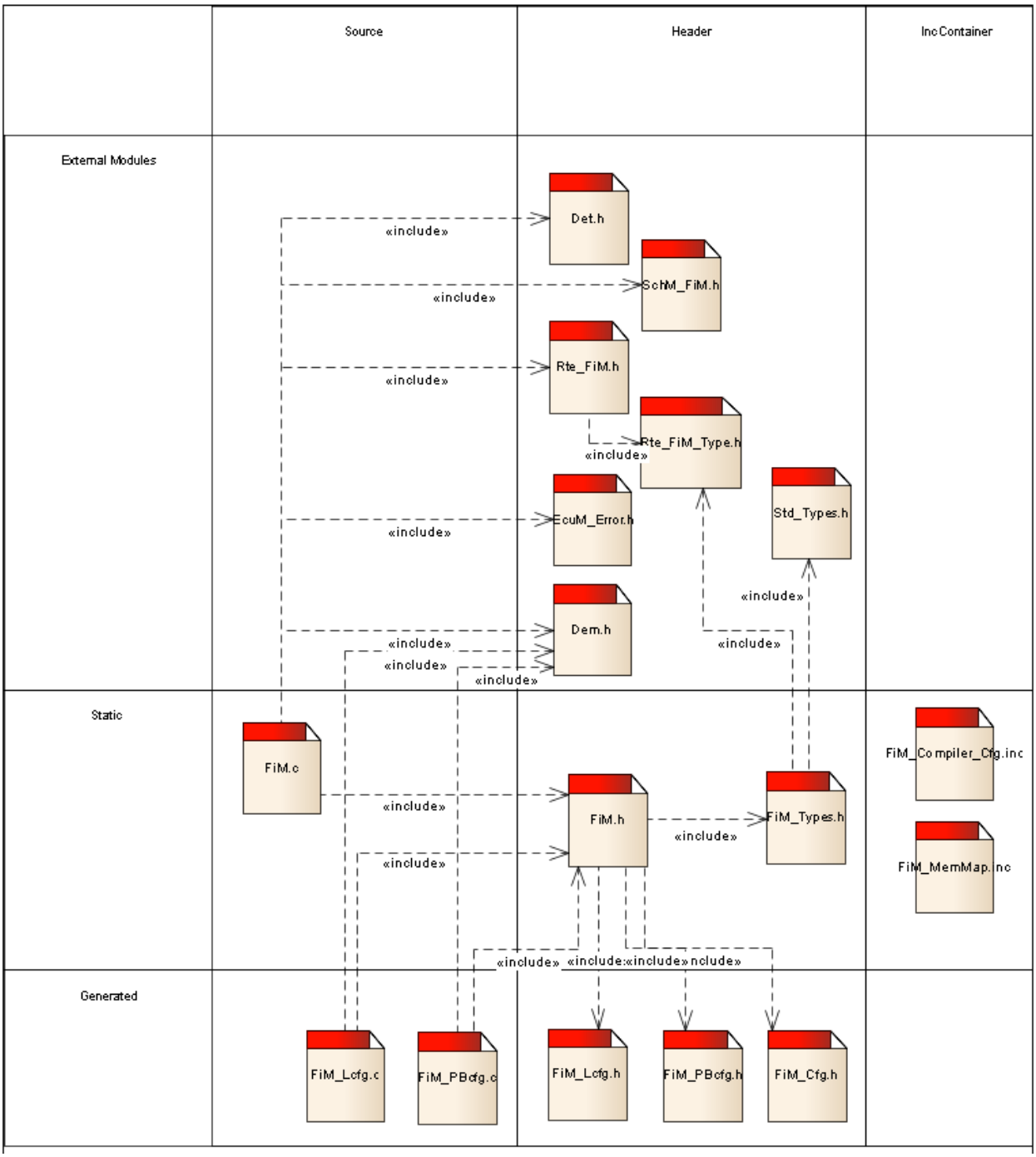

Figure 4-1 Include structure

4.3 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 of the FiM and illustrates their assignment among each other.

Compiler Abstraction Definitions / Memory Mapping Sections	FIM_CODE	FIM_CONST	FIM_VAR_NOINIT	FIM_VAR_INIT	FIM_PBCFG	FIM_PBCFG_ROOT	FIM_VAR_PBCFG
FIM_START_SEC_CODE FIM_STOP_SEC_CODE	■						
FIM_START_SEC_CONST_8BIT FIM_STOP_SEC_CONST_8BIT		■					
FIM_START_SEC_CONST_16BIT FIM_STOP_SEC_CONST_16BIT		■					
FIM_START_SEC_CONST_UNSPECIFIED FIM_STOP_SEC_CONST_UNSPECIFIED		■					
FIM_START_SEC_VAR_NOINIT_16BIT FIM_STOP_SEC_VAR_NOINIT_16BIT			■				
FIM_START_SEC_VAR_NOINIT_UNSPECIFIED FIM_START_SEC_VAR_NOINIT_UNSPECIFIED			■				
FIM_START_SEC_VAR_INIT_8BIT FIM_STOP_SEC_VAR_INIT_8BIT				■			
FIM_START_SEC_PBCFG FIM_STOP_SEC_PBCFG					■		
FIM_START_SEC_PBCFG_ROOT FIM_STOP_SEC_PBCFG_ROOT						■	
FIM_START_SEC_VAR_PBCFG FIM_STOP_SEC_VAR_PBCFG							■

Table 4-3 Compiler abstraction and memory mapping

4.4 Critical Sections

To protect internal data structures against unwanted modification, the FiM uses “Critical Sections” for blocking concurrent access.

FiM uses the critical section **`FIM_EXCLUSIVE_AREA_0`**. Length of locking is

- short during runtime to protect the inc-/decrementing of a counter or (re)setting a bit
- medium during initialization to protect the resetting of all counters and inhibition states.

It is recommended to be mapped to a global interrupt disabling.

AUTOSAR Schedule Manager APIs are used to handle critical sections (SchM_FiM.h is included).

Caution
You must take special care that the component implementing the critical section (e.g. AUTOSAR Schedule Manager) is already started before the FiM is run.

4.5 Usage with Vector's Multi-Core/Multi-Partition Diagnostic Event Manager (DEM)

Vector's multi-core/multi-partition DEM consists of one DEM satellite per partition and one DEM master.

- A DEM satellite
  - informs FiM about monitor status changes using the API **`FiM_DemTriggerOnMonitorStatus`**.
- The DEM master
  - informs FiM about event status changes (if OBD is enabled) using the API **`FiM_DemTriggerOnEventStatus`** and
  - initializes the FiM using the API **`FiM_DemInit`**.

Figure 4-2 shows an overview of the relevant software components and the communication paths between them.

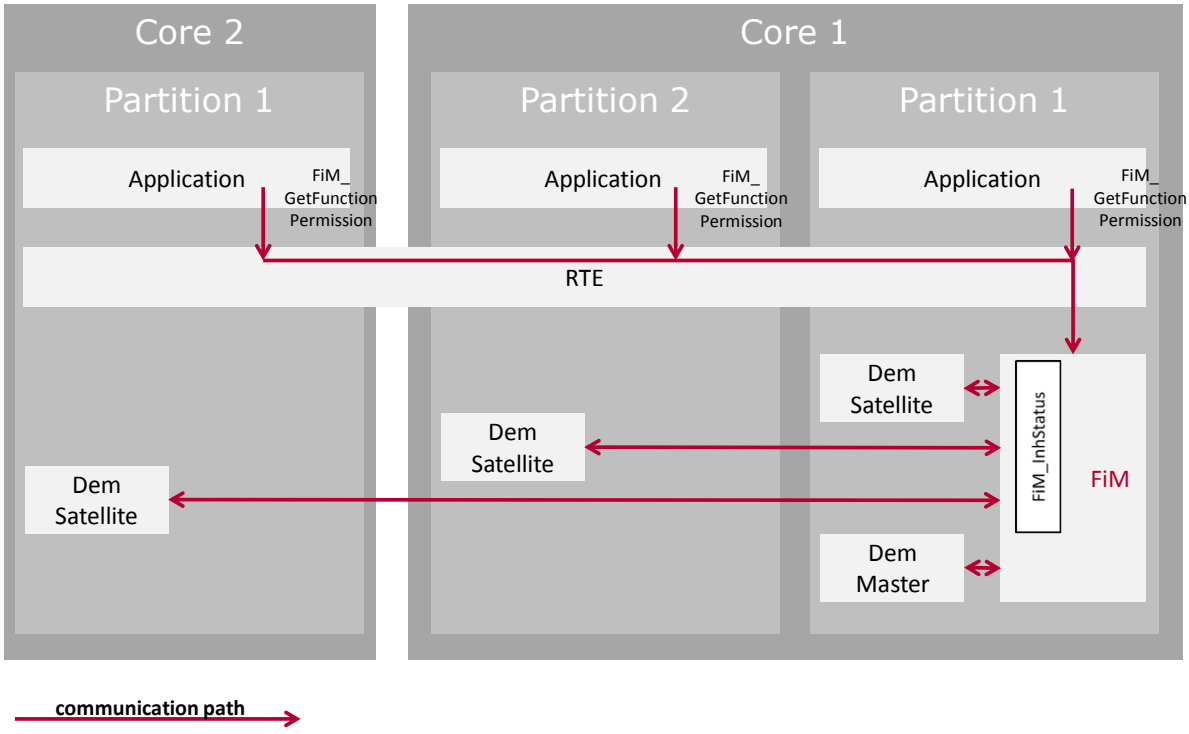

Figure 4-2 Using FiM in a multi-core/multi-partition system

4.5.1 Usage in Multi-Partition Systems

Usage of the FiM in multi-partition systems with different ASIL levels is only allowed if the following conditions are fulfilled:

- the partitions with DEM satellites get write access to the symbol **`FiM_InhStatus`** and
- FiM is located on the same partition as the DEM master.

Caution
Please take care that write access is only granted to the memory area defined by symbol **`FiM_InhStatus`** in file **`FiM_Lcfg.c`** or **`FiM_PBcfg.c`**.

Note
The function **`FiM_DemTriggerOnMonitorStatus`** is called in the context of a DEM satellite. This function potentially modifies the **`FiM_InhStatus`** variable.

Note
In addition FiM must fulfill certain pre-conditions to run in an ASIL partition. See the safety manual of your delivery for details.

Note
The functions **`FiM_DemTriggerOnEventStatus`** and **`FiM_DemInit`** are called in the context of the DEM master. These functions potentially modify FiM internal variables.

4.5.2 Usage in Multi-Core Systems

To protect concurrent write access from different cores the critical section **`FIM_EXCLUSIVE_AREA_0`** described in section 4.4 needs to be additionally mapped to a spinlock.

4.6 Integration into AUTOSAR 3 Stack

If the FiM is to be integrated into a stack according AUTOSAR 3 some manual adaptations need to be done that are described in this section.

Note
The FiM AUTOSAR 4 always requires a DEM according AUTOSAR 4. Integrating this DEM in an AUTOSAR 3 stack is not in the scope of this Technical Reference.

Since the interface to the DEM is always according AUTOSAR 4 only the interfaces to the following modules need attention:

- RTE
- SchM

4.6.1 RTE

An AUTOSAR 3 RTE does not provide the necessary include file `Rte_FiM_Type.h`. Create it and include `Rte_Type.h`:

Rte_FiM_Type.h

```
#if !defined (RTE_FIM_TYPE_H)
# define RTE_FIM_TYPE_H
# include "Rte_Type.h"
#endif  /* RTE_FIM_TYPE_H */
```

Table 4-4 Rte_FiM_Type.h

4.6.2 SchM

4.6.2.1 Geny

The Vector MICROSAR 3 SchM is configured in Geny.

In the SchM configuration create a BSW module support for FiM with one Exclusive Area **`FIM_EXCLUSIVE_AREA_0`**.

4.6.2.2 DaVinci Configurator Pro 5

In DaVinci Configurator Pro 5 include a user configuration file in the FiM module containing:

User Configuration File

```
#define SchM_Enter_FiM_FIM_EXCLUSIVE_AREA_0()
SchM_Enter_FiM(FIM_EXCLUSIVE_AREA_0)

#define SchM_Exit_FiM_FIM_EXCLUSIVE_AREA_0()
SchM_Exit_FiM(FIM_EXCLUSIVE_AREA_0)
```

Table 4-5 User Configuration File

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

The types defined by the FiM are described in this chapter.

Type Name	C-Type	Description	Value Range
FiM_FunctionIdType	uint8, uint16	Type for the FunctionID	0…255 Size depends on system complexity.
			0…65535 Size depends on system complexity.

Table 5-1 Type definitions

5.2 Services provided by FiM

5.2.1 FiM_Init()

Prototype	
`void FiM_Init ( const FiM_ConfigType* FiMConfigPtr )`	
Parameter	
`FiMConfigPtr`	pointer to a valid configuration for FiM
Return code	
`void`	N/A
Functional Description	
> sets up internal variables > the Function Inhibition Manager is not yet ready for operation afterwards (before FiM_DemInit has been called) > the FIM is set to state pre-initialized	
Particularities and Limitations	
> ServiceID = 0 > This function can be called from task context only. > This function is not reentrant. > This function is synchronous. > Precondition: Call Sequence 1: has to be called before the DEM is initialized > Precondition: Call Sequence 2: has to be called before other FiM APIs are called (except FiM_GetVersionInfo and FiM_InitMemory) > In the pre-compile variant, the ConfigPtr is unused.	

Table 5-2 FiM_Init()

5.2.2 FiM_DemInit()

Prototype	
`void FiM_DemInit ( void )`	
Parameter	
`N/A`	N/A
Return code	
`void`	N/A
Functional Description	
> this service (re-)initializes the FiM > called by DEM during initialization after DEM is prepared to receive queries for monitor and event status for initial FID calculation > called by DEM when the DEM detects a status change of a certain number of events (DEM implementation specific), e.g. clearance of event memory	
Particularities and Limitations	
> ServiceID = 3 > This function can be called from task context only. > This function is not reentrant. > This function is synchronous. > Precondition: Initialization: FiM needs to be at least pre-initialized	

Table 5-3 FiM_DemInit()

Caution
During the complete runtime of FiM_DemInit, FiM_DemTriggerOnEventStatus must not be called!

Note: Calling of FiM_DemTriggerOnMonitorStatus is ok.

Caution
The FiM loops through all configured events or FIDs and calls Dem_GetMonitorStatus and Dem_GetEventUdsStatus for each of them. Depending on the configuration, FiM_DemInit can take a considerable time.

5.2.3 FiM_DemTriggerOnMonitorStatus()

Prototype	
`void` **`FiM_DemTriggerOnMonitorStatus`** `( Dem_EventIdType EventId )`	
Parameter	
`EventId`	Identification of an Event by assigned event number. The Event Number is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: depending on configuration of Event Numbers in DEM (Max is either 255 or 65535)
Return code	
`void`	N/A
Functional Description	
> notifies the FiM that the monitor status of an event changed > called by DEM	
Particularities and Limitations	
> ServiceID = 2 > This function can be called from any context. > This function is reentrant. > This function is synchronous. > Precondition: Initialization: FiM has to be initialized	

Table 5-4 FiM_DemTriggerOnMonitorStatus()

Note
The function FiM_DemTriggerOnMonitorStatus has to look up the events that are configured in the FiM module.The search algorithm used depends on the configuration. If the event ids in FiM_EventIdTable are in continuously ascending order without gaps a direct table access is used. If they are in ascending order a binary search algorithm is used. Otherwise a linear search is applied.

5.2.4 FiM_DemTriggerOnEventStatus()

Prototype	
`void FiM_DemTriggerOnEventStatus ( Dem_EventIdType EventId, Dem_UdsStatusByteType EventStatusOld, Dem_UdsStatusByteType EventStatusNew )`	
Parameter	
`EventId`	Identification of an Event by assigned event number. The Event Number is configured in the DEM. Min.: 1 (0: Indication of no Event or Failure) Max.: depending on configuration of Event Numbers in DEM (Max is either 255 or 65535)
`EventStatusOld`	extended event status before change
`EventStatusNew`	extended event status after change
Return code	
`void`	N/A
Functional Description	
> Extension to Autosar. > Notifies the FiM that the extended status of an event changed. > Called by DEM.	
Particularities and Limitations	
> ServiceID = 0x82 > This function can be called from any context. > This function is reentrant. > This function is synchronous. > Precondition: Initialization: FiM has to be initialized > Precondition: OBD license available	

Table 5-5 FiM_DemTriggerOnEventStatus()

Caution
FiM_DemTriggerOnEventStatus may not be interrupted by a call of FiM_DemTriggerOnEventStatus for the same event id!

Interruption of FiM_DemTriggerOnEventStatus by FiM_DemTriggerOnEventStatus is only allowed if the event ids of both calls differ.

Note
The function FiM_DemTriggerOnEventStatus has to look up the events that are configured in the FiM module.The search algorithm used depends on the configuration. If the event ids in FiM_EventIdTable are in continuously ascending order without gaps a direct table access is used. If they are in ascending order a binary search algorithm is used. Otherwise a linear search is applied.

5.2.5 FiM_GetFunctionPermission()

Prototype	
`Std_ReturnType` **`FiM_GetFunctionPermission`** `( Fim_FunctionIdType FID, boolean* Permission )`	
Parameter	
`FID`	FunctionId as configured identifies a functionality Min.: 1 (0: Indication of no functionality) Max.: Result of configuration of FIDs in FIM (Max is either 255 or 65535)
`Permission`	TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
Return code	
`Std_ReturnType`	E_OK: The request is accepted E_NOT_OK: The request is not accepted, e.g. initialization of FIM not completed
Functional Description	
> Service reports the permission state of the functionality assigned to the FID. > Permission will be set to FALSE, if the FIM is not initialized or if the FID is not valid. > If development error reporting is enabled, an error will additionally be reported to the DET.	
Particularities and Limitations	
> ServiceID = 1 > This function can be called from any context. > This function is reentrant. > This function is synchronous. > Precondition: Initialization: FiM has to be initialized.	

Table 5-6 FiM_GetFunctionPermission()

Note
When no events are connected to the requested FID or all connected events are not available in DEM (signalized by DEM returning E_NOT_OK during Dem_GetMonitorStatus) FiM_GetFunctionPermission returns the FID as permitted.

Reasoning: No event inhibits the function.

5.2.6 FiM_GetFunctionPendingStatus()

Prototype	
`Std_ReturnType` **`FiM_GetFunctionPendingStatus`** `( Fim_FunctionIdType FID, boolean* pendingStatus )`	
Parameter	
`FID`	FunctionId as configured identifies a functionality Min.: 1 (0: Indication of no functionality) Max.: Result of configuration of FIDs in FIM (Max is either 255 or 65535)
`pendingStatus`	TRUE: any event connected to FID has status bit DEM_UDS_STATUS_PDTC set FALSE: no event connected to FID has status bit DEM_UDS_STATUS_PDTC set
Return code	
`Std_ReturnType`	E_OK: The request is accepted E_NOT_OK: The request is not accepted, e.g. initialization of FIM not completed
Functional Description	
> Extension to Autosar. > This function is used in context of IUMPR calculation for OBD. > Service reports the pending status of the event assigned to the FID. > Pending status will be set to FALSE, if the FIM is not initialized or if the FID is not valid. > If development error reporting is enabled, an error will additionally be reported to the DET. > See also [4] for a description of usage.	
Particularities and Limitations	
> ServiceID = 0x80 > This function can be called from any context. > This function is reentrant. > This function is synchronous. > Precondition: Initialization: FiM has to be initialized. > Precondition: OBD license available.	

Table 5-7 FiM_GetFunctionPendingStatus()

Note
When no events are connected to the requested FID or all connected events are not available in DEM (signalized by DEM returning E_NOT_OK during Dem_GetEventStatus) FiM_GetFunctionPendingStatus returns FALSE.

Reasoning: No event related to FID has a pending bit set.

5.2.7 FiM_GetVersionInfo()

Prototype	
`void FiM_GetVersionInfo ( Std_VersionInfoType* versioninfo )`	
Parameter	
`versioninfo`	pointer to where to store the version information of this module
Return code	
`void`	N/A
Functional Description	
> This service returns the version information of the FiM. > The version information contains vendor ID, moduleID, major/minor/patch version number. > The version numbers are decimal coded. > This service is only available if enabled at compile time.	
Particularities and Limitations	
> ServiceID = 4 > This function can be called from any context. > This function is reentrant. > This function is synchronous. > Precondition: Configuration: FIM_VERSION_INFO_API == STD_ON	

Table 5-8 FiM_GetVersionInfo()

5.2.8 FiM_InitMemory()

Prototype	
`void FiM_InitMemory ( void )`	
Parameter	
`N/A`	N/A
Return code	
`void`	N/A
Functional Description	
> Extension to Autosar. > Use this function to initialize static RAM variables in case the start-up code is not used to initialize RAM.	
Particularities and Limitations	
> This function can be called from any context. > This function is reentrant. > This function is synchronous.	

Table 5-9 FiM_InitMemory()

5.2.9 FiM_MainFunction()

Prototype	
`void` **`FiM_MainFunction`** `( void )`	
Parameter	
`N/A`	N/A
Return code	
`void`	N/A
Functional Description	
Main function is empty and does not need to be scheduled.	
Particularities and Limitations	
> ServiceID = 5 > This function can be called from task context only. > This function is not reentrant. > This function is synchronous. > Precondition: Initialization: FiM has to be initialized	

Table 5-10 FiM_MainFunction()

5.3 Services used by FiM

Table 5-11 lists services which are provided by other components and are used by FiM. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
Det	Det_ReportErrorStatus()
Dem	Dem_GetMonitorStatus()
Dem	Dem_GetEventUdsStatus()
EcuM	optional EcuM_BswErrorHook

Table 5-11 Services used by the FiM

5.3.1 EcuM_BswErrorHook()

Prototype	
`void` **`EcuM_BswErrorHook`** `( uint16 BswModuleId, uint8 ErrorId )`	
Parameter	
`BswModuleId`	Autosar ModuleId. The Dem will pass `FIM_MODULE_ID`.
`ErrorId`	Error code detailing the error cause, see Table 6-3
Return code	
`void`	N/A
Functional Description	
This function is called to report defunct configuration data passed to FiM_Init. The FiM will leave FiM_Init after a call to this function, without initializing. Further calls to the FiM module are not safe.	
Particularities and Limitations	
> This function is called in error cases, when initializing a Post-Build configuration fails. > It's not safe if it returns to the caller, especially if development error detection is disabled. > This function is called from FiM_Init().	

5.4 Service Ports

Note
Service ports can only be used when RTE support is configured.

5.4.1 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.

5.4.1.1 Provide Ports on FiM Side

At the Provide Ports of the FiM the API functions described in section 5.2 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 FiM 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.

5.4.1.1.1 FunctionInhibition

Operation	API Function	Port Defined Argument Values
FunctionInhibition	Fim_GetFunctionPermission	FunctionIdType FunctionId

Table 5-12 FunctionInhibition

5.4.1.2 Require Ports on FiM Side

At its Require Ports the FiM 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 FiM.

Note
Currently, the FiM does not have Require Ports.

6 Configuration

6.1 Configuration Variants

The FiM supports the configuration variants

- VARIANT-PRE-COMPILE
- VARIANT-POST-BUILD-LOADABLE

The configuration classes of the FiM parameters depend on the supported configuration variants. For their definitions please see the fim_bswmd.arxml file.

6.2 Configurable Attributes

The description of each configurable option is described within its online help in the DaVinci Configurator Pro 5 tool.

The parameters in Table 6-1 can be configured via the tool.

Configuration Parameter	Comment	Post-Build Loadable
FiMDataFixed	FALSE: no direct calibration without vPblCalib TRUE: not supported	no
FiMEventUpdate TriggeredByDem[1]	TRUE: trigger on event/monitor status change FALSE: not supported	no
FiMOptimization ForInhibitionMasks	If enabled, code generator will try to optimize usage of inhibition masks. E.g., inhibition configurations with identical event id and function id are combined to a single configuration (e.g., FIM_NOT_TESTED/ FIM_LAST_FAILED to FIM_NOT_TESTED_OR_FAILED).	yes
FiMMaxFidsPerEvent	If component vPblCalib is used this parameter determines the maximum number of inhibition configurations that can be calibrated for each DEM event. If component vPblCalib is not used this parameter is ignored.	no

Table 6-1 Configurable Parameters

The following configuration parameters are currently not supported and therefore ignored:

- FiMMaxTotalLinks
- FiMMaxSummaryLinks
- FiMMaxSummaryEvents
- FiMMaxEventsPerFid
- FiMMaxEventFidLinks

[1] Translates to FIM_CYCLIC_EVENT_EVALUATION in code with inverted meaning.

6.2.1 Inhibition Configuration Codes

The inhibition configuration consists of a table of either FID / inhibition code or EventId / inhibition code configurations. Inhibition configurations use the codes according Table 6-2.

Inhibition Configuration Code	Referenced Inhibition Configuration	Remarks
0x00	invalid	used to disable event/FID link
0x01	FIM_LAST_FAILED	DEM_MONITOR_STATUS_TF flag of DemMonitorStatus is set
0x02	FIM_NOT_TESTED	DEM_MONITOR_STATUS_TNCTOC flag of DemMonitorStatus is set
0x03	FIM_TESTED	DEM_MONITOR_STATUS_TNCTOC flag of DemMonitorStatus is not set
0x04	FIM_TESTED_AND_FAILED	DEM_MONITOR_STATUS_TF flag of DemMonitorStatus is set and DEM_MONITOR_STATUS_TNCTOC flag is not set

Table 6-2 Inhibition Configuration Codes

Note
For optimization purposes the FiM internally handles additional codes. These cannot be configured explicitly. Rather they are generated automatically.

6.3 Measurement and Calibration

6.3.1 Measurable Objects

Measurable objects are not intended to be modified as they have direct influence on FiM state machines and therefore might result in an undefined behavior. Their current value shall be read out only.

Please note that not all elements might be available – disabled features usually also disable some of the RAM tables.

The following tables describe the measurable objects:

Measureable Item	Base Type	Description
FiM_FidPendingCounter	uint16	Table with FID Pending Counters containing the number of events that currently lock an IUMPR ratio.

6.3.2 Calibration

The FiM does not support a direct calibration via a calibration tool.

Post-Build Loadable has to be used instead, see section 6.4 for details.

The component vPblCalib can be used to connect to a standard compliant calibration tool and hide the Post-Build Loadable process from the user. See [6] for details.

6.4 Post-Build support

6.4.1 Post-Build Loadable

Although calibration already is a post build method of configuration, Vector also provides a tool based approach superior to calibration. While calibration only modifies existing configuration tables, the Post-Build Loadable approach also allows to validate the configuration change preventing misconfiguration, and to use compacted table structures – with benefits to run-time and ROM usage.

Note
Adding new FIDs to an existing configuration during Post-Build is not supported. If you have ‘inactive’ functions that are enabled by calibration or other means, set up the FID for this function at pre-compile time and disable it by not configuring an event-to-FID link to it.

6.4.1.1 Initialization

During the startup of the ECU, the FiM expects to receive the pointer to its configuration data in `FiM_Init()`. Typically, this pointer is passed by the MICROSAR EcuM based on the post-build configuration. If no MICROSAR EcuM is used, the procedure of how to find the proper initialization pointer is out of scope of this document.

The FiM module will verify the received pointer for three criteria before it is accepted to initialize the module. If the initialization fails, an EcuM error hook (see chapter 5.3.1) is called with an error code according to Table 6-3.

Error Code	Reason
`ECUM_BSWERROR_NULLPTR`	Initialization with a null pointer.
`ECUM_BSWERROR_MAGICNUMBER`	Magic pattern check failed. This pattern is appended at the end of the initialization root structure. An error here is a strong indication of random data, or a major incompatibility between the code and the configuration data.
`ECUM_BSWERROR_COMPATIBILITYVERSION`	The configuration data was created by an incompatible generator. This is also tested by verification of a ‘magic’ pattern, so initialization with random data can also cause this error code.

Table 6-3 Error Codes possible during Post-Build initialization failure

If no MICROSAR EcuM is used, these error hooks and the error code constants have to be provided by the environment.

1. If the pointer equals NULL_PTR, initialization is rejected.

2. If the initialization structure does not end with the correct magic number it is rejected.
3. If the initialization structure was created by an incompatible generator version it is rejected (starting magic number check)

Caution
The verification steps performed during initialization are neither intended nor sufficient to detect corrupted configuration data. They are intended only to detect initialization with a random pointer, and to reject data created by an incompatible generator version.

6.4.1.2 Configuration of Post-Build Loadable

The configuration of post-build loadable is described in [5].

The FiM supports several configuration data elements for post-build loadable. Besides parameters marked in Table 6-1 this also includes:

- event ids known by FiM
- event-to-FID links
- inhibition configurations

7 AUTOSAR Standard Compliance

7.1 Deviations

See Table 3-2 Not supported AUTOSAR standard conform features.

7.2 Additions/ Extensions

See Table 3-3 Features provided beyond the AUTOSAR standard.

7.3 Limitations

Limitations are not known.

8 Glossary and Abbreviations

8.1 Glossary

Term	Description
DaVinci Configurator Pro 5	Configuration and generation tool for MICROSAR components

Table 8-1 Glossary

8.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AR	AUTOSAR
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
EcuM	ECU Manager
FiM	Function Inhibition Manager
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
IUMPR	In Use Monitor Performance Ratio
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
OBD	Onboard Diagnostics
PPort	Provide Port
RPort	Require Port
RTE	Runtime Environment
SchM	Schedule Manager
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector.com