

Document Title	Specification of Function Inhibition Manager
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Version	1.1.1
Document Status	Final
Part of Release	2.1
Revision	0014

Document Change History			
Date	Version	Changed by	Change Description
24.01.2007	1.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> • “Advice for users” revised • “Revision Information” added
05.12.2006	1.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Modification of the FIM data structure: Several summarized events can be assigned to the FimInhibition-Configuration • Inserted corrected sequence charts for FIM initialization phase and Fim_DemTriggerOnEventStatus • Added file MemMap.h to header file structure • Added requirement for extended header file structure (Schedule Manager) • Added SchM_FIM.h to header file structure • Legal disclaimer revised
24.04.2006	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

Disclaimer

Any use of these specifications requires membership within the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of these specifications.

Following the completion of the development of the AUTOSAR specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Copyright © 2004-2006 AUTOSAR Development Partnership. All rights reserved.

Advice to users of AUTOSAR Specification Documents:

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

1	Introduction and functional overview	6
2	Acronyms and abbreviations	7
3	Related documentation.....	8
3.1	Input documents.....	8
3.2	Related standards and norms	9
4	Constraints and assumptions	10
4.1	Limitations	10
4.2	Applicability to car domains.....	10
5	Dependencies to other modules.....	11
5.1	AUTOSAR service.....	12
5.1.1	Architecture.....	12
5.2	Requirements	12
5.2.1	Use Cases.....	12
5.3	Specification of the Ports and Port Interfaces	12
5.3.1	Description of the Interfaces.....	12
5.4	Internal behavior.....	13
5.5	Configuration of the FIDs	13
5.6	File structure	14
5.6.1	Code file structure	14
5.6.2	Header file structure.....	14
6	Requirements traceability	15
7	Functional specification	20
7.1	Background & Rationale.....	20
7.2	Requirements	20
7.2.1	FIM core variables.....	20
7.2.1.1	‘Diagnostic Event’ definition	20
7.2.1.2	‘Summarized Event’ definition	20
7.2.1.3	‘Function Identifier’ definition.....	21
7.2.1.4	‘Function Identifier permission status’ definition	22
7.2.2	FIM core functionalities	22
7.2.2.1	FIM Data Structure.....	22
7.2.2.2	Interaction between DEM and Function Inhibition Manager (FIM)	23
7.2.2.3	Interaction between SW-Components and Function Inhibition Manager (FIM)	26
7.2.2.4	Application example for FIM usage.....	26
7.2.2.5	Initialization	27
7.2.3	Auxiliary explanations and definitions	27
7.2.3.1	Output for other WPs	27
7.2.4	Version check.....	27
7.3	Error classification	27
7.4	Error detection.....	28

7.5	Error notification	28
8	API specification	29
8.1	Imported types.....	29
8.1.1	Standard types	29
8.1.2	Dem types	29
8.2	Type definitions	29
8.2.1	Fim_FunctionIdType	29
8.3	Function definitions	30
8.3.1	Interface ECU State Manager ↔ FIM	30
8.3.1.1	Fim_Init	30
8.3.1.2	Fim_Shutdown	30
8.3.2	Interface SW-Components ↔ FIM	31
8.3.2.1	Fim_GetFunctionPermission.....	31
8.3.3	Interface DEM ↔ FIM	31
8.3.3.1	Fim_DemTriggerOnEventStatus	31
8.3.3.2	Fim_DemInit.....	32
8.3.4	Fim_GetVersionInfo	33
8.4	Call-back notifications	34
8.5	Scheduled functions	34
8.5.1	Fim_MainFunction.....	34
8.6	Expected Interfaces.....	34
8.6.1	Mandatory Interfaces	34
8.6.2	Optional Interfaces	35
8.6.3	Configurable interfaces	35
9	Sequence diagrams	36
9.1	Initialization sequence of FIM.....	36
9.2	Fim_DemTriggerOnEventStatus	37
10	Configuration specification	38
10.1	How to read this chapter	38
10.1.1	Configuration and configuration parameters	38
10.2	Containers and configuration parameters	39
10.2.1	Variants.....	39
10.2.2	FimConfiguration.....	40
10.2.3	FimSummaryEventId.....	43
10.2.4	FimEventSummary.....	44
10.2.5	FimInhibitionConfiguration	45
10.3	Published Information.....	46

1 Introduction and functional overview

The Function Inhibition Manager is responsible for providing a control mechanism for software components and the functionality therein. In this context, a functionality can be built up of the contents of one, several or parts of runnable entities with the same set of permission / inhibit conditions. By means of the FIM, inhibiting these functionalities can be configured and even modified by calibration. Therefore, the adaptability of a functionality into a new system context with modified physical boundary conditions and influences is significantly enhanced.

Functionality and runnable entity are different and independent types of classifications. Runnable entities are mainly featured by their scheduling requirements. In contrast to that, functionalities are classified by their inhibit conditions. The services of the FIM focus on functionalities in SW-Cs, however, they are not limited to them. Functionalities of the BSW can also use the FIM services.

The functionalities are assigned to an identifier (FID – function identifier) along with the inhibit conditions for that particular identifier. The functionalities poll for the permission state of their respective FIDs before execution. If an inhibit condition applies for a particular identifier, the corresponding functionality shall not be executed anymore.

The FIM is closely related to the DEM, since diagnostic events and their status information are supported as inhibit conditions. Hence, functionality which needs to be stopped in case of a failure, e.g. of a certain sensor, can be represented by a particular identifier. If the failure is detected and the event is reported to the DEM, the FIM then inhibits the FID and therefore the corresponding functionality.

In order to handle the relation of functionality and linked events, the identifier and inhibit conditions of the functionality have to be introduced into the SW-C template (equivalence for BSW) and during configuration, data structures are built up to deal with the sensitiveness of the identifiers against certain events. By means of configuration, these relations become unique per FIM.

Software components can be integrated into a new environment as event relation can be configured without big effort. Furthermore, system analysis is supported when questions like “Which functionality is inhibited if a particular event is detected?” arise. The data basis of the FIM serves as documentation of the configured interactions.

In AUTOSAR the RTE deals with SW-C in terms of their interfaces and scheduling requirements. In contrast to that, the FIM deals with inhibit conditions and provides supporting mechanisms for controlling functionalities via respective identifiers (FID). Therefore, the FIM concept and RTE concept do not interfere with each other.

The basic targets of the FIM specification document are:

- Standardization of APIs
- Introduction of possible implementations
- Ability for a common approach for OEM and supplier

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
Activity state	The activity state is the status of a software component being executed. The activity state results from the permission state as a precondition and physical enable conditions, too. It is not calculated by the FIM and not available as a status variable. It can only be derived from local information within a software component. For further details, see chapter 7.2.1.4.
API	Application Programming Interface
BSW	Basic Software
DEM	Diagnostic Event Manager
ECU	Electronic Control Unit
FID	Function Identifier
FIM	Function Inhibition Manager
Functionality	<p>Functionality comprises User-visible and User-non-visible functional aspects of a system (AUTOSAR_Glossary.pdf).</p> <p>In addition to that - in the FIM context - a functionality can be built up of the contents of one, several or parts of runnable entities with the same set of permission / inhibit conditions. By means of the FIM, the inhibition of these functionalities can be configured and even modified by calibration. Each functionality is represented by a unique FunctionId. A functionality is featured by a specific set of inhibit condition in contrast to runnable entities having specific scheduling conditions.</p>
HW	Hardware
ID	Identification/Identifier
ISO	International Standardization Organization
MIL	Malfunction Indication Light
Monitoring function	<ul style="list-style-type: none"> • Part of the Software Component. • Mechanism to monitor and finally to detect a fault of a certain sensor, actuator or could be a plausibility check • Reports states about events from internal processing of a SW-C or from further processing of return values of other basic software modules. • See also AUTOSAR_SWS_DEM
NVRAM	Non volatile Memory
OBD	Onboard Diagnostics
OEM	Original Equipment Manufacturer
OS	Operating System
Permission state	The permission state contains the information whether a functionality, represented by its FID, can be executed or whether it shall not run. The state is controlled by the FIM based on reported events. For further details, see chapter 7.2.1.4.
RAM	Random Access Memory
ROM	Read-only Memory
RTE	Runtime Environment
Runnable entity	A Runnable Entity is a part of an Atomic Software-Component, which can be executed and scheduled independently from the other Runnable Entities of this Atomic Software-Component. It is described by a sequence of instructions that can be started by the RTE. Each runnable entity is associated with exactly one EntryPoint.
SW-C	Software Component
Xxx_	Placeholder for an API provider

3 Related documentation

3.1 Input documents

- [1] List of Basic Software Modules
https://svn.autosar.org/repos/10Releases/AUTOSAR_BasicSoftwareModules.pdf
- [2] Layered Software Architecture
https://svn.autosar.org/repos/10Releases/AUTOSAR_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules
https://svn.autosar.org/repos/10Releases/AUTOSAR_SRS_General.pdf
- [4] Requirements on Function Inhibition Manager
https://svn.autosar.org/repos/10Releases/AUTOSAR_SRS_FIM.pdf
- [5] Specification of ECU Configuration
https://svn.autosar.org/repos/10Releases/AUTOSAR_ECU_Configuration.pdf
- [6] Specification of Software Component Template
https://svn.autosar.org/repos/10Releases/AUTOSAR_SoftwareComponentTemplate.pdf
- [7] Specification of RTE Software
https://svn.autosar.org/repos/10Releases/AUTOSAR_SWS_RTE.pdf
- [8] Specification of the Virtual Functional Bus
https://svn.autosar.org/repos/10Releases/AUTOSAR_VirtualFunctionBus.pdf
- [9] Specification of Diagnostic Communication Manager
https://svn.autosar.org/repos/10Releases/AUTOSAR_SWS_DCM.pdf
- [10] Specification of Diagnostic Event Manager
https://svn.autosar.org/repos/10Releases/AUTOSAR_SWS_DEM.pdf
- [11] Software Component Template
https://svn.autosar.org/repos/10Releases/AUTOSAR_SoftwareComponentTemplate.pdf

3.2 Related standards and norms

- [12] IEC 7498-1 The Basic Model, IEC Norm, 1994
- [13] D1.5-General Architecture; ITEA/EAST-EEA, Version 1.0; chapter 3, page 72 et seq.
- [14] D2.1-Embedded Basic Software Structure Requirements; ITEA/EAST-EEA, Version 1.0 or higher
- [15] D2.2-Description of existing solutions; ITEA/EAST-EEA, Version 1.0 or higher.

4 Constraints and assumptions

FIM007: FID numbers shall be unique per FIM. Since communication between software components and basic software is limited to one ECU the FIM can only control FIDs being located on the same ECU. Note that RTE does currently not support communication between basic software and software components located on different ECUs.

4.1 Limitations

Timing constraints have to be considered for the whole system. Note that the process and response times strongly depend on the implementation. Hence, if there are explicit needs for faster responses of the FIM than the cycle (time slice of the task) these have to be requested specifically. Special measures have to be implemented which are not explicitly specified in this AUTOSAR document, since here, the implementation is – on purpose – not prescribed.

FIM043: The permission of a FID shall be computed by the FIM independent of the state of other FIDs. Interdependencies between FIDs are not supported by the FIM. That means an FID does not influence another FID.

4.2 Applicability to car domains

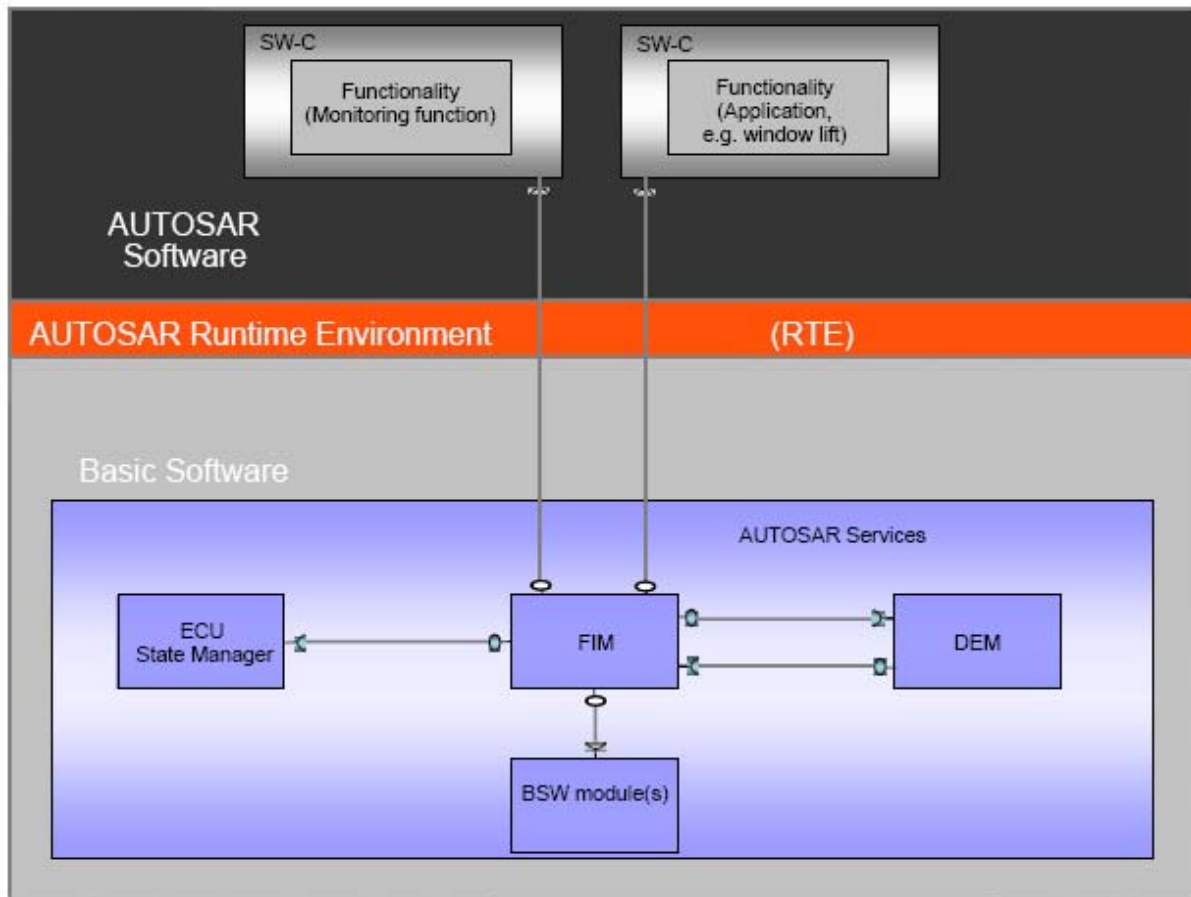
The FIM is designed to fulfill the design demands for ECUs in respect to a central handling of reactions of the system upon detected malfunctions, e.g. open circuit or shortcut. The immediate domain of applicability is therefore currently body, chassis and powertrain ECUs. However, there is no reason that the FIM cannot be used to implement ECUs for other car domains like infotainment.

One major constraint is that FIM alone will NOT be able to handle SW-Components that are:

1. time critical - might be too slow for local reconfigurations (fast backup reaction in case of e.g. invalid signals)
2. physical interactive – might not be sufficiently flexible
3. safety critical – might not have sufficient software integrity

5 Dependencies to other modules

FIM044: The AUTOSAR **Function Inhibition Manager (FIM)** has interfaces and dependencies to the following Basic software modules and Software Components:



- The **Diagnostic Event Manager (DEM)** is in charge of the handling of detected malfunctions, denoted as events and reported by monitoring functions. The DEM informs and updates the Function Inhibition Manager (FIM) upon changes of the event status in order to stop or release functionalities according to assigned dependencies.
- **SW-Components (SW-C) with FID interface** query for permission at the FIM. The existence of FIDs has to be provided by the SW components.
- **ECU State manager** is responsible for the basic initialization and de-initialization of BSW-components.
- **BSW module(s)** that are supposed to be inhibited by FIM can use the FIM interface to ask for permission. Therefore, the affected BSW modules have to provide the corresponding configuration data (EventID – FID – Inhibition mask relation) at configuration time, which comes with a template similar to the SW-component template. The interface handling for BSW modules corresponds to the one for SW-components.

5.1 AUTOSAR service

This chapter is an addition to the specification of the FIM. That specification currently defines the behavior and the C-interfaces of the corresponding basic software module. Based on this, this chapter formally specifies the corresponding AUTOSAR Service, which will be visible on the VFB.

5.1.1 Architecture

In the AUTOSAR ECU Architecture [2] the Diagnostic Event Manager [10] implements an AUTOSAR Service.

5.2 Requirements

There are three sources of requirements for this specification:

- The requirements for the functionality of the FIM service are specified in [4]. In order to model the VFB view of the Service, the chapter on AUTOSAR Services of the VFB specification [8] has to be considered as an additional requirement.
- For the formal description of the SW-C attributes [11] gives the requirements.

5.2.1 Use Cases

On each ECU we have typically one instance of the FIM Service and several Atomic Software Component instances, named “clients” further on in this document, which are using this Service. In addition, there are parts of the basic software, which either control the FIM Manager (e.g. the ECU State Manager for initiation and shutdown) or need to query the FIM for execution permission themselves.

5.3 Specification of the Ports and Port Interfaces

This chapter specifies the ports and port interfaces which are needed in order to operate the FIM functionality over the VFB.

5.3.1 Description of the Interfaces

A client can query the FIM for execution permission for a specific function. Functions are represented in the FIM by FunctionIds (FIDs). These FIDs are not used by the client SW-C. Instead, the mechanism of “port-defined argument values” is used and every FID is mapped to a separate port.

The following pseudo code defines the interfaces between the SW-C and the FIM.

```
ClientServerInterface FunctionInhibition {  
    GetFunctionPermission(OUT Boolean Permission);  
}
```

The following pseudo code shows an example definition of the FIM.

```
Service FIM {
    ProvidePort FunctionInhibition FIM1;
    ProvidePort FunctionInhibition FIM2;
    ProvidePort FunctionInhibition FIM3;
}
```

5.4 Internal behavior

Function identifiers are configured per FIM. The FIM Specification does not standardize the basic type to be used for identifying function identifiers, since the needed binary size is ECU dependent. This type has to be defined for a specific ECU as follows:

```
IntegerType FunctionIdType {
    LOWER-LIMIT = 0;
    UPPER-LIMIT = <xx>;
};
```

This type does not show up in the service ports of the client components, because the block identifier is implemented as port defined argument value, which is part of the InternalBehavior of the FIM Service. So the ECU dependency of FunctionIdType is not visible for the clients.

The InternalBehavior of the FIM Service is only seen by the local RTE. Besides the definition of the block identifiers as port defined arguments, it must specify the operation invoked runnables:

```
InternalBehavior FIM {
    // definition of associated operation-invoked RTE-events not shown
    // (it is done in the same way as for any SWC type)

    // section "runnable entities":
    RunnableEntity GetFunctionPermission
        symbol "Fim_GetFunctionPermission"
        canBeInvokedConcurrently = FALSE
}
```

5.5 Configuration of the FIDs

The FIDs of the FIM are modeled as "port defined argument values". Thus, the configuration of those values is part of the RTE configuration. Pre-compile configuration can be done by changing the XML specification for the argument values on the FIM Service. The connection between this part of the SWCT and the ECU configuration template is currently worked out and not part of this Specification. Note that the ports visible on the client side are not affected by this.

5.6 File structure

5.6.1 Code file structure

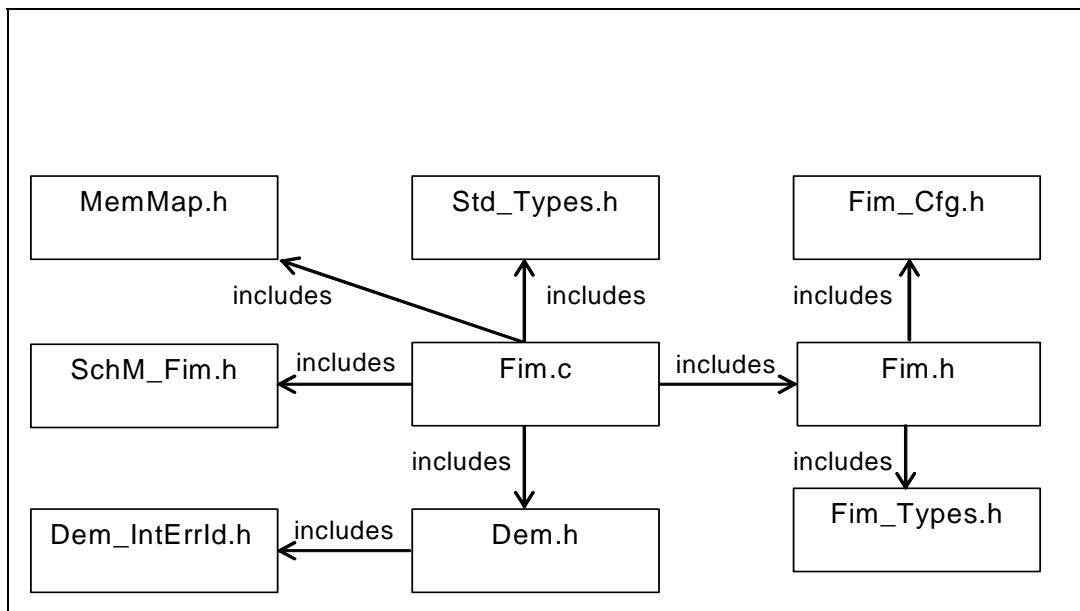
FIM030: The code file structure shall not be defined within this specification completely. At this point, it shall be pointed out that the code-file structure shall include the following file named:

- Fim_PBcfg.c – for post build time configurable parameters.

This file shall contain all post-build time configurable parameters.

5.6.2 Header file structure

FIM029:



FIM031: The module shall include the Dem.h file. By this inclusion, the APIs to report errors as well as the required EventId symbols are included. This specification defines the name of the EventId symbols, which are provided to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the EventId symbols and publishes the symbols in Dem_IntErrId.h.

6 Requirements traceability

Document: General Requirements on Basic Software Modules

Requirement	Satisfied by
[BSW00003] Version identification	FIM023
[BSW00004] Version check	FIM023
[BSW00006] Platform independency	Not applicable (Implementation requirement)
[BSW00007] HIS MISRA C	Not applicable (Implementation requirement)
[BSW00005] No hard coded horizontal interfaces within MCAL	Not applicable (requirement for μ C abstraction layer)
[BSW00009] Module User Documentation	Not applicable (Documentation requirement)
[BSW00010] Memory resource documentation	Not applicable (Documentation requirement)
[BSW00101] Initialization interface	FIM004 , FIM006
[BSW00158] Separation of configuration from implementation	FIM013
[BSW00159] Tool-based configuration	Not applicable (Requirement for tool WP)
[BSW00160] Human-readable configuration data	Not applicable (Requirement for tool WP)
[BSW00161] Microcontroller abstraction	Not applicable (requirement for μ C abstraction layer)
[BSW00162] ECU layout abstraction	Not applicable (requirement for μ C abstraction layer)
[BSW00164] Implementation of interrupt service routines	Not applicable (requirement for OS, complex drivers and μ C abstraction)
[BSW00166] BSW Module interfaces	FIM004 , FIM006 , FIM011 , FIM021
[BSW00167] Static configuration checking	Not applicable (Requirement for tool WP)
[BSW00168] Diagnostic Interface of SW components	Not applicable (requirement for DCM)
[BSW00170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (requirement for application layer)
[BSW00171] Configurability of optional functionality	FIM032 , FIM033 , FIM040
[BSW00172] Compatibility and documentation of scheduling strategy	Not applicable (Documentation requirement)
[BSW00300] Module naming convention	Implemented
[BSW00301] Limit imported information	Not applicable (Implementation requirement)
[BSW00302] Limit exported information	Not applicable (Implementation requirement)
[BSW00304] AUTOSAR integer data types	FIM027 , FIM041
[BSW00305] Self-defined data types naming convention	FIM027
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (Implementation requirement)
[BSW00307] Global variables naming convention	Not applicable (Implementation requirement)
[BSW00308] Definition of global data	Not applicable (Implementation requirement)

Requirement	Satisfied by
[BSW00309] Global data with read-only constraint	Not applicable (Implementation requirement)
[BSW00310] API naming convention	FIM004 , FIM006 , FIM011 , FIM021
[BSW00312] Shared code shall be reentrant	FIM011 , FIM021
[BSW00314] Separation of interrupt frames and service routines	Not applicable (Implementation requirement)
[BSW00318] Format of module version numbers	FIM041
[BSW00321] Enumeration of module version numbers	FIM041
[BSW00323] API parameter checking	Not applicable (Implementation requirement)
[BSW00324] Do not use HIS I/O Library	Not applicable (requirement for μ C abstraction layer)
[BSW00325] Runtime of interrupt service routines	Not applicable (Implementation requirement)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (requirement for OS and RTE)
[BSW00327] Error values naming convention	FIM047
[BSW00328] Avoid duplication of code	Not applicable (Implementation requirement)
[BSW00329] Avoidance of generic interfaces	Implemented
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (Implementation requirement)
[BSW00331] Separation of error and status values	FIM015 , FIM047
[BSW00333] Documentation of callback function context	Not applicable (Documentation requirement)
[BSW00334] Provision of XML file	Not applicable (Implementation requirement)
[BSW00335] Status values naming convention	Implemented
[BSW00336] Shutdown interface	Not applicable (No shutdown interface required for FIM)
[BSW00337] Classification of errors	FIM047 , FIM048 , FIM049
[BSW00338] Detection and Reporting of development errors	FIM051
[BSW00339] Reporting of production relevant errors and exceptions	Not applicable (No production relevant errors available)
[BSW00341] Microcontroller compatibility documentation	Not applicable (no μ C incompatibility)
[BSW00342] Usage of source code and object code	Not applicable (Implementation requirement)
[BSW00343] Specification and configuration of time	Not applicable (no timing dependency specified)
[BSW00344] Post-Build configuration	FIM013
[BSW00345] Pre-Build configuration	FIM013
[BSW00346] Basic set of module files	FIM029
[BSW00347] Naming separation of drivers	Not applicable (requirement for μ C abstraction layer)
[BSW00348] Standard type header	FIM029
[BSW00350] Development error detection keyword	FIM040
[BSW00353] Platform specific type header	Not applicable (hardware dependency)
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (Implementation requirement)
[BSW00357] Standard API return type	Not applicable (Only void functions used)

Requirement	Satisfied by
[BSW00358] Return type of init() functions	FIM004 , FIM006 , FIM045 , FIM059
[BSW00359] Return type of callback functions	Not applicable (No callback function used)
[BSW00360] Parameters of callback functions	Not applicable (No callback function used)
[BSW00361] Compiler specific language extension header	Not applicable (hardware dependency)
[BSW00369] Do not return development error codes via API	FIM051
[BSW00370] Separation of callback interface from API	Not applicable (Implementation requirement)
[BSW00371] Do not pass function pointers via API	Implemented
[BSW00373] Main processing function naming convention	FIM060
[BSW00374] Module vendor identification	FIM041
[BSW00375] Notification of wake-up reason	Not applicable (no wake-up interrupt specified)
[BSW00376] Return type and parameters of main processing functions	FIM060
[BSW00377] Module specific API return types	FIM027
[BSW00378] AUTOSAR boolean type	Not applicable (Implementation requirement)
[BSW00379] Module identification	FIM032 , FIM033 , FIM041
[BSW00380] Separate C-Files for configuration parameters	FIM030
[BSW00381] Separate configuration header file for pre-compile time parameters	FIM029
[BSW00382] Not-used configuration elements need to be listed	Not applicable (no configuration element available that is not used)
[BSW00383] List dependencies of configuration files	FIM029 , FIM031
[BSW00384] List dependencies to other modules	FIM004 , FIM044
[BSW00385] List possible error notifications	FIM051
[BSW00386] Configuration for detecting an error	Not applicable (FIM cannot detect errors)
[BSW00387] Specify the configuration class of callback function	Not applicable (No callback function used)
[BSW00388] Introduce containers	FIM037 , FIM038 , FIM039 , FIM040
[BSW00389] Containers shall have names	FIM037 , FIM038 , FIM039 , FIM040
[BSW00390] Parameter content shall be unique within the module	FIM037 , FIM038 , FIM039 , FIM040
[BSW00391] Parameter shall have unique names	FIM037 , FIM038 , FIM039 , FIM040
[BSW00392] Parameters shall have a type	FIM037 , FIM038 , FIM039 , FIM040
[BSW00393] Parameters shall have a range	FIM037 , FIM038 , FIM039 , FIM040
[BSW00394] Specify the scope of the parameters	FIM037 , FIM038 , FIM039 , FIM040
[BSW00395] List the required parameters (per parameter)	FIM037 , FIM038 , FIM039 , FIM040
[BSW00396] Configuration classes	FIM037 , FIM038 , FIM039 , FIM040
[BSW00397] Pre-compile-time parameters	FIM037 , FIM038 , FIM039 , FIM040
[BSW00398] Link-time parameters	FIM037 , FIM038 , FIM039 , FIM040
[BSW00399] Loadable Post-build time parameters	FIM037 , FIM038 , FIM039 ,

Requirement	Satisfied by
	FIM040
[BSW00400] Selectable Post-build time parameters	FIM037 , FIM038 , FIM039 , FIM040
[BSW00401] Documentation of multiple instances of configuration parameters	FIM037 , FIM039 , FIM040
[BSW00402] Published information	FIM041
[BSW00404] Reference to post build time configuration	FIM062
[BSW00405] Reference to multiple configuration sets	FIM062
[BSW00406] Check module initialization	FIM045 , FIM055 , FIM056 , FIM057 , FIM058 , FIM059
[BSW00407] Function to read out published parameters	FIM032 , FIM033
[BSW00408] Configuration parameter naming convention	FIM037 , FIM038 , FIM039 , FIM040
[BSW00409] Header files for production code error IDs	Not applicable (No production code errors)
[BSW00410] Compiler switches shall have defined values	FIM037 , FIM038 , FIM039 , FIM040
[BSW00411] Get version info keyword	FIM032 , FIM033
[BSW00412] Separate H-File for configuration parameters	FIM029
[BSW00413] Accessing instances of BSW modules	FIM041
[BSW00414] Parameter of init function	FIM004
[BSW00415] User dependent include files	FIM029
[BSW00416] Sequence of Initialization	FIM004 , FIM018
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (FIM is BSW module)
[BSW00419] Separate C-Files for pre-compile time configuration parameters	FIM030
[BSW00420] Production relevant error event rate detection	Not applicable (requirement for DEM)
[BSW00421] Reporting of production relevant error events	Not applicable (No production code errors)
[BSW00422] Debouncing of production relevant error status	Not applicable (requirement for DEM)
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (Implementation requirement)
[BSW00424] BSW main processing function task allocation	Not applicable (Implementation requirement)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (Implementation requirement)
[BSW00426] Exclusive areas in BSW modules	Not applicable (Implementation requirement)
[BSW00427] ISR description for BSW modules	Not applicable (Implementation requirement)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (Implementation requirement)
[BSW00429] Restricted BSW OS functionality access	Not applicable (Implementation requirement)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (Implementation requirement)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (no requirement for FIM)
[BSW00433] Calling of main processing functions	Not applicable (requirement for scheduler)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (requirement for scheduler)
[BSW00435] Header File Structure for the Basic Software Scheduler	FIM029
[BSW00436] Module Header File Structure for the Basic Software	FIM029

Requirement	Satisfied by
Memory Mapping	

Document: Requirements on FIM

Requirement	Satisfied by
[BSW04700] Interface for querying the FID permission status	FIM010 , FIM011
[BSW04701] Functionality supervised by the FIM	FIM002 , FIM003 , FIM007
[BSW04702] Support of inhibit options	FIM012
[BSW04706] Individual configuration of inhibit conditions of functionalities	FIM008 , FIM013 , FIM016 , FIM043
[BSW04709] Evaluation of permission state before executing functionalities	FIM011
[BSW04712] Initialization of the permission states at start up	FIM004 , FIM018
[BSW04713] Methods for the computation of permission states	FIM009 , FIM015 , FIM020
[BSW04717] Updating the permission states	FIM021 , FIM022
[BSW04719] Mechanism for summarized diagnostic event states	FIM037 , FIM061

7 Functional specification

7.1 Background & Rationale

The Function Inhibition Manager allows querying the permission / inhibition status of software components and the functionality therein. The functionalities are assigned to an identifier (FID – function identifier) along with the inhibit conditions for that particular identifier. The functionalities poll for the permission state of their FID before execution. If an inhibit condition applies for a particular identifier, the corresponding functionality is not allowed to be executed anymore. By means of the FIM, the inhibition of these functionalities can be configured and even modified by calibration. DEM events and their status information are supported as inhibit conditions.

In order to handle the relation of functionality and associated affecting events, the identifier (FID) and inhibit conditions (events) of the functionality are included in the SW component template (equivalence for BSW) and during configuration, data structures as an inhibit matrix are built up to deal with the sensitiveness of the identifiers against certain events.

7.2 Requirements

7.2.1 FIM core variables

7.2.1.1 'Diagnostic Event' definition

A 'Diagnostic Event' defines the atomic unit that is handled by the DEM. The status of a 'Diagnostic Event' represents the result of a monitoring function or the report of a Basic Software Module. Note, that reliable permission states can only be expected after complete FIM initialization, which follows the DEM initialization. Therefore, reactions on BSW malfunctions via FIM might be delayed.

See AUTOSAR_SWS_DEM document for further details [10].

7.2.1.2 'Summarized Event' definition

FIM061: The FIM configuration shall support summarizing events. A summarized event consists of several single DEM events.

During the configuration process, these single events can be combined to a summarized event ([FIM037](#)). The goal is to deal with events being associated with / represented by one particular event. This particular event could possibly be used as an inhibit condition for simplicity in the SW-C templates. The FIM shall also be able to process the inhibit condition if one of the associated events is reported. Hence, the particular event is just meant to be a representative (ref. 10.2.4). A use case for summarized events is for example to combine all faults that indicate a failed sensor.

7.2.1.3 'Function Identifier' definition

The functionalities are addressed via numbers called function identifiers (FID). A FID represents the contents of one or more or parts of runnable entities or BSW with the same set of permission / inhibition conditions.

FIM002 (see also **FIM007**): The FunctionIds shall be unique per FIM. Two distinct functionalities with different dependencies on events can never have the same FunctionId. The configuration process shall guarantee that.

FIM003: The FunctionId shall be used to directly point to the assigned functionality information (permission status etc.) The SW-C template contains the symbolic names of all FIDs ("FID_XXX") relevant for the respective SW-C. The subsequent numbering of all FIDs within a node is accomplished by configuration process.

FIM010: The flow of information starts with the API call of the DEM providing changes of the event information. This information is processed and dependencies to FIDs are evaluated. Finally, the permission state of the FIDs is accessed via API through the RTE (Figure 1).

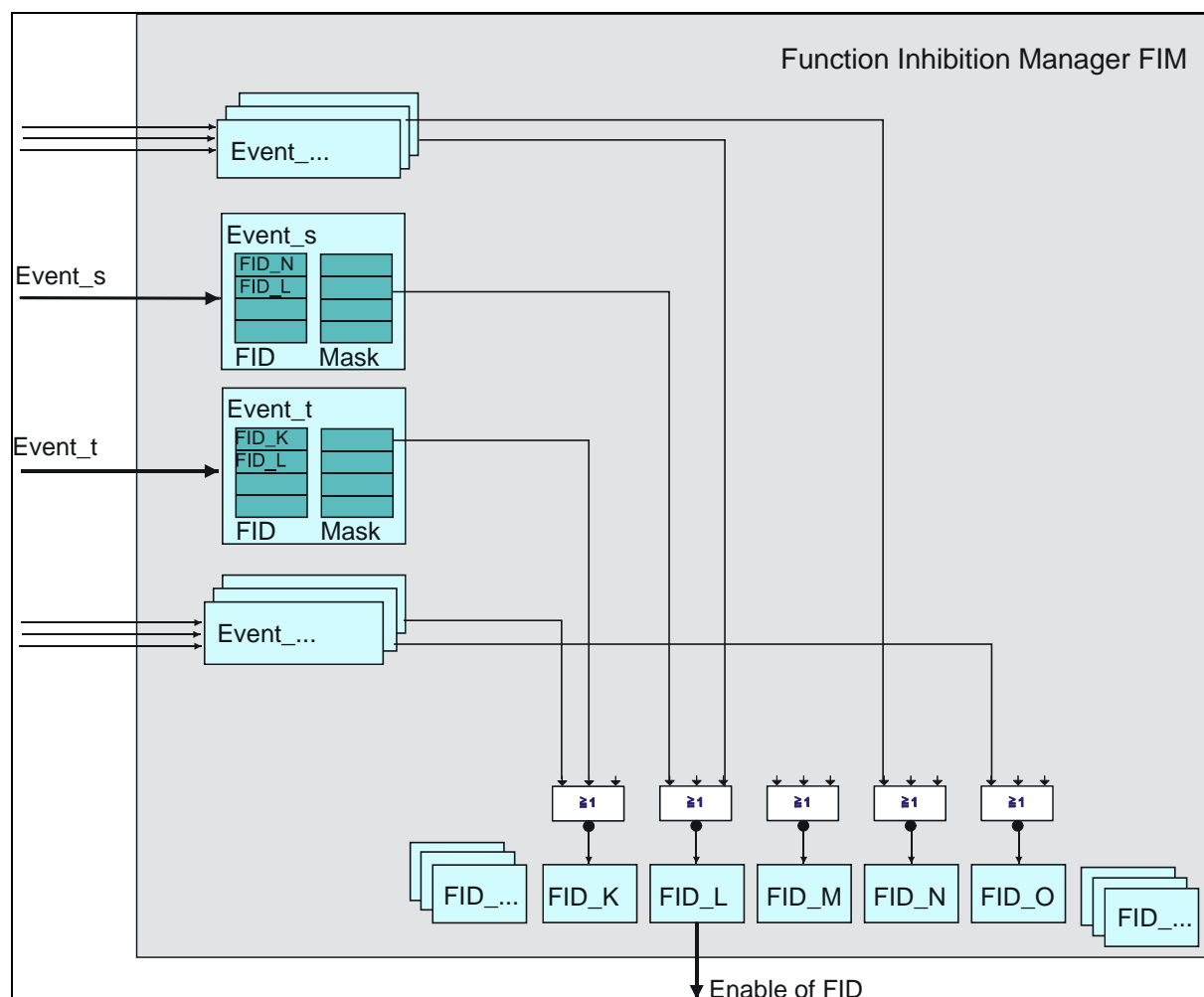


Figure 1: Logical information flow to determine FID permission states for an implementation with permission state stored in RAM

The permission state of all FIDs is calculated based on the EventIds assigned to one specific FID. Afterwards, the calculated permission states of each FID (e.g. FID_K) are “or-ed” to determine the resulting permission state.

This implies an implementation where the FIM stores the permission state of the FIDs in RAM.

Alternatively, FIM can poll the event status to re-calculate the permission state. The polling is triggered either by a functionality requesting its permission state (SW-C or BSW) or in a cyclic task. Then, there is no increased process effort within the FIM at changes of any event.

7.2.1.4 ‘Function Identifier permission status’ definition

FIM015: The permission state contains the information whether a functionality represented by its FID can be executed. This permission state is based on events reported through the DEM. Hence, there are no physical enable conditions (e.g. temperature, engine speed...) considered and the permission state = TRUE does only mean that the code is permitted to be executed but it might not be possible for other (physical) reasons. On the other hand, if permission state = FALSE, the functionality is not allowed to be executed, e.g. of a monitoring function if the related sensor is defect – even if the physical enable conditions are met.

Beside the permission state as prerequisite, the activity state includes physical enable conditions representing whether the functionality is indeed executed or not, i.e. is active or not.

As stated above, one possible implementation is to provide the permission state in status variables. An alternative is to compute the permission on the query based on the underlying dependencies.

If the permission states are stored in status variables, they are unique values per FID. SW-components access the status via `Fim_GetFunctionPermission`.

FIM009: If the implementation uses status variables for the permission of the FIDs, the status variables shall be readable for tracking purposes for the calibration system (to be defined by AUTOSAR) during test phase.

7.2.2 FIM core functionalities

7.2.2.1 FIM Data Structure

FIM013: The FIM shall build up its data structure during configuration where inhibit relations (EventID – FID – applicable mask) can be stored. A configurable number of EventIds and inhibition masks are assigned to one FID (the number of EventIds and inhibit masks per FID have to match so that you have a corresponding inhibit mask for each configured event). The inhibition mask contains the inhibition conditions for a FID provided that the associated EventIds have a certain status (`Dem_EventStatusExtendedType`). These masks define which states of an event the FID is sensitive to. However, the mask does not only address certain bits according to the `Dem_EventStatusExtendedType`, it rather selects an algorithm to calculate the boolean inhibition condition from the `Dem_EventStatusExtendedType`.

The implementation of the FIM data structure cannot be prescribed. A possible implementation of the inhibit matrix could be a block of calibration values for each inhibit source (=EventId). That means for each EventId a list of FIDs and masks is available that shall be inhibited by this EventId. A possible FIM structure consisting of such a configuration and a FID status array is shown below as an example (Figure 2).

There is a mask assigned to every FID and both are assigned to a particular event. If this event has a certain state, the inhibition of the FID becomes active if the event state matches the configured mask.

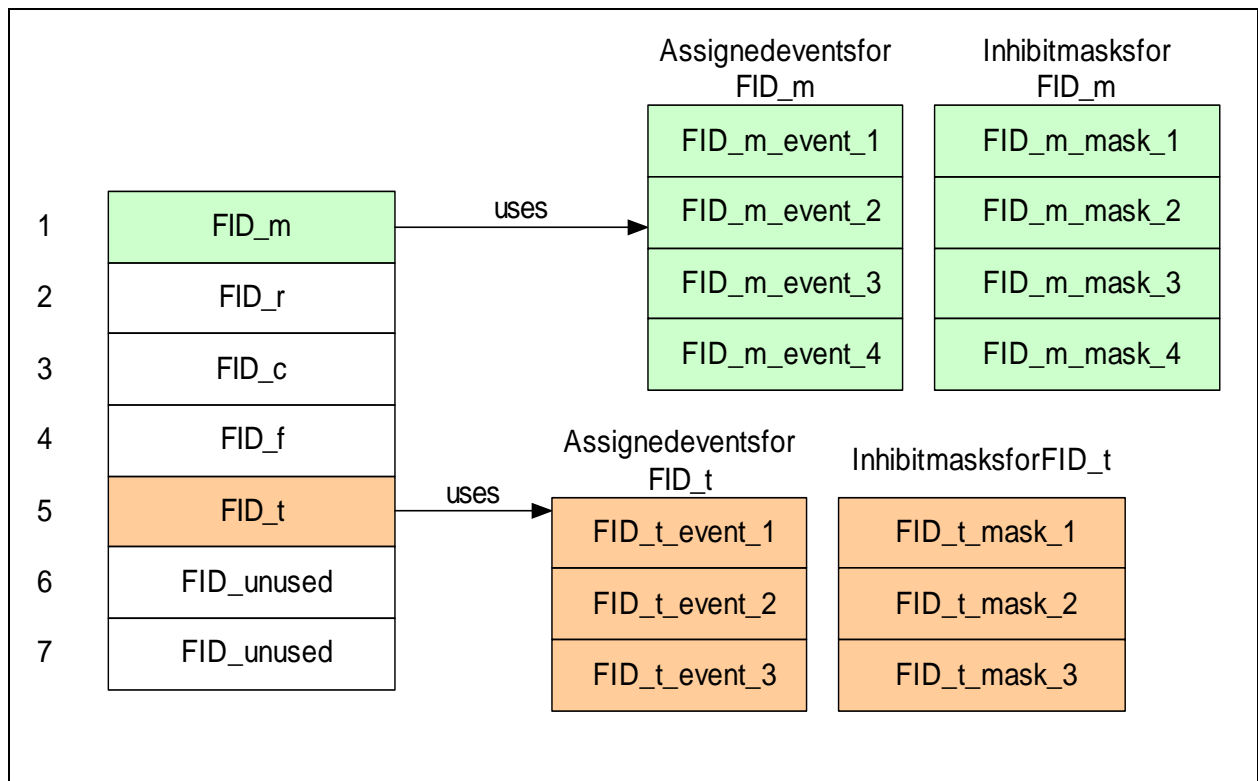


Figure 2: Inhibit mask

FIM008: It shall be possible to modify the inhibit conditions by calibration.

Depending on the implementation, it might not be possible to:

- add new events.
- extend the number of inhibited FID's per event.
- extend the specified configuration parameters concerning number of events, number of FIDs and number of links.

7.2.2.2 Interaction between DEM and Function Inhibition Manager (FIM)

FIM022: The purpose of the FIM is to provide services to control (permit / inhibit) functionality within SW-Cs based on DEM events being supported as inhibit conditions.

The Function Inhibition Manager shall use the FID – EventIDs – inhibition masks relations provided by the software components to determine the permission state for all configured FIDs.

Upon changes of a reported event status, the DEM shall inform the FIM (or other SW-C) about the new status. For this purpose, the API `Fim_DemTriggerOnEventStatus` shall be used.

Using this API-call, the inhibit links assigned to this source can be updated immediately every time an inhibit source changes its status.

Note, that this is only one possible implementation. The implementations listed below differ in the availability of FID status information in RAM and the timing of processing the reported event information:

1. Computed “on event“ (status change of an EventId) when the fault is reported (if requirements on timing are very tough). This requires storage of information about currently inhibiting events for each FID.
2. Computed in a cyclic FIM-process (FIM checks cyclically the states of the EventIds) independent of the point in time when a fault has been reported.
3. “Inhibition” is computed on event (permission state of FID is “inhibited”) while the release after ok-detection (tested and not failed) is computed cyclically. Only the status of the FID is stored.
4. Inhibit/release status is computed every time when the status of an FID is requested by a software component (functionality). No data is stored in the FIM.

As mentioned in chapter (4.1) special requirements for the implementation shall be derived from an application with particular timing requirements for this specific application.

The API `Fim_DemTriggerOnEventStatus` is only relevant if a status variable per FID is stored. In an alternative implementation when no status is stored and the permission, status is calculated every time when queried the API `Fim_DemTriggerOnEventStatus` is empty.

As an example implementation, the following figure shows the calculation of a single EventId-FID link. On the left hand side, the event status is reported by the DEM in the format of `Dem_EventStatusExtendedType`. This status is evaluated and compared to the mask related to the FID to be inhibited. The output on the right hand of the figure determines whether the inhibition counter (i.e. inhibition status) is incremented or decremented (Figure 3). An inhibition counter is assigned to each FID. The inhibition counter contains the number of currently inhibiting EventIds. Hence, a simple check for inhibition can be implemented: An FID is inhibited at a value > 0 and vice versa released at a value $= 0$. Furthermore, it is important to be sure upon a reported passed result whether any other inhibition reason applies. Such a passed result

yields to a decrementation but only if the value of the inhibition counter then reaches 0, the FID becomes released.

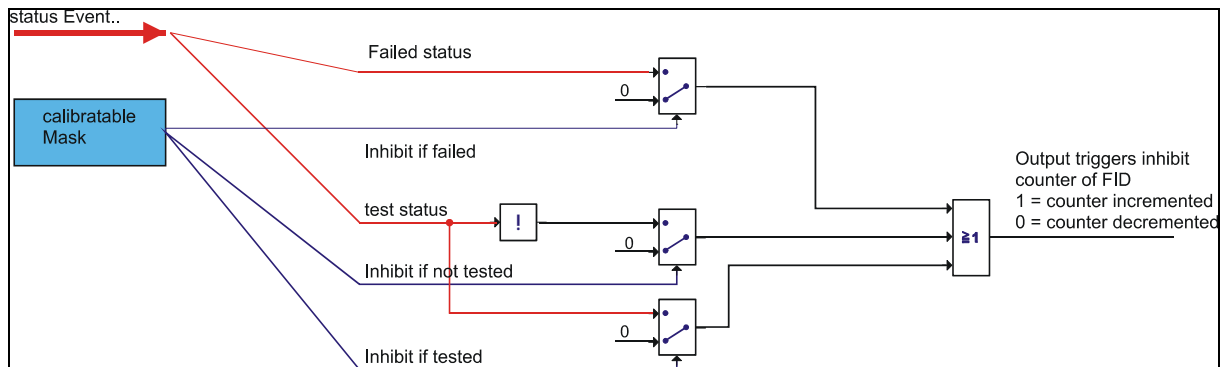


Figure 3: Calculation of permission state based on event status information

FIM012: The inhibit status is calculated based on the actual status of the inhibit source and the calibrated mask which exists for each inhibit source (ref. 10.2.5). The FID is inhibited if the Event status is equal to the calibrated mask (=Defect, Tested, NotTested), i.e. the counter is incremented. The inhibition is deactivated if the mask of the event does not match anymore the calibrated value, i.e. the counter is decremented.

Optionally, the tested status can be used for inhibiting. Depending on the inhibition condition, the inhibition can be active if the event has status "Tested" or "NotTested". If no tested value is selected, the tested status is not relevant.

The available combinations of status flags are assigned to a predefined value which has verbal representation like "Tested", "Not_Tested" or Last_Failed". As an example, the following combinations might be useful:

Limit value	Verbal representation	(Extended) Event status
0	Tested	Tested flag = TRUE, Failed flag = DONT CARE
1	Not_Tested	Tested flag = FALSE, Failed flag = DONT CARE
2	Last_Failed	Tested flag = DONT CARE, Failed flag = TRUE

Table 1: Possible limit values

According to the bit positions of the Dem_EventStatusExtendedType (cf. DEM_SWS document), the Tested flag corresponds to bit6 and the Last_Failed flag to bit0.

7.2.2.3 Interaction between SW-Components and Function Inhibition Manager (FIM)

FIM016: In order to handle the dependencies of functionalities and events in the FIM, the inhibit conditions for each FID have to be provided at best at compile time. Note, that modifications by calibrations shall be possible. The configuration mechanism of the FIM using SW-component template contents shall take these requirements into account.

First, the FID needs to be introduced and allocated. Furthermore, for each FID a list of events plus associated mask causing the inhibition of the FID shall be provided by the SW-component. In chapter 10 it is introduced, how the SW-component template considers these configuration requirements.

During the configuration process, the data structures are built up. Depending on the implementation this could, e.g. be a mapping of an event onto all affected FIDs or alternatively vice versa, a mapping of a FID onto all events affecting it.

Controlling implies that within the implemented functionality, the permission of a FID is queried via AUTOSAR service.

This structure within the code of the SW-components allows for additional commands before and after the functionality without breaking the module into fragments. Furthermore, default or limp operation can be activated if the permission is rejected.

FIM020: For an immediate control of functionality, a synchronous response is necessary – either by storing the permission state as a status variable or by evaluation of the event states upon permission query.

7.2.2.4 Application example for FIM usage

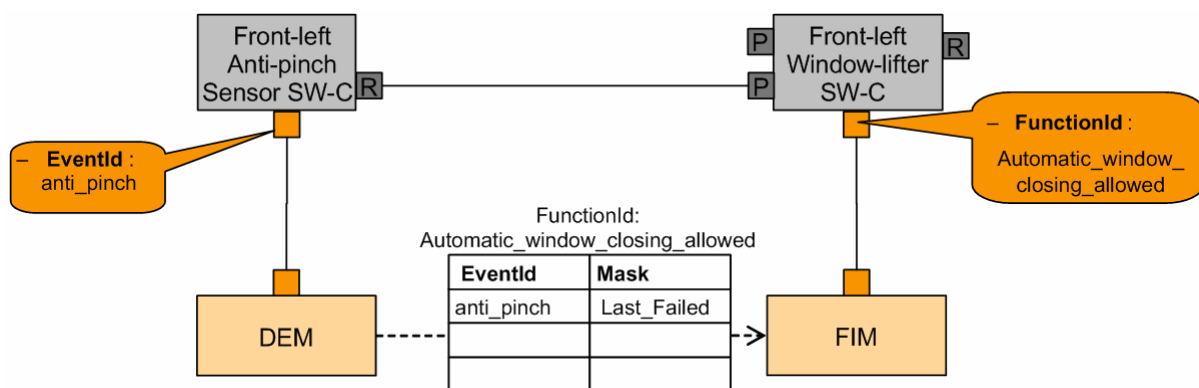


Figure 4: FIM usage

- The configuration of the FIM actually establishes the relationship between the EventId and the assigned FunctionId(s)
- The needed information is
 - For each FunctionId: How does the status of the FunctionId depend on the status of one/several EventIds

- The mask determines the relationship between the EventId status and the inhibit status of the FunctionId
- The row result is 'OR'ed to come up with the overall result for one FunctionId, if it depends on several EventIds

7.2.2.5 Initialization

FIM018: Only relevant if status information is used:

Based on all restored event status information (not only events stored in the fault memory) of the DEM, the FIM needs to compute the permission state for all FIDs at the initialization. Hence, the FIM needs to be initialized after the DEM.

In order to have a time-efficient initialization the FIM needs direct access to event information (structure) rather than using the API Dem_GetEventStatus. This is only possible if FIM and DEM are delivered as one bundle.

7.2.3 Auxiliary explanations and definitions

7.2.3.1 Output for other WPs

In order to be runtime-efficient, the event status information need to be evaluated quickly, e.g. in the Fim_Init. If DEM and FIM come as one package, the DEM-APIs with access to event status information are not necessarily used and so direct access to event status information is allowed (see AUTOSAR conformance classes).

7.2.4 Version check

FIM023: A pre-processor check in the *.c file shall ensure that the version of the *.h file is matching. This shall be done by using the parameters `FIM_SW_MAJOR_VERSION`, `FIM_SW_MINOR_VERSION` and `FIM_SW_PATCH_VERSION`.

7.3 Error classification

The FIM checks for certain faults during development and integration phase.

FIM047: Development error values are of type uint8.

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
Fim_GetPermissionStatus called before complete initialization	Development	FIM_E_WRONG_PERMISSION_REQ	0x01
DEM calls FIM before it is initialized	Development	FIM_E_WRONG_TRIGGER_ON_EVENT	0x02
Fim_GetPermissionStatus called with wrong FID	Development	FIM_E_FID_OUT_OF_RANGE	0x03
DEM calls FIM with wrong EventId	Development	FIM_E_EVENTID_OUT_OF_RANGE	0x04

7.4 Error detection

FIM048: The detection of development errors is configurable (*ON / OFF*) at pre-compile time.

The switch *FIM_DEV_ERROR_DETECT* (see chapter 10) shall activate or deactivate the detection of all development errors.

FIM049: If the *FIM_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.3 and chapter 7.3.

7.5 Error notification

FIM051: Detected development errors will be reported to the error hook of the Development Error Tracer (DET) if the pre-processor switch *FIM_DEV_ERROR_DETECT* is set (see chapter 10).

8 API specification

8.1 Imported types

8.1.1 Standard types

In this chapter, all included standard types are listed:

- Std_VersionInfoType

8.1.2 Dem types

In this chapter, all types included from module Dem are listed.

- Dem_EventStatusExtendedType
- Dem_EventIdType

8.2 Type definitions

8.2.1 Fim_FunctionIdType

FIM027:

Type:	unit8/uint16	
Range:	0...255 0...65535	Identifier of functionality Configurable, size depends on System complexity. Remark: Not all numbers are valid. The FIM data generation tool shall only assign valid values.
Description:	Identification of functionality by assigned FunctionId. The FID is configured in the FIM. Example: FID_x (=1) refers to functionality x, FID_y (=2) refers to functionality y...	

8.3 Function definitions

This is a list of functions provided for upper layer modules.

8.3.1 Interface ECU State Manager ↔ FIM

8.3.1.1 Fim_Init

Service name:	Fim_Init
Syntax:	void Fim_Init(void)
Service ID [hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non-Reentrant
Description:	<p>FIM004: This function shall be used to initialize the permission states of the FIDs based on the event data of the DEM. This function shall be called by the ECU State Manager during the startup phase of the ECU. The FIM calculates the permission states based on event status information stored in the DEM. So, the DEM must be initialized before the FIM. SW-Components controlled by FID interface are initialized afterwards.</p> <p>In order to restore the permission states quickly, it is recommended that the DEM provides direct access to event status information if DEM and FIM come as a bundle.</p> <p>FIM045: FIM shall report an error to the DET if the initialization has not been completed successfully and not permitted access has been detected.</p> <p>FIM059: A static status variable denoting if the FIM is initialized shall be initialized with value 0 before any APIs of the FIM is called. Fim_Init shall set the static status variable to a value not equal to 0.</p> <p>In case DEM and FIM are delivered as one bundle, the FIM needs to have knowledge about the data structure of the DEM so that it can directly access EventId states. If DEM and FIM are delivered as two separate modules the access to EventId states has to be performed via API call (Dem_GetEventStatus).</p>
Caveats:	The FIM is not functional until this function has been called. The DEM has to be initialized before the FIM.
Configuration:	--

8.3.1.2 Fim_Shutdown

There is no explicit action during shutdown. The permission states remain valid until the ECU is shut down, since they directly depend on the event status information.

8.3.2 Interface SW-Components ⇔ FIM

8.3.2.1 Fim_GetFunctionPermission

FIM011:

Service name:	Fim_GetFunctionPermission	
Syntax:	<pre>void Fim_GetFunctionPermission (Fim_FunctionIdType FID, boolean* Permission)</pre>	
Service ID [hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FID	Identification of a functionality by assigned FID. The FunctionId is configured in the FIM. Min.: 1 (0: Indication of no functionality) Max.: Result of configuration of FIDs in FIM (Max is either 255 or 65535)
Parameters (out):	Permission	TRUE: FID has permission to run FALSE: FID has no permission to run, i.e. shall not be executed
Return value:	None	
Description:	Service for reporting the permission state to the functionality. This service shall be used to query for the permission to run. API is called from the SW component or BSW represented by the respective FID. Fim025: Note that this return value shall be delivered synchronously because the information is directly used for controlling and executing the underlying code in the software component. FIM055: During development phase, a plausibility check of the FID range shall be supported. If an FID is out of range the DET shall be notified and no permission (FALSE) shall be returned. FIM056: During development phase the API call shall check for complete initialization of the FIM. Permission requests before initialization complete shall be notified as an error to the DET and no permission (FALSE) shall be returned.	
Caveats:	FIM configuration during integration of functionality is system specific. The API relies on proper configuration that all incoming FIDs are in a plausible range.	
Configuration:	FID	

8.3.3 Interface DEM ⇔ FIM

8.3.3.1 Fim_DemTriggerOnEventStatus

FIM021:

Service name:	Fim_DemTriggerOnEventStatus	
Syntax:	<pre>void Fim_DemTriggerOnEventStatus (Dem_EventIdType EventId, Dem_EventStatusExtendedType EventStatusOld,</pre>	

	Dem_EventStatusExtendedType EventStatusNew)						
Service ID [hex]:	0x02						
Sync/Async:	Synchronous						
Reentrancy:	Reentrant						
Parameters (in):	<table border="1"> <tr> <td>EventId</td> <td>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.: Result of configuration of Event Numbers in DEM (Max is either 255 or 65535)</td> </tr> <tr> <td>EventStatusOld</td> <td>Extended event status before change</td> </tr> <tr> <td>EventStatusNew</td> <td>Detected / reported of event status</td> </tr> </table>	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.: Result of configuration of Event Numbers in DEM (Max is either 255 or 65535)	EventStatusOld	Extended event status before change	EventStatusNew	Detected / reported of event status
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.: Result of configuration of Event Numbers in DEM (Max is either 255 or 65535)						
EventStatusOld	Extended event status before change						
EventStatusNew	Detected / reported of event status						
Parameters (out):	None						
Return value:	None						
Description:	Method to be provided to DEM in order to call FIM upon status changes. FIM057: During development phase, a plausibility check of the EventId range shall be supported. If an EventId is out of range, the DET shall be notified. FIM058: During development phase, the API call shall check for complete initialization of the FIM. Calls before initialization complete shall be notified as an error to the DET.						
Caveats:	Service for requesting the permission state by the functionality. API is called from the SW module represented by the respective FID.						
Configuration:	--						

This function is to be used by the DEM if changes are processed on event. If the calculations are done in a cyclic process by the FIM, it has to query all event status information from the DEM. Then the API above does not have to be called by the DEM. The FIM will evaluate the differences between two cycles and calculate the corresponding FID states. If the permission information is stored in status variables per FID, always the updated event status are used to determine the permission state.

If there are no permission state variables, the permission query needs access to the event status information.

8.3.3.2 Fim_DemInit

FIM006:

Service name:	Fim_DemInit
Syntax:	void Fim_DemInit(void)
Service ID [hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non-reentrant
Parameters (in):	None
Parameters	None

(out):	
Return value:	None
Description:	Fim_DemInit shall be called by DEM to re-initialize the FIM in case the DEM detects a status change of a certain number of events (DEM implementation specific), e.g. clearance of event memory in the DEM (on service 04/ISO15031-5 request). The FIM has then to re-compute the permission state for all FIDs. In case DEM and FIM are delivered as one bundle, the FIM needs to have knowledge about the data structure of the DEM so that it can directly access EventId states. If DEM and FIM are delivered as two separate modules the access to EventId states has to be performed via API call (Dem_GetEventStatus).
Caveats:	--
Configuration:	--

8.3.4 Fim_GetVersionInfo

Service name:	Fim_GetVersionInfo	
Syntax:	<pre>void Fim_GetVersionInfo (Std_VersionInfoType *versioninfo)</pre>	
Service ID [hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	none	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	none	
Description:	<p>FIM032: This service returns the version information of this module. The version information includes:</p> <ul style="list-style-type: none"> - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). <p>FIM033: This function shall be pre compile time configurable On/Off by the configuration parameter: FIM_VERSION_INFO_API</p> <p>Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.</p>	
Caveats:	--	
Configuration:	FIM_VERSION_INFO_API	

8.4 Call-back notifications

No callback specified

8.5 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non-reentrant.

8.5.1 Fim_MainFunction

FIM060:

Service name:	Fim_MainFunction
Syntax:	void Fim_MainFunction(void)
Service ID [hex]:	0x05
Description:	The evaluation of permission states can be performed either on event change or cyclically. In case of cyclic evaluation, the Fim_MainFunction has to be implemented.
Timing:	Variable cyclic
Pre condition:	--
Configuration:	Reference to chapter 10

Terms and definitions:

Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring that fixed timing (e.g. filters).

Variable cyclic: Variable cyclic means that the cycle times are defined at configuration, but might be mode dependent and therefore vary during runtime.

On pre condition: On pre condition means that no cycle time can be defined. The function will be called when conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

8.6 Expected Interfaces

In this chapter, all interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

This chapter defines all interfaces, which are required to fulfill the core functionality of the module.

API function	Module	Description
Dem_GetEventStatus	Dem	Reading status of a specific event

SchM_ActMainFunction_Fim	SchM	The FIM module implementation invokes the SchM_ActMainFunction API to trigger the activation of a corresponding main processing function.
SchM_CancelMainFunction_Fim	SchM	The FIM module implementation invokes the SchM_CancelMainFunction API to trigger the cancellation of a corresponding main processing function.

8.6.2 Optional Interfaces

This chapter defines all interfaces, which are required to fulfill an optional functionality of the module.

API function	Module	Description	Configuration parameter (description see chapter 10)
Det_ReportError	DET	Development error notification	DEM_DEV_ERROR_DETECT

8.6.3 Configurable interfaces

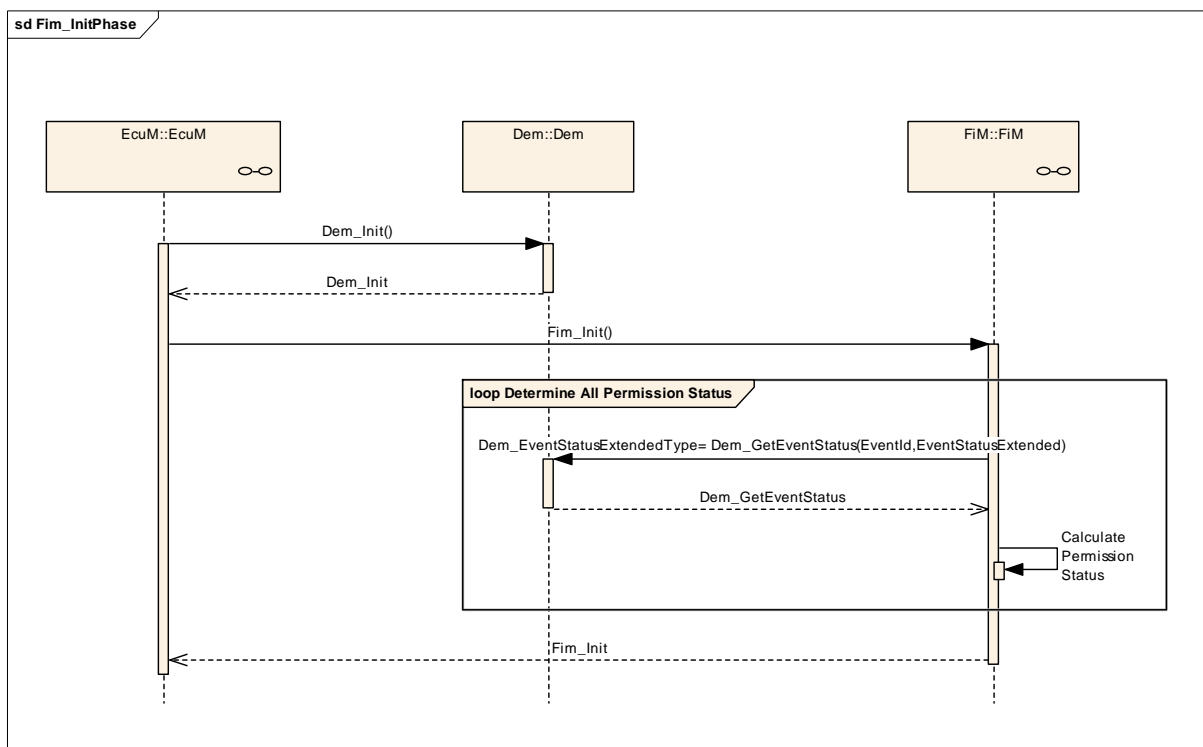
In this chapter, all interfaces are listed where the target function could be configured. The target function is usually a call-back function. The names of these kinds of interfaces are not fixed because they are configurable.

No configurable interface

9 Sequence diagrams

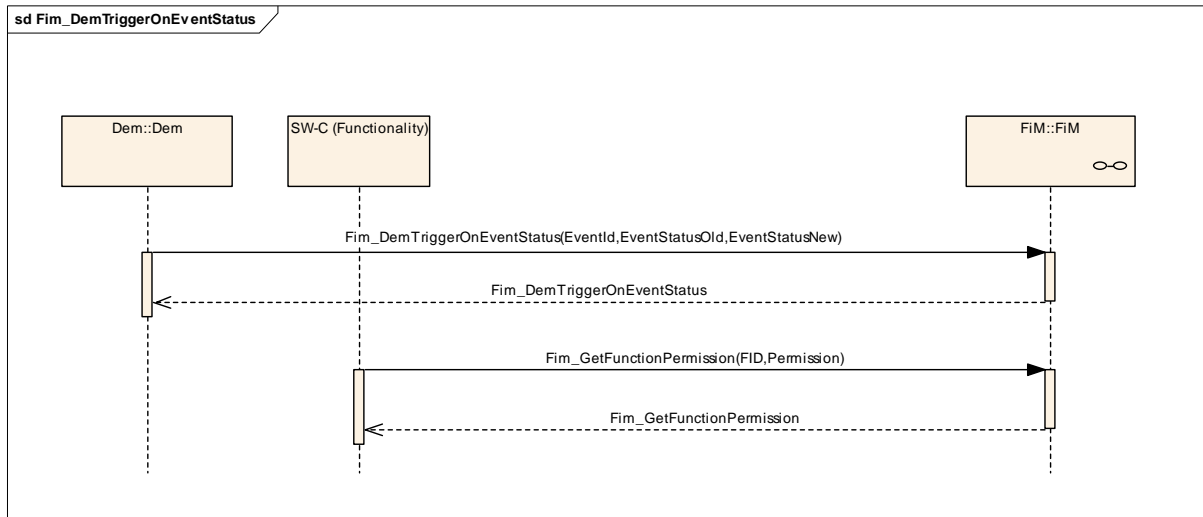
9.1 Initialization sequence of FIM

Dem_Init is called by the ECU State Manager as soon as the NVRAM Manager has copied the error memory relevant data block(s) from NVRAM to RAM ([10], DEM065). After the initialization of the DEM the ECU State Manager is able to call Fim_Init and the FIM is able to calculate the permission states of all FIDs based on the event status information afterwards. The FIM loops over all configured FIDs and reads the event status information by calling Dem_GetEventStatus. If the copying process of NVRAM data was not successful the ECU State Manager shall not call Dem_Init and Fim_Init.



9.2 Fim_DemTriggerOnEventStatus

The sequence diagram below illustrates how the DEM informs the FIM about the change of a certain event status by calling `Fim_DemTriggerOnEventStatus`. Furthermore, it indicates how the FID is affected by requesting permission status using `Fim_GetFunctionPermission`.



10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module FIM.

Chapter 10.3 specifies published information of the module FIM.

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [5]
This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration meta-model in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 7.3.

FIM062:

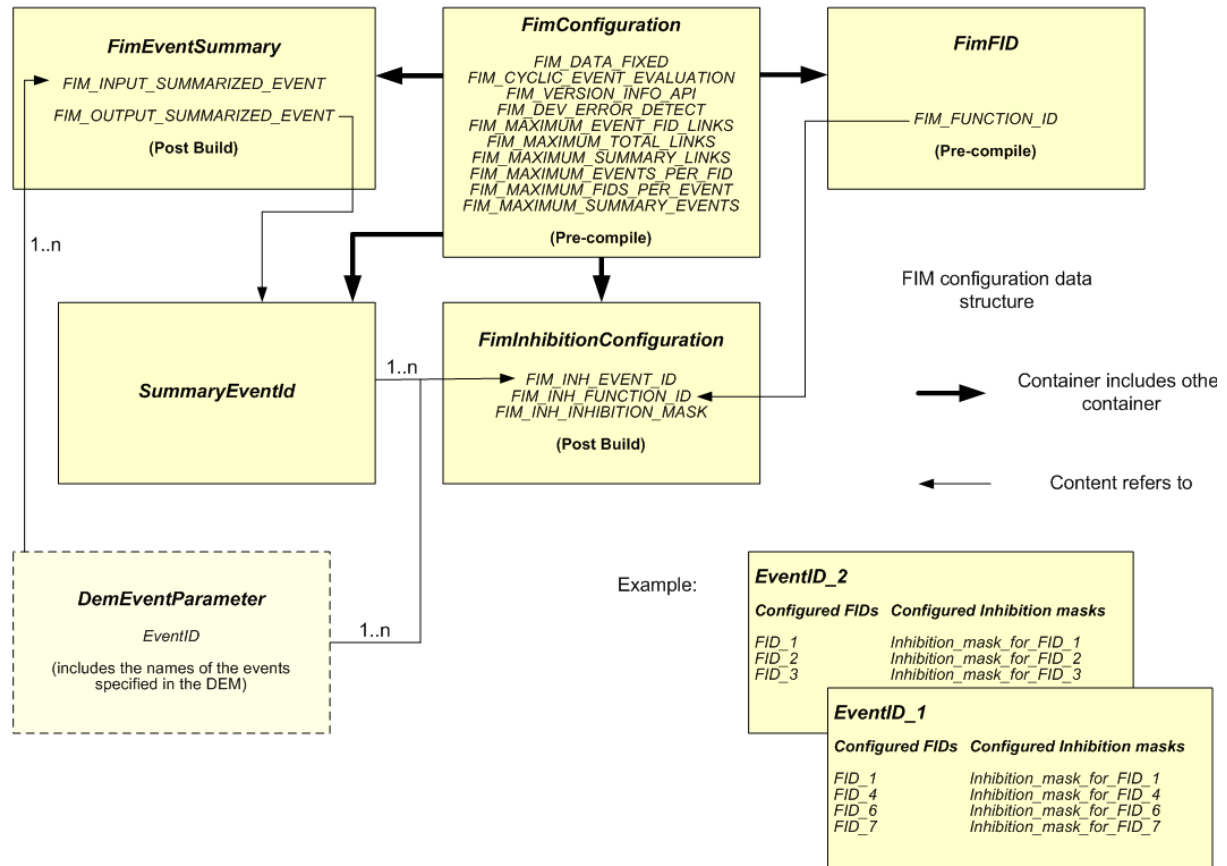


Figure 5: FIM configuration

10.2.1 Variants

Variant 1: This variant is limited to pre-compile configuration parameters only.

Variant 2: This variant allows a mix of pre-compile time- and post build time-configuration parameters.

However, there is also another meaning of variants namely variant coding and applicability of the same software/ECU to different applications/vehicles. This is not explicitly supported due to high resource usage. The inhibit conditions provided by the software components can be considered as a superset of all inhibit conditions for all variants. Based on that, the inhibit configuration has to be derived for all events and FIDs in one project. If an EventId or FID is not supported in a certain variant, the link between them has no effect. The requirements BSW00404 and BSW00405 are therefore not applicable.

10.2.2 FimConfiguration

SWS Item	FIM040
Container Name	FimConfiguration
Description	--
Configuration Parameters	

Name	FIM_DATA_FIXED		
Description	Enable or disable calibration of inhibit relations The scope of the parameter is to meet the requirement (FIM008) to have the option to calibrate inhibit data on the one hand side and also to provide the option to protect inhibit data for consistency reasons.		
Type or Unit	#define		
Range	STD_ON	Calibration disabled	
	STD_OFF	Calibration enabled	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_CYCLIC_EVENT_EVALUATION		
Description	This configuration parameter specifies whether the evaluation of DEM events is performed by the FIM cyclically or the DEM informs FIM about changes of event states.		
Type or Unit	#define		
Range	STD_ON	FIM cyclically evaluates event states in the DEM	
	STD_OFF	DEM informs FIM about changes of event states	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_VERSION_INFO_API		
Description	This configuration parameter is used to switch on or to switch off the API to get the version information.		
Type or Unit	#define		
Range	STD_ON	Version info API is supported	
	STD_OFF	Version info API is not supported	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_DEV_ERROR_DETECT		
Description	This configuration parameter is used to switch on or to switch off the detection of development errors during development.		
Type or Unit	#define		

Range	STD_ON	Detection of development errors is supported	
	STD_OFF	Detection of development errors is not supported	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_MAXIMUM_EVENT_FID_LINKS		
Description	This configuration parameter specifies the total maximum number of links between EventIds and FIDs.		
Type or Unit	#define		
Range	1...	--	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_MAXIMUM_TOTAL_LINKS		
Description	This configuration parameter specifies the total maximum number of links between EventIds and FIDs plus the number of links between EventIds and summarized events.		
Type or Unit	#define		
Range	1...	--	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_MAXIMUM_SUMMARY_LINKS		
Description	This configuration parameter specifies the total maximum number of links between EventIds and summarized events.		
Type or Unit	#define		
Range	1...	--	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_MAXIMUM_EVENTS_PER_FID		
Description	This configuration parameter specifies the maximum number of EventIds that can be linked to a single FID.		
Type or Unit	#define		
Range	1...	--	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_MAXIMUM_FIDS_PER_EVENT		
Description	This configuration parameter specifies the maximum number of FIDs that can be linked to a single event.		
Type or Unit	#define		
Range	1...	--	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Name	FIM_MAXIMUM_SUMMARY_EVENTS		
Description	This configuration parameter specifies the maximum number of summarized events that can be configured.		
Type or Unit	#define		
Range	0 ...	--	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	Local		
Dependency	--		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
FimFID	0...*	This container includes symbolic names of all FIDs.
FimSummaryEventId	0...*	This container defines the existence of summary events.
FimEventSummary	0...*	This container includes all configuration parameters regarding summarized events (the relationship between DemEvents and SummaryEvents).
FimInhibitionConfiguration	0...*	This container includes all configuration parameters concerning the relationship between event and FID.

SWS Item	FIM039
Container Name	FimFID
Description	--
Configuration Parameters	

Name	FIM_FUNCTION_ID		
Description	The configuration parameter is used as an ID which represents a functionality. FIM_FUNCTION_ID is the unique identifier assigned during FIM configuration.		
Type or Unit	Fim_FunctionIdType		
Range	Symbolic name of an FID used by FIM.	Name is resolved by FIM configuration tool by means of numbering included in Fim.h.	
Configuration Class	Pre-compile	X	All variants
	Link time	--	--
	Post Build	--	--
Scope	ECU		
Dependency	--		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
--	--	--

SWS Item	FIM037
Container Name	FimSummaryEventId
Description	The summarized EventId definition record defines the existence of a summarized event with a specific name. This summarized event can be referenced in the EventSummary (as FIM_OUTPUT_SUMMARIZED_EVENT) and Inhibition configuration (as FIM_INH_EVENT_ID).
Configuration Parameters	

10.2.3 FimSummaryEventId

Name	FIM_SUMMARIZED_EVENT		
Description	This container defines the name of a summarized event.		
Type or Unit	Dem_EventIdType		
Range	Symbolic name (must not use existing DemEventId)		
Configuration Class	Pre-compile	X	Variant 1
	Link time	--	--
	Post Build	L	Variant 2
Scope	Local		
Dependency	DEM configuration		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
--	--	--

10.2.4 FimEventSummary

SWS Item	FIM037
Container Name	FimEventSummary
Description	<p>The summarized EventId definition record consists of a summarized event ID and a specific EventId. This record means that a particular FID that has to be disabled in case of summarized event (defined above) is to be disabled in any of the specific events.</p> <p>A possible solution could be assigning events as summarized events along with a list of specific events. During the configuration process, the summarized event substitutes the referenced single events. However, it is not outlined how this requirement is solved – whether by configuration process or by implementation within the FIM. The FIM configuration tool could also build up a suitable data structure for summarized events and deal with it in the FIM implementation.</p>
Configuration Parameters	

Name	FIM_INPUT_SUMMARIZED_EVENT		
Description	The configuration parameter is used as an input for a summarized event.		
Type or Unit	Dem_EventIdType		
Range	Symbolic name of an existing DEM event (assigned with an EventId by the DEM) or summarized event	Name is resolved by FIM configuration tool by means of numbering included in Dem.h.	
Configuration Class	Pre-compile	X	Variant 1
	Link time	--	--
	Post Build	L	Variant 2
Scope	Local		
Dependency	DEM configuration		

Name	FIM_OUTPUT_SUMMARIZED_EVENT		
Description	The configuration parameter is used as an output for a summarized event.		
Type or Unit	Dem_EventIdType		
Range	Symbolic name of summarized event used by FIM	Depending on the implementation, this might only be a virtual event of the FIM or the FIM configuration tool might generate a real instance of DEM event.	
Configuration Class	Pre-compile	X	Variant 1
	Link time	--	--
	Post Build	L	Variant 2
Scope	Local		
Dependency	--		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
--	--	--
--	--	--

10.2.5 FimInhibitionConfiguration

SWS Item	FIM038
Container Name	FimInhibitionConfiguration
Description	--
Configuration Parameters	

Name	FIM_INH_EVENT_ID		
Description	The configuration parameter is used for an existing DEM event and summarized events as well.		
Type or Unit	Dem_EventIdType		
Range	Symbolic name of an existing DEM event (assigned with an EventId by the DEM) or summarized event	Name is resolved by FIM configuration tool by means of numbering included in Dem.h or implementation specific for summarized events.	
Configuration Class	Pre-compile	X	Variant 1
	Link time	--	--
	Post Build	L	Variant 2
Scope	Local		
Dependency	DEM configuration		

Name	FIM_INH_FUNCTION_ID		
Description	The configuration parameter is used as an ID which represents a functionality. FIM_INH_FUNCTION_ID is the FID to be considered in a particular inhibit relation EventId-FID-Mask.		
Type or Unit	Fim_FunctionIdType		
Range	Symbolic name of an FID used by FIM.	Name is resolved by FIM configuration tool by means of numbering included in Fim.h.	
Configuration Class	Pre-compile	X	Variant 1
	Link time	--	--
	Post Build	L	Variant 2
Scope	Local		
Dependency	--		

Name	FIM_INH_INHIBITION_MASK		
Description	The configuration parameter is used to specify the inhibition mask for an event – FID relation.		
Type or Unit	Implementation specific		
Range	Symbolic name of an inhibition mask used by FIM.	Name is resolved by FIM configuration tool by means of numbering included in Fim.h.	
	FIM_LAST_FAILED	Last Failed – Use case: Re-configuration, avoiding follow-up errors	
	FIM_TESTED	Tested – Use case: Self-deactivation, check during driving cycle	
	FIM_NOT_TESTED	Not Tested this cycle – Use case: Scheduling of monitors	
	FIM_TESTED_AND_FAILED	Tested and Failed – Use case: Avoiding deadlocks, repeated monitoring	
Configuration Class	Pre-compile	X	Variant 1
	Link time	--	--
	Post Build	L	Variant 2
Scope	Local		

Dependency	--
-------------------	----

Included Containers		
Container Name	Multiplicity	Scope / Dependency
--	--	--

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

SWS Item	FIM041	
Information elements		
Information element name	Type / Range	Information element description
FIM_VENDOR_ID	#define/ uint16	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
FIM_MODULE_ID	#define/ 0x0B	Module ID of this module from Module List
FIM_AR_MAJOR_VERSION	#define/ uint8	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
FIM_AR_MINOR_VERSION	#define/ uint8	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
FIM_AR_PATCH_VERSION	#define/ uint8	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
FIM_SW_MAJOR_VERSION	#define/ uint8	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
FIM_SW_MINOR_VERSION	#define/ uint8	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
FIM_SW_PATCH_VERSION	#define/ uint8	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.