Document Title	Specification of Function Inhibition Manager
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	082
Document Classification	Standard

Document Version	1.2.0
Document Status	Final
Part of Release	3.0
Revision	0001

Document Change History

Date	Version	Changed by	Change Description
18.12.2007	1.2.0	AUTOSAR Administration	• Error classification extended to report invocation with NULL pointer • Corrected InternalBehavior FIM to fit to API's reentrant behavior • Minimum value of parameter FimMaxSummaryLinks fixed • Document meta information extended • Small layout adaptations made
24.01.2007	1.1.1	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
05.12.2006	1.1.0	AUTOSAR Administration	• 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

Table of Contents

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 (→ deactivation of application function) these functionalities can be configured and even modified during runtime (post-built configuration).

Functionality and runnable entity are different and independent types of classifications. Runnable entities are mainly characterized 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 comes true 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 been 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

Software components can be integrated into a new environment as a collection of events which can be configured without big effort. Furthermore, system analysis is supported when questions as, for example, “Which functionality is inhibited if a particular event is detected?” arise. The data basis of the FIM serves as documentation of the configured relations between events and the SW-C to be inhibited.

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 implementation approaches
- Provide the ability for a common approach of 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 condition, 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	Functionality comprises User-visible and User-non-visible functional aspects of a system (AUTOSAR_Glossary.pdf). 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 characterized by a specific set of inhibit condition in contrast to runnable entities having specific scheduling conditions.
HW	Hardware
ID	Identification/Identifier
ISO	International Standardization Organization
MIL	Malfunction Indication Light
Monitoring function	• 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:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[4] Requirements on Function Inhibition Manager
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_FIM.pdf

[5] Specification of ECU Configuration
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_ECU_Configuration.pdf

[6] Specification of Software Component Template
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate.pdf

[7] Specification of RTE Software
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_RTE.pdf

[8] Specification of the Virtual Functional Bus
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_VirtualFunctionBus.pdf

[9] Specification of Diagnostic Communication Manager
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_DCM.pdf

[10] Specification of Diagnostic Event Manager
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_DEM.pdf

[11] Software Component Template
https:/svn2.autosar.org/repos2/22_Releases
AUTOSAR_SoftwareComponentTemplate.pdf

[12] AUTOSAR Basic Software Module Description Template,

https://svn2.autosar.org/repos2/22_Releases/
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

[13] IEC 7498-1 The Basic Model, IEC Norm, 1994

[14] D1.5-General Architecture; ITEA/EAST-EEA, Version 1.0; chapter 3, page 72 et seq.

[15] D2.1-Embedded Basic Software Structure Requirements; ITEA/EAST-EEA, Version 1.0 or higher

[16] 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 the RTE does currently not support communication between basic software and software components located on different ECUs.

4.1 Limitations

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

FIM043: The FIM shall compute the permission of a FID independently 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

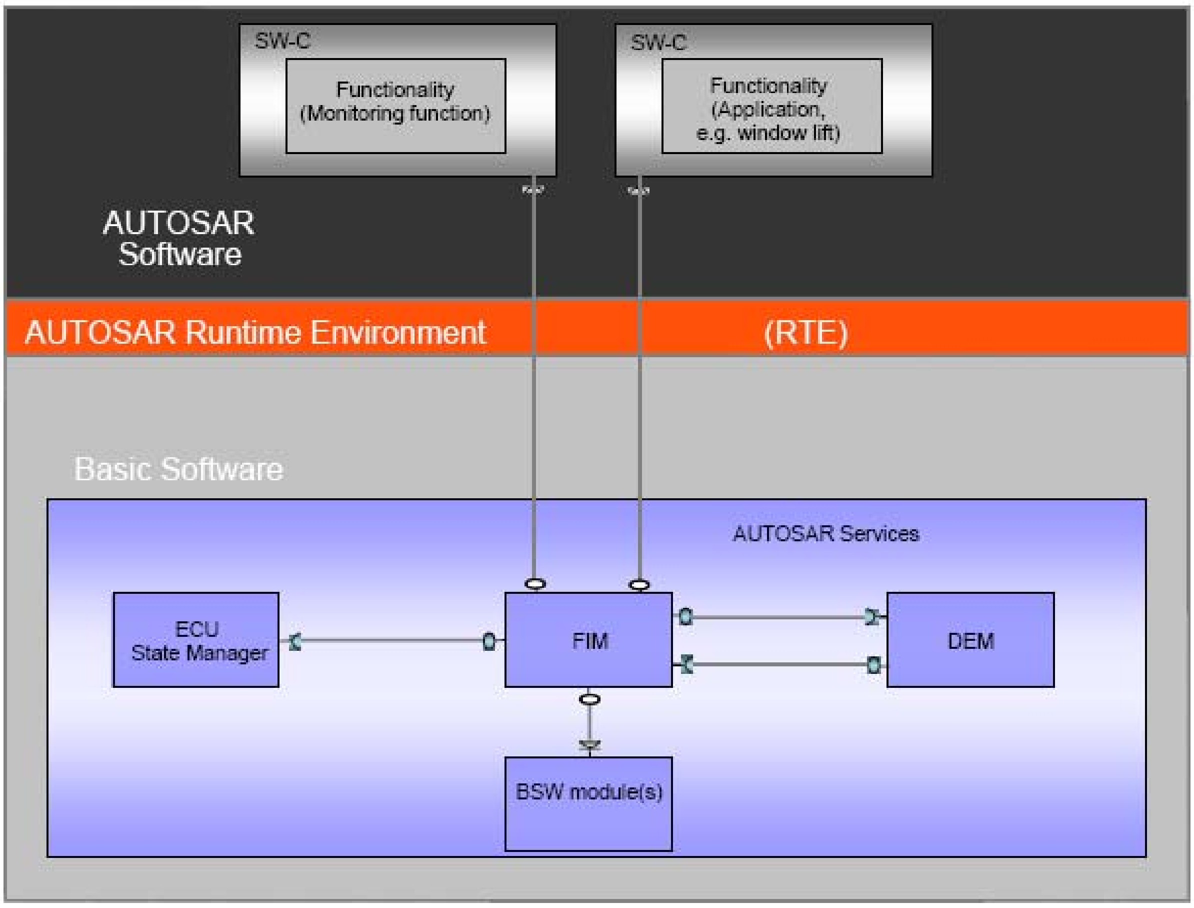
The FIM is designed to fulfill the design demands for ECUs with respect to a central handling of reactions of the system upon detected malfunctions, e.g. open circuit or shortcut. Therefore, the immediate domain of applicability of the FIM is currently body, chassis and powertrain ECUs. However, there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

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

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

5 Dependencies on other modules

FIM044: The AUTOSAR **Function Inhibition Manager (FIM)** has interfaces and dependencies on the Diagnostic Event Manager (DEM), the Software Components (SW-C) with FID interface, the ECU State Manager and the BSW modules supposed to be inhibited by the FIM.


SW-C
Functionality (Monitoring function)
SW-C
Functionality (Application, e.g. window lift)
AUTOSAR Software
AUTOSAR Runtime Environment
(RTE)
Basic Software
AUTOSAR Services
ECU State Manager
FIM
DEM
BSW module(s)

- The **Diagnostic Event Manager (DEM)** is in charge of handling 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 to execute functionality identified by an FID at the FIM. The FIDs have 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 the FIM shall 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 realized by using a template similar to the SW-component template. The interface handling for BSW modules corresponds to the interface handling for SW-components.

5.1 AUTOSAR service

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

5.1.1 Architecture

In the AUTOSAR ECU Architecture [2], the Diagnostic Event Manager [10] implements an AUTOSAR Service which is responsible for the handling and storage of faults.

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, typically one instance of the FIM Service and several Atomic Software Component instances using this Service are employed. The Atomic Software Components are named “clients” further on in this document.

Additionally, 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 that is responsible for the data exchange via RTE.

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 required value range 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. Additionally to 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 = TRUE
}
```

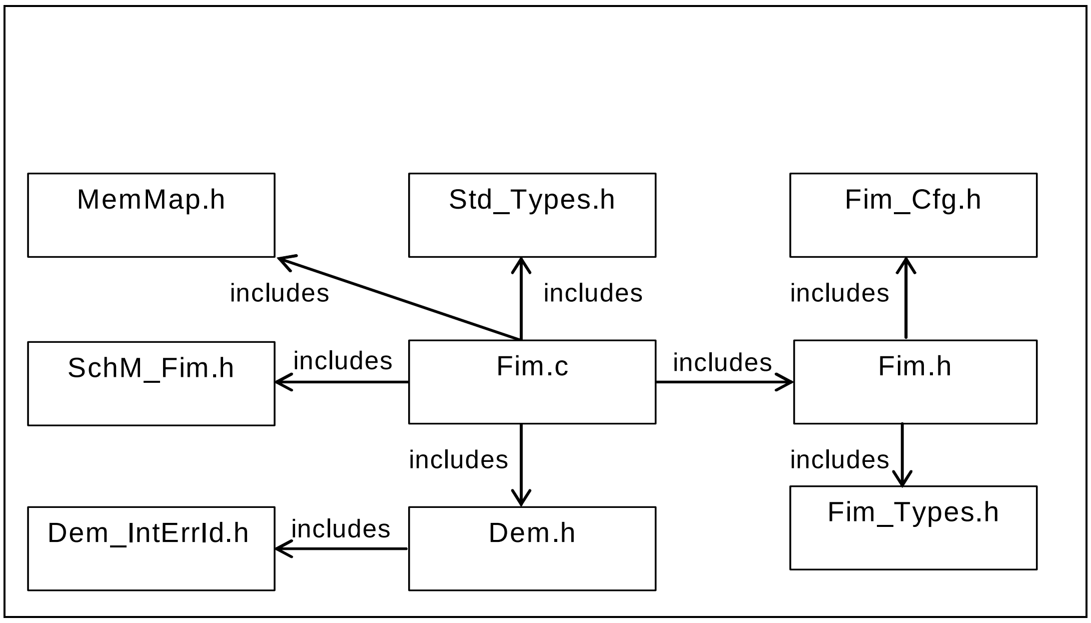
5.5 Configuration of the FIDs

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 for the FIDs. Note that the ports visible on the client side are not affected by this.

5.6 File structure

FIM030: The FIM module shall provide a file named `Fim_PBcfg.c` containing all post build time configurable parameters.

FIM029: The FIM module shall adhere to the following include file structure:



FIM031: The FIM 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 µC 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 µC abstraction layer)
[BSW00162] ECU layout abstraction	Not applicable (requirement for µC abstraction layer)
[BSW00164] Implementation of interrupt service routines	Not applicable (requirement for OS, complex drivers and µC 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	FIM027FIM041
[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 µC 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 µC 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 µC 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)
[BSW00358] Return type of init() functions	FIM004, FIM006, FIM045, FIM059

Requirement	***Satisfied by***
[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, 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

Requirement	***Satisfied by***
	(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
Memory Mapping	Not applicable (Implementation requirement)

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	FIM011

functionalities	
[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. In the FIM context an FID (FID – function identifier) identifies an application functionality 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). During configuration of the FIM, data structures (i.e. 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 Definition of ´Diagnostic Event´

A ´Diagnostic Event´ is an identifier provided by the DEM to a specific diagnostic monitor function to report an error. The status of a ´Diagnostic Event´ represents the result of a monitoring function or the report of a Basic Software Module.
See AUTOSAR_SWS_DEM document for further details [10].

7.2.1.2 Definition of ´Summarized Event´

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

During the configuration process, these single events can be combined to a summarized event (FIM037). A summarized event simplifies dealing with the multiple events that are associated with or represented by the particular summarized event. For simplicity, this particular summarized event can be used as an inhibit condition in the SW-C templates.

FIM064: The FIM shall also be able to process the inhibit condition if one of the associated summarized events is reported to the FIM.

Hence, the particular summarized event is just a representative of multiple diagnostic events (ref.10.2.4). A use case for summarized events is for example the combination of all error conditions that indicate a failed sensor.

7.2.1.3 Definition of ´Function Identifier´

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

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

FIM003: The FIM module's environment shall use the FunctionId to directly point to the associated functionality information (permission status etc.)

Note: 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 the 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).

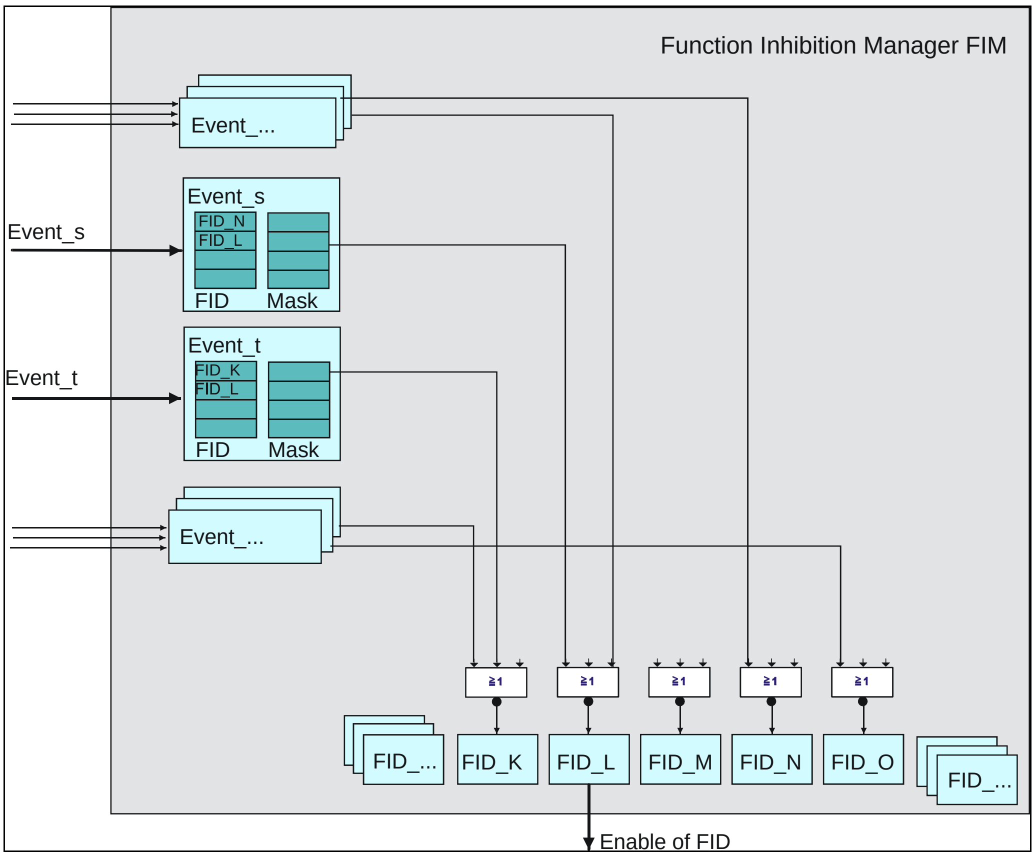


Figure 1: Logical information flow to determine FID permission states for an implementation with permission state stored in RAM

The permission state of each FID is calculated based on the EventIds assigned to a 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, the 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. In this case, there is no increased process effort within the FIM at changes of any event.

7.2.1.4 Definition of ´Function Identifier permission state´

FIM015: The FID permission state contains the information whether a functionality represented by its FID can be executed. If the permission state == TRUE, the functionality associated with the FID is permitted to be executed. If the permission state == FALSE, the functionality associated with the FID is not allowed to be executed.

The permission state is based on events reported by the DEM. Therefore, the permission state does not directly consider physical conditions (e.g. temperature, engine speed…) but those conditions reported to the DEM (e.g. sensor defect).

Additionally to the permission state as prerequisite, the activity state (is the function active or not) 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.

Hint: 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 by the calibration system (to be defined by AUTOSAR) during the development phase of the ECU.

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.

7.2.2 FIM core functionalities

7.2.2.1 FIM Data Structure

FIM013: The configuration process of the FIM shall create data structures within the FIM module to store the inhibit relations (EventID – FID – applicable mask).

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 for each configured event, a corresponding inhibit mask exists.

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 exemplarily shown in Figure 2.

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

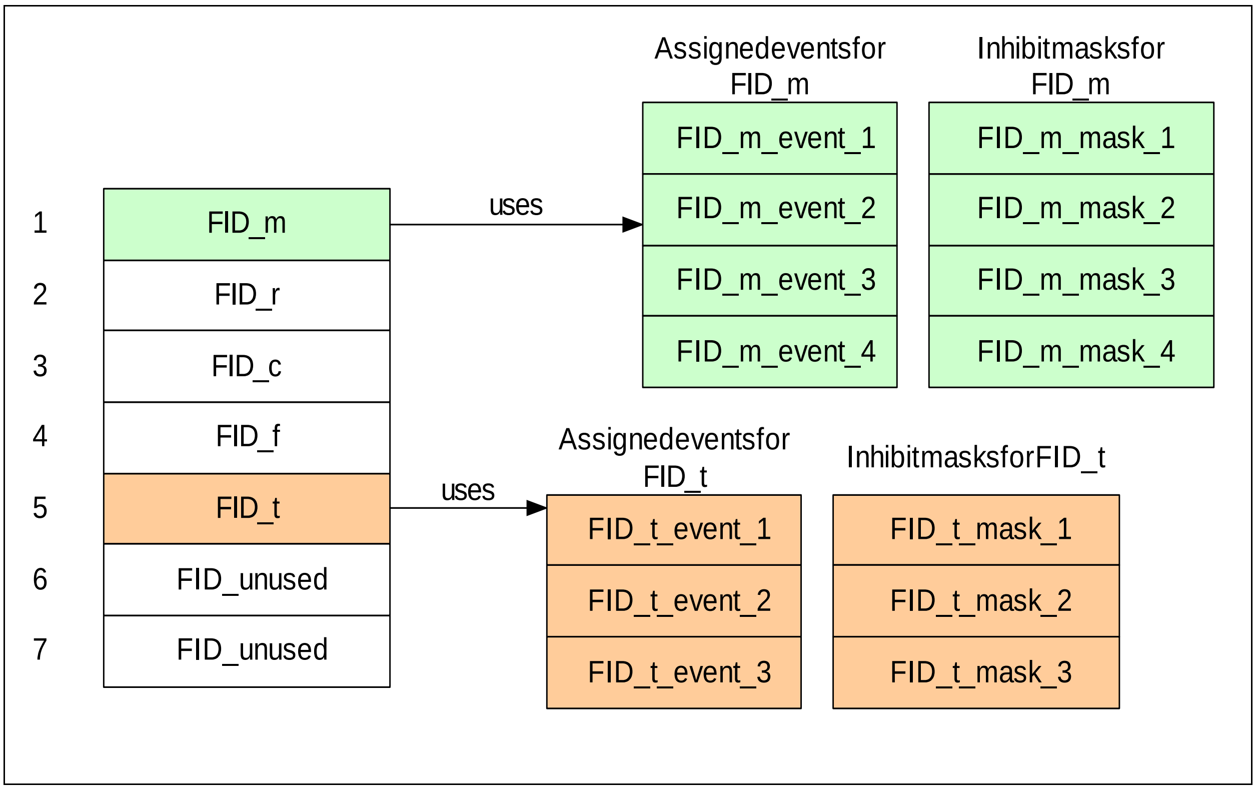


Figure 2: Inhibit mask

FIM008: The FIM module shall provide the possibility to modify the inhibit conditions by post-built configuration.

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 module is to provide services to control (permit / inhibit) functionality within SW-Cs based on DEM events being supported as inhibit conditions.

FIM065: 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 if the FIM is not configured to use polling mode. For this purpose, it shall use the API function `Fim_DemTriggerOnEventStatus`.

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), the implementation of the FIM highly depends on requirements (e.g. timing requirements) derived from applications. If an application requires fast reaction times the FIM has to provide FID information sufficiently fast to allow triggering limp-home functionality.

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 without effect.

As an example implementation, Figure 3 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 if the inhibition counter > 0 and vice versa released if the inhibition counter == 0.

Furthermore, it is important to be sure upon a reported “passed” result (which means that the diagnostic monitor function did not detect an error and therefore reported “Dem_SetEventStatus(eventId, passed)” to the DEM) whether any other inhibition reason applies. Such a passed result leads to a decrementation but only if the value of the inhibition counter then reaches 0, the FID becomes released.

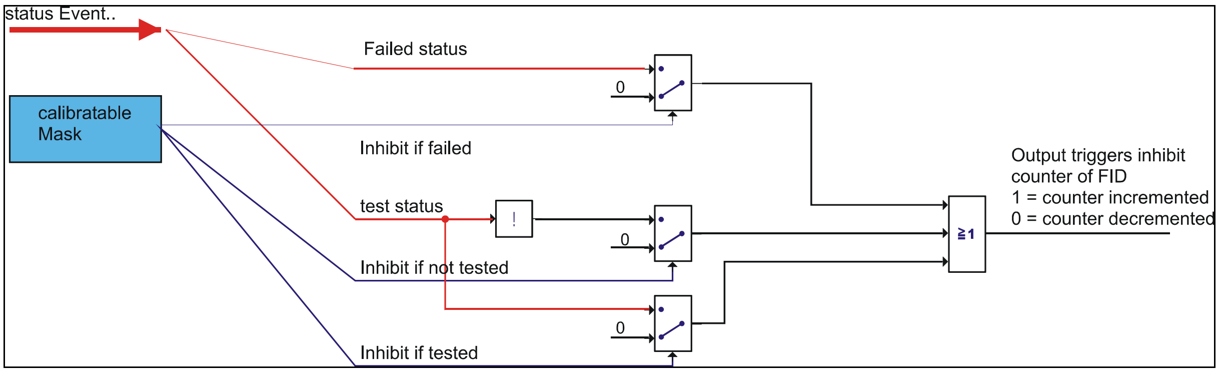


Figure 3: Calculation of permission state based on event status information

FIM012: The FIM module shall calculate the inhibit status based on the actual status of the inhibit source and the calibrated mask which exists for each inhibit source (ref. 10.2.5). The FIM module shall inhibit the FID if the Event status is equal to the calibrated mask (=Defect, Tested, NotTested). The inhibition is deactivated if the mask of the event does not match anymore the calibrated value.

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: The configuration engineer shall provide at compile time the inhibit conditions for each FID required for handling the dependencies of functionalities and events in the FIM module.

Note, that modifications by calibration shall be possible. The configuration mechanism of the FIM using SW-component template contents shall consider these requirements.

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. Chapter 10 introduces 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.

FIM020: The FIM module shall ensure an immediate control of functionality by synchronously responding to an incoming permission query. The FIM module shall realize this behavior 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

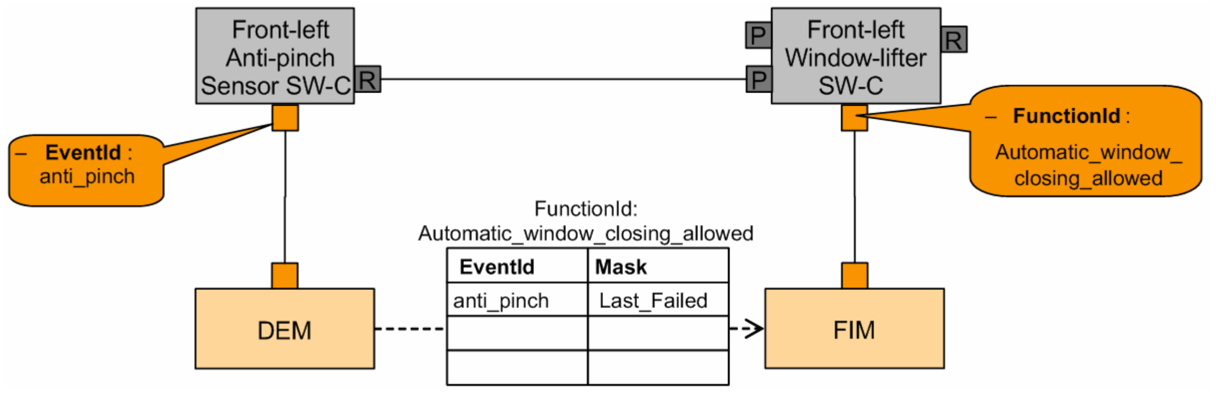

Front-left Anti-pinch Sensor SW-C
R
P
P
Front-left Window-lifter SW-C
R
– EventId : anti_pinch
– FunctionId : Automatic_window_closing_allowed
FunctionId: Automatic_window_closing_allowed
EventId
Mask
anti_pinch
Last_Failed
DEM
FIM

Figure 4: FIM usage

- ➢ The configuration of the FIM actually establishes the relationship between the EventId and the assigned FunctionId(s)
- ➢ The required 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: If DEM events status information is used, the FIM module shall compute the permission states for all FIDs at its initialization based on all restored event status information (not only events stored in the fault memory) of the DEM.

Hence, the FIM needs to be initialized after the DEM.

If FIM and DEM are delivered as one bundle, the FIM may have direct access to event information (structure) in the DEM rather than using the API `Dem_GetEventStatus`. In this way, a time-efficient initialization of the FIM can be achieved.

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 needs to be evaluated quickly, e.g. in the `Fim_Init` function. If DEM and FIM are implemented 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 matches. For this purpose, the pre-processor check shall use 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.

FIM076: The FIM module shall detect the following errors and exceptions depending on its configuration (development/production):

Type or error	***Relevance***	***Related error code***	***Value [hex]***
Fim_GetFunctionPermission 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_GetFunctionPermission 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`
API is invoked with NULL Pointer	Development	`FIM_E_INVALID_POINTER`	`0x05`
FIM is not initialized	Development	`FIM_E_NOT_INITED`	`0x06`

7.4 Error detection

FIM048: The detection of development errors within the FIM module shall be configurable (*ON* / *OFF*) at pre-compile time. The switch `FimDevErrorDetect` (see chapter 10) shall activate or deactivate the detection of all development errors.

FIM049: If the `FimDevErrorDetect` switch is enabled, API parameter checking within the FIM module 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: The FIM module shall report detected development errors to the error hook of the Development Error Tracer (DET) if the pre-processor switch `FimDevErrorDetect` is set (see chapter 10).

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed:

FIM081:

Header file	***Imported Type***
Dem_Types.h	Dem_EventIdType
Fim_Types.h	Fim_FunctionIdType
Std_Types.h	Std_VersionInfoType
	Std_ReturnType
SchM_Types.h	SchM_ReturnType

8.2 Type definitions

8.2.1 Fim_FunctionIdType

Name:	`Fim_FunctionIdType`	
Type:	`uint16,uint8`	

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:	Type for the FunctionID	

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

FIM077:

Service name:	Fim_Init

Syntax:	`void Fim_Init(`
	
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This service initializes the FIM.

FIM004: The ECU State Manager shall call the function `Fim_Init` during the startup phase of the ECU in order to initialize the permission states of the FIDs based on the event data of the DEM.

The FIM is not functional until this function has been called.

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.

FIM071: The FIM module's environment shall make sure that the DEM has been initialized before the FIM is initialized.

FIM045: If development error detection is turned on the FIM module shall report an error to the DET if it has not successfully completed the initialization and has detected not permitted access.

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.

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 are implemented as a cluster. In this case, the FIM needs to have knowledge about the data structure of the DEM so that it can directly access EventId states.

FIM072: If DEM and FIM are implemented as two separate modules, the FIM module shall access the EventId states through the API call Dem_GetEventStatus.

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:	`void Fim_GetFunctionPermission( Fim_FunctionIdType FID, boolean* Permission )`	
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 (inout):	None	
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:	This service reports the permission state to the functionality.	

FIM066: The SW Components and the BSW shall use the function `Fim_GetFunctionPermission` to query for the permission to execute a certain functionality represented by the respective FID.

Fim025: The function `Fim_GetFunctionPermission` shall deliver the return value synchronously to enable direct use of this information for controlling and executing the underlying code in the software component.

FIM055: If development error detection for the module Fim is enabled: the function `Fim_GetFunctionPermission` shall perform a plausibility check on the FID range. If a FID is out of range, the function shall raise a development error and return no permission (FALSE).

FIM056: If development error detection for the module Fim is enabled: the function `Fim_GetFunctionPermission` shall check that the initialization of the module FIM has been completed. If the function detects that the initialization is not complete, it shall raise a development error and return no permission (FALSE).

8.3.3 Interface DEM ⇔ FIM

8.3.3.1 Fim_DemTriggerOnEventStatus

FIM021:

Service name:	Fim_DemTriggerOnEventStatus	
Syntax:	`void Fim_DemTriggerOnEventStatus( Dem_EventIdType EventId, unit8 EventStatusOld, uint8 EventStatusNew )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	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 (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service to be provided to the DEM in order to call FIM upon status changes.	

FIM073: If the FIM module does *not* calculate the inhibition states in a cyclic way (as defined in configuration parameter FimCyclicEventEvaluation), the module DEM shall call the function `Fim_DemTriggerOnEventStatus` whenever the status of an events changes.

In case, the FIM module calculates the inhibition states in a cyclic way (as defined in configuration parameter FimCyclicEventEvaluation), the FIM has to query all event status information from the DEM. In that case, the DEM does *not* have to call the function Fim_DemTriggerOnEventStatus.

FIM067: 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 shall be used to determine the permission state.
If there are no permission state variables, the permission query shall access the event status information.

Service for requesting the permission state by the functionality. API is called from the SW module represented by the respective FID.

FIM057: If development error detection for the module FIM is enabled: the function `Fim_DemTriggerOnEventStatus` shall perform a plausibility check on the EventId. If an EventId is out of range, the function shall raise a development error.

FIM058: If development error detection for the module FIM is enabled: the function `Fim_DemTriggerOnEventStatus` shall check for complete initialization of the FIM. If the function detects that the initialization is not complete, it shall raise a development error.

8.3.3.2 Fim_DemInit

FIM006:

Service name:	Fim_DemInit

Syntax:	`void Fim_DemInit(`
	
	`)`

Service ID[hex]:	0x03
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This service re-initializes the FIM.

FIM068: The DEM shall call `Fim_DemInit` to re-initialize the FIM module 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).

FIM069: The function `Fim_DemInit` shall re-compute the permission state for all FIDs.

FIM082: If DEM and FIM are implemented as two separate modules, the function Fim_DemInit shall synchronously access the EventId states via the function `Dem_GetEventStatus`.

In case DEM and FIM are implemented as one bundle, the FIM module needs to have knowledge about the data structure of the DEM so that it can directly access the EventId states.

8.3.4 Fim_GetVersionInfo

FIM078:

Service name:	Fim_GetVersionInfo
Syntax:	`void Fim_GetVersionInfo(`

	`Std_VersionInfoType* versioninfo`
	 `)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	This service returns the version information of this module.	

FIM032: The function `Fim_GetVersionInfo` shall return the version information of the FIM module. The version information includes:
- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

FIM033: The function `Fim_GetVersionInfo` shall be pre-compile time configurable `On/Off` by the configuration parameter `FimVersionInfoApi`

FIM074: If source code for caller and callee of the function `Fim_GetVersionInfo` is available, the module FIM should realize this function as a macro, defined in the module's header file.

8.4 Call-back notifications

This chapter lists all functions provided by the FIM module and used by lower layer modules.

No callback notifications are specified.

8.5 Scheduled functions

This chapter lists all functions provided by the FIM module and called directly by the Basic Software Module Scheduler.

8.5.1 Fim_MainFunction

FIM060:

Service name:	Fim_MainFunction

Syntax:	`void Fim_MainFunction(`
	
	 `)`

Service ID[hex]:	0x05
Timing:	FIXED_CYCLIC
Description:	--

The evaluation of permission states can be performed either on event change or cyclically.

FIM070: In case, the FIM module is configured for cyclic evaluation of the permission states (as defined in configuration parameter FimCyclicEventEvaluation), the FIM module shall provide the function `Fim_MainFunction`. Within this function, the FIM shall cyclically calculate the permission states.

8.6 Expected Interfaces

This chapter lists all functions the module FIM requires from other modules.

8.6.1 Mandatory Interfaces

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

FIM079:

API function	***Description***
SchM_ActMainFunction_<ModulePrefix>	Invokes the SchM_ActMainFunction function to trigger the activation of a corresponding main processing function.
Dem_GetEventStatus	Gets the extended event status of an event.
SchM_CancelMainFunction_<ModulePrefix>	Invokes the SchM_CancelMainFunction function to trigger the cancellation of the requested activation 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.

FIM080:

API function	***Description***
Det_ReportError	Service to report development errors.

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.

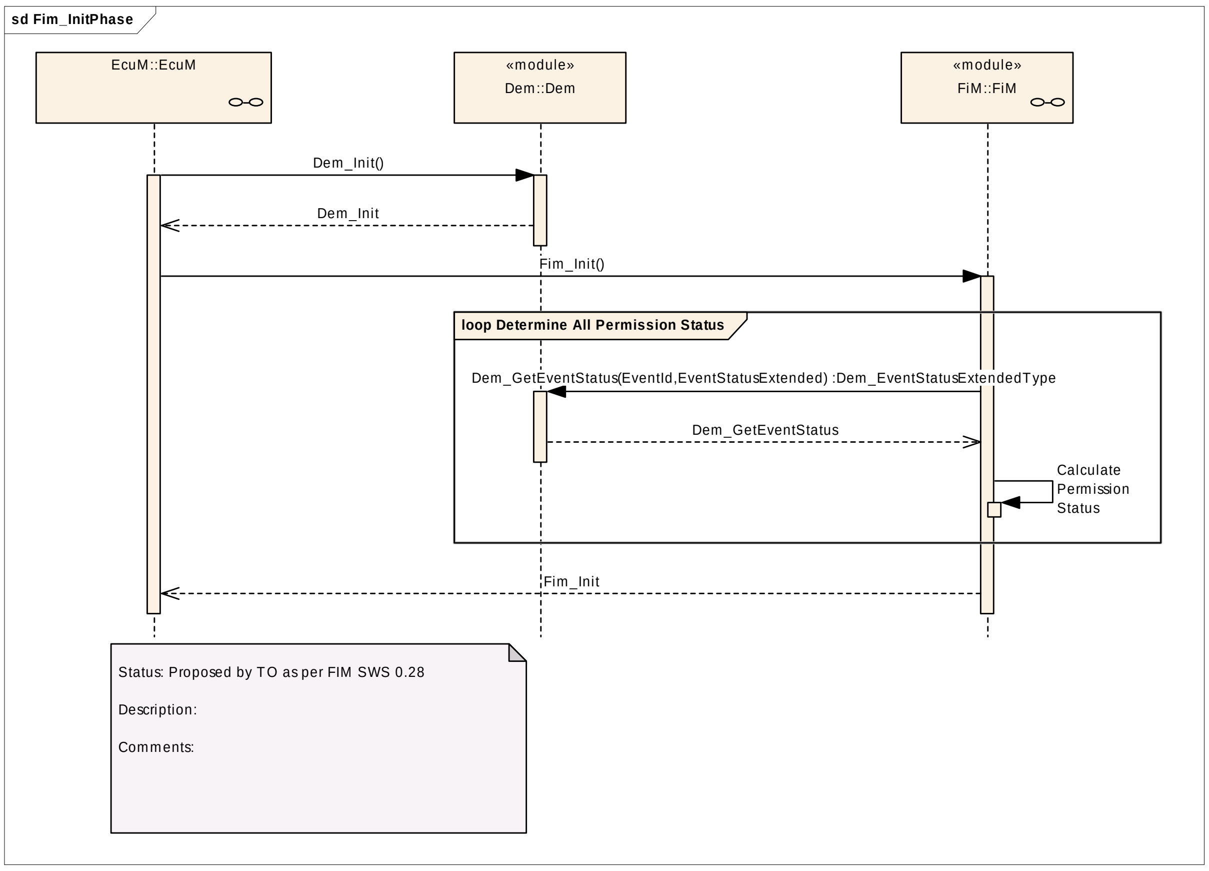
No configurable interface

9 Sequence diagrams

9.1 Initialization sequence of FIM

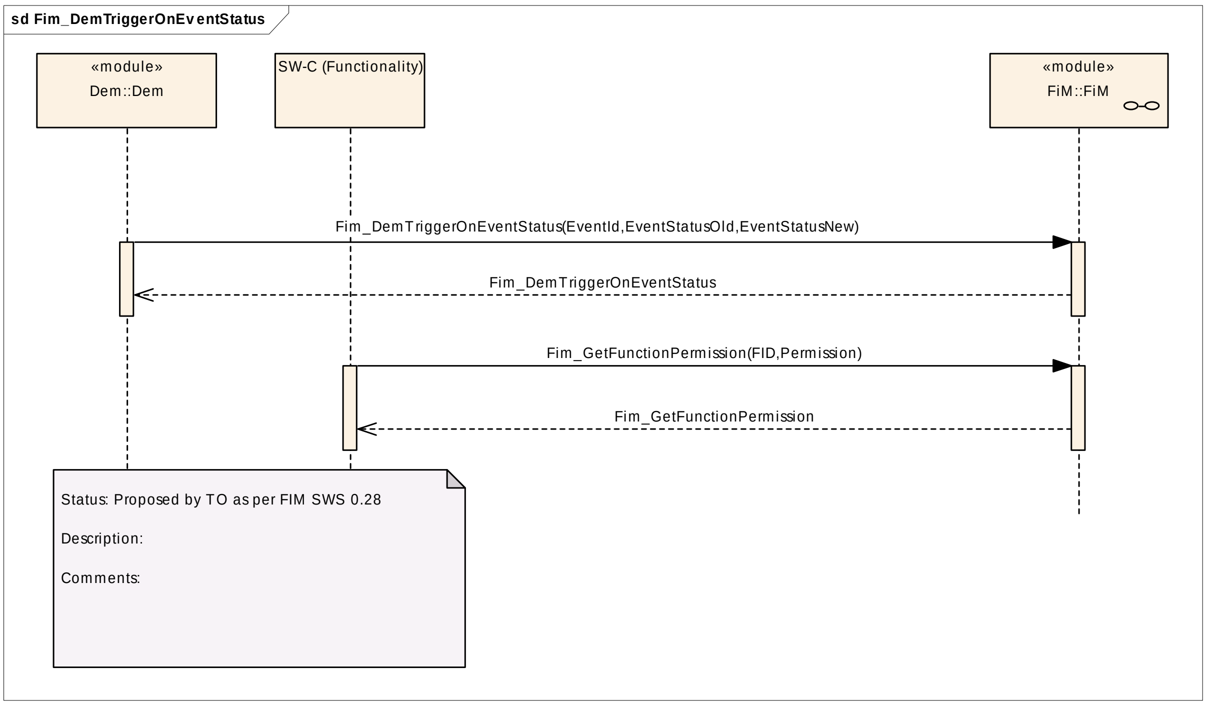
`Dem_Init` is called by the ECU State Manager as soon as the NVRAM Manager has copied the error memory relevant data block(s) from NVRAM to RAM ([10], DEM065). After the initialization of the DEM, the ECU State Manager is able to call `Fim_Init` and the FIM is able to calculate the permission states 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`.

FIM075: If the copying process of NVRAM data has not been successful, the ECU State Manager shall not call `Dem_Init` and `Fim_Init`.


sd Fim_InitPhase
EcuM::EcuM
«module»
Dem::Dem
«module»
FiM::FiM
Dem_Init()
Dem_Init
Fim_Init()
loop Determine All Permission Status
Dem_GetEventStatus(EventId,EventStatusExtended) :Dem_EventStatusExtendedType
Dem_GetEventStatus
Calculate Permission Status
Fim_Init
Status: Proposed by TO as per FIM SWS 0.28
Description:
Comments:

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`.


sd Fim_DemTriggerOnEventStatus
«module»
Dem::Dem
SW-C (Functionality)
«module»
FiM::FiM
Fim_DemTriggerOnEventStatus(EventId,EventStatusOld,EventStatusNew)
Fim_DemTriggerOnEventStatus
Fim_GetFunctionPermission(FID,Permission)
Fim_GetFunctionPermission
Status: Proposed by TO as per FIM SWS 0.28
Description:
Comments:

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.

10.1.1Configuration 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 are described in Chapter 7 and Chapter 7.3.

FIM062:

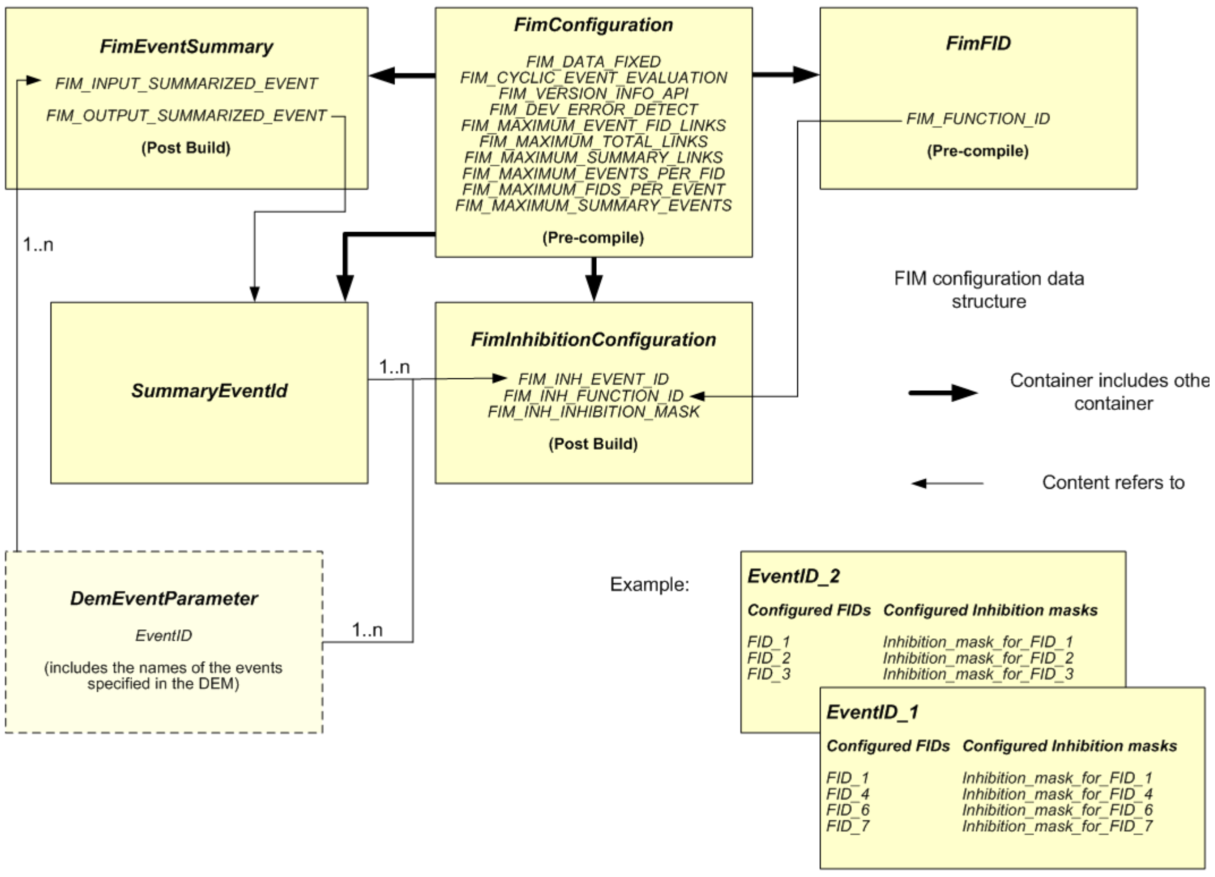

FimEventSummary
FIM_INPUT_SUMMARIZED_EVENT
FIM_OUTPUT_SUMMARIZED_EVENT
(Post Build)
FimConfiguration
FIM_DATA_FIXED
FIM_CYCLIC_EVENT_EVALUATION
FIM_VERSION_INFO_API
FIM_DEV_ERROR_DETECT
FIM_MAXIMUM_EVENT_FID_LINKS
FIM_MAXIMUM_TOTAL_LINKS
FIM_MAXIMUM_SUMMARY_LINKS
FIM_MAXIMUM_EVENTS_PER_FID
FIM_MAXIMUM_FIDS_PER_EVENT
FIM_MAXIMUM_SUMMARY_EVENTS
(Pre-compile)
FimFID
FIM_FUNCTION_ID
(Pre-compile)
1..n
FIM configuration data structure
SummaryEventId
1..n
FimInhibitionConfiguration
FIM_INH_EVENT_ID
FIM_INH_FUNCTION_ID
FIM_INH_INHIBITION_MASK
(Post Build)
Container includes othe container
Content refers to
DemEventParameter
EventID
(includes the names of the events specified in the DEM)
1..n
Example:
EventID_2
Configured FIDs Configured Inhibition masks
FID_1 Inhibition_mask_for_FID_1
FID_2 Inhibition_mask_for_FID_2
FID_3 Inhibition_mask_for_FID_3
EventID_1
Configured FIDs Configured Inhibition masks
FID_1 Inhibition_mask_for_FID_1
FID_4 Inhibition_mask_for_FID_4
FID_6 Inhibition_mask_for_FID_6
FID_7 Inhibition_mask_for_FID_7

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 Fim

Module Name	**Fim**
Module Description	Configuration of the Fim (Function Inhibition Manager) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FimEventSummary	0..*	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. 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.
FimFid	0..*	This container includes symbolic names of all FIDs.
FimGeneral	1	--
FimInhibitionConfiguration	0..*	This container includes all configuration parameters concerning the relationship between event and FID.
FimSummaryEventId	0..*	This container defines the name of a summarized event.

10.2.3 FimGeneral

SWS Item	**FIM040 :**
Container Name	FimGeneral{FimConfiguation}
Description	--
Configuration Parameters	

SWS Item	**FIM086 :**		
Name	FimCyclicEventEvaluation {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. true: FIM cyclically evaluates event states in the DEM. false: DEM informs FIM about changes of event states.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM008 :**
Name	FimDataFixed {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.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM087 :**		
Name	FimDevErrorDetect {FIM_DEV_ERROR_DETECT}		
Description	This configuration parameter is used to switch on or to switch off the detection of development errors during development.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM088 :**		
Name	FimMaxEventFidLinks {FIM_MAXIMUM_EVENT_FID_LINKS}		
Description	This configuration parameter specifies the total maximum number of links between EventIds and FIDs.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM089 :**		
Name	FimMaxEventsPerFid {FIM_MAXIMUM_EVENTS_PER_FID}		
Description	This configuration parameter specifies the maximum number of EventIds that can be linked to a single FID.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM090 :**
Name	FimMaxFidsPerEvent {FIM_MAXIMUM_FIDS_PER_EVENT}
Description	This configuration parameter specifies the maximum number of FIDs that

	can be linked to a single event.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM091 :**		
Name	FimMaxSummaryEvents {FIM_MAXIMUM_SUMMARY_EVENTS}		
Description	This configuration parameter specifies the maximum number of summarized events that can be configured.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM092 :**		
Name	FimMaxSummaryLinks {FIM_MAXIMUM_SUMMARY_LINKS}		
Description	This configuration parameter specifies the total maximum number of links between EventIds and summarized events.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM093 :**		
Name	FimMaxTotalLinks {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.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FIM094 :**		
Name	FimVersionInfoApi {FIM_VERSION_INFO_API}		
Description	This configuration parameter is used to switch on or to switch off the API to get the version information.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.4FimFid

SWS Item	**FIM039 :**
Container Name	FimFid{FimFID}
Description	This container includes symbolic names of all FIDs.
Configuration Parameters	

SWS Item	**FIM085 :**		
Name	FimFuntionId {FIM_FUNCTION_ID}		
Description	The configuration parameter is used as an ID which represents a functionality. FimFuntionId is the unique identifier assigned during FIM configuration. Implementation Type: Fim_FunctionIdType.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.5FimEventSummary

SWS Item	**FIM037 :**
Container Name	FimEventSummary{FimEventSummary}
Description	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. 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.
Configuration Parameters	

SWS Item	**FIM083 :**		
Name	FimInputSumEventRef {FIM_INPUT_SUMMARIZED_EVENT}		
Description	--		
Multiplicity	1		
Type	Reference to DemEventParameter		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	**FIM084 :**		
Name	FimOutputSumEventRef {FIM_OUTPUT_SUMMARIZED_EVENT}		
Description	--		
Multiplicity	1		
Type	Reference to FimSummaryEventId		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.6FimInhibitionConfiguration

SWS Item	**FIM038** :
Container Name	FimInhibitionConfiguration{FimInhibitionConfiguration}
Description	This container includes all configuration parameters concerning the relationship between event and FID.
Configuration Parameters	

SWS Item	**FIM096** :		
Name	FimInhInhibitionMask {FIM_INH_INHIBITION_MASK}		
Description	The configuration parameter is used to specify the inhibition mask for an event - FID relation.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FIM_LAST_FAILED	Last Failed - Use case: Re-configuration, avoiding follow-up errors	
	FIM_NOT_TESTED	Not Tested this cycle - Use case: Scheduling of monitors.	
	FIM_TESTED	Tested - Use case: Self-deactivation, check during driving cycle.	
	FIM_TESTED_AND_FAILED	Tested and Failed - Use case: Avoiding deadlocks, repeated monitoring.	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD

Scope / Dependency	

SWS Item	**FIM095** :		
Name	FimInhFunctionIdRef {FIM_INH_FUNCTION_ID}		
Description	--		
Multiplicity	1		
Type	Reference to FimFid		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FimInhEventId	1	The configuration parameter is used for an existing DEM event and summarized events as well.

10.2.7 FimInhEventId

SWS Item	**FIM097 :**
Container Name	FimInhEventId{FIM_INH_EVENT_ID}
Description	The configuration parameter is used for an existing DEM event and summarized events as well.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FimInhRefChoice	1	--

10.2.8 FimInhChoiceDemRef

SWS Item	**FIM099 :**
Container Name	FimInhChoiceDemRef
Description	--
Configuration Parameters	

SWS Item	**FIM100 :**		
Name	FimInhEventRef		
Description	--		
Multiplicity	1		
Type	Reference to DemEventParameter		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.9 FimInhChoiceSumRef

SWS Item	**FIM101 :**
Container Name	FimInhChoiceSumRef
Description	--
Configuration Parameters	

SWS Item	**FIM102 :**		
Name	FimInhSumRef		
Description	--		
Multiplicity	1		
Type	Reference to FimSummaryEventId		
ConfigurationClass	***Pre-compile time***	--	
	Link time	--	
	Post-build time	--	
Scope / Dependency			

No Included Containers

10.2.10 FimInhRefChoice

SWS Item	**FIM098 :**
Choice Container Name	FimInhRefChoice
Description	--

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
FimInhChoiceDemRef	1	--
FimInhChoiceSumRef	1	--

10.2.11 FimSummaryEventId

SWS Item	**FIM103 :**
Container Name	FimSummaryEventId{FimSummaryEventId}
Description	This container defines the name of a summarized event.
Configuration Parameters	

SWS Item	**FIM104 :**		
Name	FimEventSumId {FIM_SUMMARIZED_EVENT}		
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 FimSummaryEventId) and Inhibition configuration (as FimInhEventId).		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	L	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

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.

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [12] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.

11 Changes by AUTOSAR Technical Office

11.1 Deleted SWS Items

SWS Item	***Rationale***

11.2 Replaced SWS Items

SWS Item of Release 1	***replaced by SWS Item***	***Rationale***

11.3 Changed SWS Items

Many requirements have been changed to improve understandability without changing the technical contents.

11.4 Added SWS Items

SWS Item	***Rationale***
FIM064	SWS Improvement: Requirement had no ID
FIM065	SWS Improvement: Requirement had no ID
FIM066	SWS Improvement: Requirement had no ID
FIM067	SWS Improvement: Requirement had no ID
FIM068	SWS Improvement: Requirement had no ID
FIM069	SWS Improvement: Requirement had no ID
FIM070	SWS Improvement: Requirement had no ID
FIM071	Out of Fim_Init table
FIM072	Requirement had no ID
FIM073	Requirement had no ID
FIM074	Requirement had no ID
FIM075	Requirement had no ID
FIM076	Gave an id to an existing table
FIM081	UML Model linking of imported types
FIM077	UML Model linking of Fim_Init
FIM011	UML Model linking of Fim_GetFunctionPermission
FIM021	UML Model linking of Fim_DemTriggerOnEventStatus
FIM006	UML Model linking of Fim_DemInit
FIM078	UML Model linking of Fim_GetVersionInfo
FIM060	UML Model linking of Fim_MainFunction
FIM079	UML Model linking of mandatory interfaces
FIM080	UML Model linking of optional interfaces
FIM082	Requirement had no ID