

<b>Document Title</b>	Specification of Module Memory Abstraction Interface
<b>Document Owner</b>	AUTOSAR GbR
<b>Document Responsibility</b>	AUTOSAR GbR
<b>Document Identification No</b>	285
<b>Document Classification</b>	Standard

<b>Document Version</b>	1.2.1
<b>Document Status</b>	Final
<b>Part of Release</b>	3.1
<b>Revision</b>	0001

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
23.06.2008	1.2.1	AUTOSAR Administration	Legal disclaimer revised
13.11.2007	1.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Broadcast identifier properly explained</li> <li>• Small reformulations resulting from table generation</li> <li>• Tables in chapters 8 and 10 generated from UML model</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>
14.02.2007	1.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• File include structure updated</li> <li>• Return types of various APIs adapted</li> <li>• Ranges of configuration parameters adjusted</li> <li>• Legal disclaimer revised</li> <li>• Release Notes added</li> <li>• “Advice for users” revised</li> <li>• “Revision Information” added</li> </ul>
27.04.2006	1.0.0	AUTOSAR Administration	Initial Release

Page left intentionally blank

## Disclaimer

This document of a specification as released by the AUTOSAR Development Partnership is intended **for the purpose of information only**. The commercial exploitation of material contained in this specification requires membership of the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of this specification. 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 © 2004-2008 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 .....	8
4	Constraints and assumptions .....	9
4.1	Limitations .....	9
4.2	Applicability to car domains.....	9
5	Dependencies to other modules.....	10
5.1	File structure .....	10
5.1.1	Code file structure .....	10
5.2	Header file structure .....	10
6	Requirements traceability .....	12
7	Functional specification .....	19
7.1	General behavior.....	19
7.2	Error classification.....	19
7.3	Error detection.....	19
7.4	Error notification .....	19
8	API specification.....	20
8.1	Imported types.....	20
8.1.1	Standard types .....	20
8.2	Type definitions .....	20
8.2.1	MemIf_StatusType .....	20
8.2.2	MemIf_JobResultType .....	21
8.2.3	MemIf_ModeType .....	21
8.3	Function definitions .....	21
8.3.1	MemIf_SetMode.....	22
8.3.2	MemIf_Read.....	23
8.3.3	MemIf_Write.....	24
8.3.4	MemIf_Cancel .....	24
8.3.5	MemIf_GetStatus .....	24
8.3.6	MemIf_GetJobResult .....	25
8.3.7	MemIf_InvalidateBlock .....	25
8.3.8	MemIf_GetVersionInfo .....	26
8.3.9	MemIf_EraseImmediateBlock .....	26
8.4	Call-back notifications .....	27
8.5	Scheduled functions .....	27
8.6	Expected Interfaces.....	27
8.6.1	Mandatory Interfaces .....	27
8.6.2	Optional Interfaces .....	27
8.6.3	Configurable interfaces .....	28
9	Sequence diagrams .....	29

- 10 Configuration specification ..... 30
  - 10.1 How to read this chapter ..... 30
    - 10.1.1 Configuration and configuration parameters ..... 30
    - 10.1.2 Containers ..... 30
    - 10.1.3 Specification template for configuration parameters ..... 30
  - 10.2 Containers and configuration parameters ..... 31
    - 10.2.1 Variants ..... 31
    - 10.2.2 MemIf ..... 31
    - 10.2.3 MemIfGeneral ..... 31
  - 10.3 Published Information ..... 32
- 11 Changes during SWS Improvements by Technical Office ..... 34
  - 11.1 Deleted SWS Items ..... 34
  - 11.2 Replaced SWS Items ..... 34
  - 11.3 Changed SWS Items ..... 34
  - 11.4 Added SWS Items ..... 34

# 1 Introduction and functional overview

This specification describes the functionality, API and configuration of the AUTOSAR Basic Software Module “Memory Abstraction Interface” (MemIf). This module allows the NVRAM manager to access several memory abstraction modules (FEE or EA modules) (see Figure 1).

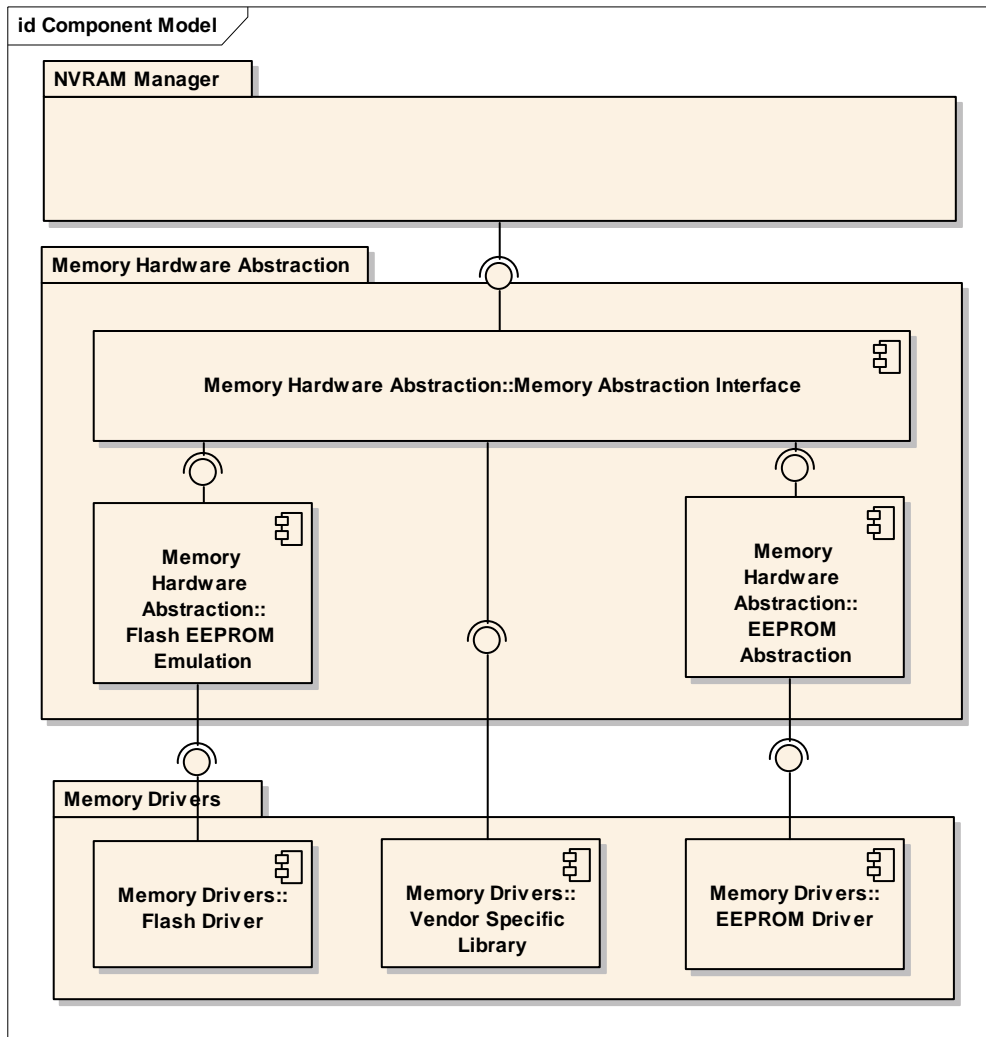


Figure 1: Module overview of memory hardware abstraction layer

**MemIf001:** The Memory Abstraction Interface (MemIf) shall abstract from the number of underlying FEE or EA modules and provide upper layers with a virtual segmentation on a uniform linear address space.

## 2 Acronyms and abbreviations

Acronyms and abbreviations which have a local scope and therefore are not contained in the AUTOSAR glossary must appear in a local glossary.

<b>Abbreviation / Acronym:</b>	<b>Description:</b>
EA	EEPROM Abstraction
EEPROM	Electrically Erasable and Programmable ROM (Read Only Memory)
FEE	Flash EEPROM Emulation
LSB	Least significant bit / byte (depending on context). Here it's bit.
MemIf	Memory Abstraction Interface
MSB	Most significant bit / byte (depending on context). Here it's bit.
NvM	NVRAM Manager
NVRAM	Non-volatile RAM (Random Access Memory)
Fast Mode	E.g. during startup / shutdown the underlying driver may be switched into fast mode in order to allow for fast reading / writing in those phases.  <i>Note: Whether this is possible depends on the implementation of the driver and the capabilities of the underlying device. Whether it is done depends on the configuration of the NVRAM manager and thus on the needs of a specific project.</i>
Slow Mode	During normal operation the underlying driver may be used in slow mode in order to reduce the resource usage in terms of runtime or blocking time of the underlying device / communication media.  <i>Note: Whether this is possible depends on the implementation of the driver and the capabilities of the underlying device. Whether it is done depends on the configuration of the NVRAM manager and thus on the needs of a specific project.</i>

## 3 Related documentation

### 3.1 Input documents

[1] List of Basic Software Modules

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_BasicSoftwareModules.pdf

[2] Layered Software Architecture

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SRS\_General.pdf

[4] General Requirements on SPAL

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SRS\_SPAL\_General.pdf

[5] Requirements on Memory Hardware Abstraction Layer

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SRS\_MemHW\_AbstractionLayer.doc

[6] Specification of Development Error Tracer

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SWS\_DET.pdf

### 3.2 Related standards and norms

[7] AUTOSAR Specification of NVRAM Manager

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SWS\_NVRAM\_Manager.doc

[8] Specification of Flash EEPROM Emulation

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SWS\_Flash\_EEPROM\_Emulation.pdf

[9] Specification of EEPROM Abstraction

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_SWS\_EEPROM\_Abstraction.pdf

[10] AUTOSAR Basic Software Module Description Template,

[https://svn2.autosar.org/repos2/22\\_Releases](https://svn2.autosar.org/repos2/22_Releases)

AUTOSAR\_BSW\_Module\_Description.pdf



## **4 Constraints and assumptions**

### **4.1 Limitations**

No limitations.

### **4.2 Applicability to car domains**

No restrictions.

## 5 Dependencies to other modules

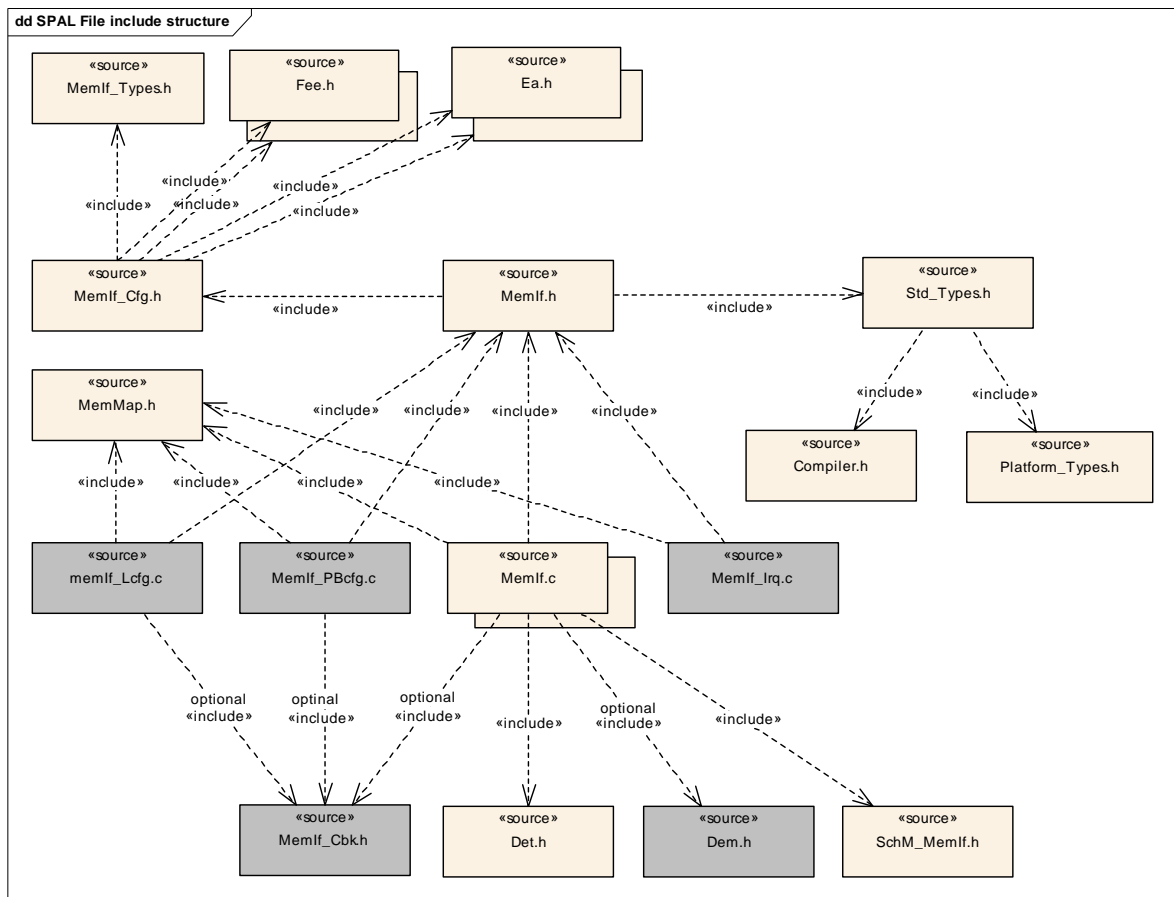
### 5.1 File structure

#### 5.1.1 Code file structure

**MemIf033:** The code file structure shall not be defined within this specification.

### 5.2 Header file structure

**MemIf002:** The file include structure shall be as follows:



**Figure 2: Memory Abstraction Layer File Include Structure**

- MemIf\_Cfg.h shall include MemIf\_Types.h and the header files of all underlying memory abstraction modules (FEE and EA modules)
- MemIf.h shall include Std\_Types.h and MemIf\_Cfg.h
- Only MemIf.h shall be included by the upper layer modules
- MemIf.c (if implemented) shall include MemIf.h, MemMap.h and other standard header files (if needed by the implementation)..

- Fee\_x.h shall include MemIf\_Types.h and the header file of the underlying flash driver
- Ea\_y.h shall include MemIf\_Types.h and the header file of the underlying EEPROM driver

**MemIf034:** The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem\_IntErrId.h.

## 6 Requirements traceability

Document: General Requirements on Basic Software Modules

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00344] Reference to link-time configuration	Not applicable (this module does not provide link-time configuration)
[BSW00404] Reference to post build time configuration	Not applicable (this module does not provide post build time configuration)
[BSW00405] Reference to multiple configuration sets	Not applicable (this module does not support multiple configuration sets)
[BSW00345] Pre-compile-time configuration	<a href="#">MemIf025</a>
[BSW159] Tool-based configuration	Not applicable (requirement on configuration, not for the specification)
[BSW167] Static configuration checking	<a href="#">MemIf005</a> , <a href="#">MemIf025</a>
[BSW171] Configurability of optional functionality	<a href="#">MemIf032</a>
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (requirement for SW-C)
[BSW00380] Separate C-File for configuration parameters	Not applicable (no link-time or post build time configuration parameters)
[BSW00381] Separate configuration header file for pre-compile time parameters	<a href="#">MemIf002</a>
[BSW00412] Separate H-File for configuration parameters [approved]	Not applicable (no link-time or post build time configuration parameters)
[BSW00383] List dependencies of configuration files	<a href="#">MemIf002</a>
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Chapter 8.6
[BSW00388] Introduce containers	Chapter 10.1
[BSW00389] Containers shall have names	Chapter 10.1
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2.2
[BSW00391] Parameter shall have unique names	Chapter 10.2.2
[BSW00392] Parameters shall have a type	Chapter 10.2.2
[BSW00393] Parameters shall have a range	Chapter 10.2.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2.2
[BSW00395] List the required parameters (per parameter)	Chapter 10.2.2
[BSW00396] Configuration classes	Chapter 10.2.2
[BSW00397] Pre-compile-time parameters	Chapter 10.2.2
[BSW00398] Link-time parameters	Not applicable (no link-time configuration parameters)
[BSW00399] Loadable Post-build time parameters	Not applicable (no post build time configuration parameters)
[BSW00400] Selectable Post-build time parameters	Not applicable (no post build time configuration parameters)
[BSW00402] Published information	Chapter 10.2.2
[BSW00375] Notification of wake-up reason	Not applicable (this module does not provide wakeup capabilities)

[BSW101] Initialization interface	Not applicable (this module does not need an initialization)
[BSW00416] Sequence of Initialization	Not applicable (requirement on system design, not a single module)
[BSW00406] Check module initialization	Not applicable (this module does not need an initialization)
[BSW168] Diagnostic Interface of SW components	Not applicable (this module does not provide special diagnostic features)
[BSW00407] Function to read out published parameters	Chapter 8.3.8, <a href="#">MemIf026</a>
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (this module does not provide an AUTOSAR interface)
[BSW00424] BSW main processing function task allocation	Not applicable (requirement on system design, not on a single module)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (requirement on the BSW module description template)
[BSW00426] Exclusive areas in BSW modules	Not applicable (no exclusive areas defined in this module)
[BSW00427] ISR description for BSW modules	Not applicable (this module does not implement any ISRs)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (only one main processing function in this module)
[BSW00429] Restricted BSW OS functionality access	Not applicable (this module does not use any OS functionality)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (requirement on the BSW scheduler)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (only one main processing function in this module)
[BSW00433] Calling of main processing functions	Not applicable (requirement on system design, not on a single module)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (requirement on the schedule module - this is not it)
[BSW00336] Shutdown interface	Not applicable (this module does not need to be shut down)
[BSW00337] Classification of errors	<a href="#">MemIf006</a>
[BSW00338] Detection and Reporting of development errors	<a href="#">MemIf007</a> , <a href="#">MemIf028</a>
[BSW00369] Do not return development error codes via API	<a href="#">MemIf028</a>
[BSW00339] Reporting of production relevant error status	Not applicable (this module does not know any production relevant errors)
[BSW00421] Reporting of production relevant error events	Not applicable (no production relevant errors defined for this module)
[BSW00422] Debouncing of production relevant error status	Not applicable (requirement on the DEM, not this module)
[BSW00420] Production relevant error event rate detection	Not applicable (requirement on the DEM, not this module)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (requirement on non BSW modules)

[BSW00323] API parameter checking	<a href="#">MemIf022</a>
[BSW004] Version check	<a href="#">MemIf005</a>
[BSW00409] Header files for production code error IDs	<a href="#">MemIf029</a>
[BSW00385] List possible error notificatons	Chapter 8.6
[BSW00386] Configuration for detecting an error	<a href="#">MemIf006</a> , <a href="#">MemIf007</a> , <a href="#">MemIf023</a> , <a href="#">MemIf028</a>
[BSW161] Microcontroller abstraction	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW162] ECU layout abstraction	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW00324] Do not use HIS I/O Library	Not applicable (architecture decision)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW00415] User dependent include files	Not applicable (only one user for this module)
[BSW164] Implementation of interrupt service routines	<a href="#">Not</a> applicable (this module does not implement any ISRs)
[BSW00325] Runtime of interrupt service routines	<a href="#">Not</a> applicable (this module does not implement any ISRs or callback routines)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (requirement on implementation, not on specification)
[BSW00342] Usage of source code and object code	Not applicable (requirement on AUTOSAR architecture, not a single module)
[BSW00343] Specification and configuration of time	<a href="#">Not</a> applicable (this module does not provide any timing configuration)
[BSW160] Human-readable configuration data	Not applicable (requirement on documentation, not on specification)
[BSW007] HIS MISRA C	Not applicable (requirement on implementation, not on specification)
[BSW00300] Module naming convention	Not applicable (requirement on implementation, not on specification)
[BSW00413] Accessing instances of BSW modules	Requirement can not be implemented in R2.0 timeframe.
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (requirement on the implementation, not on the specification)
[BSW00305] Self-defined data types naming convention	Chapter 8.2
[BSW00307] Global variables naming convention	Not applicable (requirement on the implementation, not on the specification)
[BSW00310] API naming convention	Chapter 8.3
[BSW00373] Main processing function naming convention	Not applicable (this module does not provide a scheduled function)
[BSW00327] Error values naming convention	<a href="#">MemIf006</a> , <a href="#">MemIf008</a>
[BSW00335] Status values naming convention	Chapter 8.2.1

[BSW00350] Development error detection keyword	<a href="#">MemIf007</a> , <a href="#">MemIf028</a> , <a href="#">MemIf025</a>
[BSW00408] Configuration parameter naming convention	Chapter 10.1
[BSW00410] Compiler switches shall have defined values	Chapter 10.1
[BSW00411] Get version info keyword	Chapter 10.2.2
[BSW00346] Basic set of module files	<a href="#">MemIf002</a>
[BSW158] Separation of configuration from implementation	<a href="#">MemIf002</a>
[BSW00314] Separation of interrupt frames and service routines	Not applicable (this module does not implement any ISRs)
[BSW00370] Separation of callback interface from API	Not applicable (this module does not implement any callback routines)
[BSW00348] Standard type header	Not applicable (requirement on the standard header file)
[BSW00353] Platform specific type header	Not applicable (requirement on the platform specific header file)
[BSW00361] Compiler specific language extension header	Not applicable (requirement on the compiler specific header file)
[BSW00301] Limit imported information	<a href="#">MemIf002</a>
[BSW00302] Limit exported information	Not applicable (requirement on the implementation, not on the specification)
[BSW00328] Avoid duplication of code	Not applicable (requirement on the implementation, not on the specification)
[BSW00312] Shared code shall be reentrant	Not applicable (requirement on the implementation, not on the specification)
[BSW006] Platform independency	Not applicable (this is a module of the microcontroller abstraction layer)
[BSW00357] Standard API return type	Chapter 8.3.2, Chapter 8.3.3. Chapter 8.3.7, Chapter 8.3.9
[BSW00377] Module specific API return types	Chapter 8.3.5, Chapter 8.3.6
[BSW00304] AUTOSAR integer data types	Not applicable (requirement on implementation, not for specification)
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (requirement on implementation, not for specification)
[BSW00378] AUTOSAR boolean type	Not applicable (requirement on implementation, not for specification)
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (requirement on implementation, not for specification)
[BSW00308] Definition of global data	Not applicable (requirement on implementation, not for specification)
[BSW00309] Global data with read-only constraint	Not applicable (requirement on implementation, not for specification)
[BSW00371] Do not pass function pointers via API	Not applicable (no function pointers in this specification)

[BSW00358] Return type of init() functions	Not applicable (this module does not provide an initialization function)
[BSW00414] Parameter of init function	Not applicable (this module does not provide an initialization function)
[BSW00376] Return type and parameters of main processing functions	Not applicable (this module does not provide a scheduled function)
[BSW00359] Return type of callback functions	Not applicable (this module does not provide any callback routines)
[BSW00360] Parameters of callback functions	Not applicable (this module does not provide any callback routines)
[BSW00329] Avoidance of generic interfaces	Chapter 8.3 (explicit interfaces defined)
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable (requirement on implementation, not for specification)
[BSW00331] Separation of error and status values	<a href="#">MemIf028</a>
[BSW009] Module User Documentation	Not applicable (requirement on documentation, not on specification)
[BSW00401] Documentation of multiple instances of configuration parameters	Not applicable (all configuration parameters are single instance only)
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (no internal scheduling policy)
[BSW010] Memory resource documentation	Not applicable (requirement on documentation, not on specification)
[BSW00333] Documentation of callback function context	Not applicable (requirement on documentation, not for specification)
[BSW00374] Module vendor identification	<a href="#">MemIf026</a>
[BSW00379] Module identification	<a href="#">MemIf026</a>
[BSW003] Version identification	<a href="#">MemIf026</a>
[BSW00318] Format of module version numbers	<a href="#">MemIf026</a>
[BSW00321] Enumeration of module version numbers	Not applicable (requirement on implementation, not for specification)
[BSW00341] Microcontroller compatibility documentation	Not applicable (requirement on documentation, not on specification)
[BSW00334] Provision of XML file	Not applicable (requirement on documentation, not on specification)



Document: General Requirements on SPAL

<b>Requirement</b>	<b>Satisfied by</b>
[BSW12263] Object code compatible configuration concept	Not applicable (this module does not provide post-compile time parameters)
[BSW12056] Configuration of notification mechanisms	Not applicable (this module does not support any notification mechanisms)
[BSW12267] Configuration of wake-up sources	Not applicable (this module does not provide any wakeup capabilities)
[BSW12057] Driver module initialization	Not applicable (this module does not provide an initialization routine)
[BSW12125] Initialization of hardware resources	Not applicable (this module has no direct hardware access)
[BSW12163] Driver module de-initialization	Not applicable (this module does not provide an de-initialization routine)
[BSW12058] Individual initialization of overall registers	Not applicable (this module has no direct hardware access)
[BSW12059] General initialization of overall registers	Not applicable (this module has no direct hardware access)
[BSW12060] Responsibility for initialization of one-time writable registers	Not applicable (this module has no direct hardware access)
[BSW12461] Responsibility for register initialization [approved]	Not applicable (this module has no direct hardware access)
[BSW12462] Provide settings for register initialization [approved]	Not applicable (this module has no direct hardware access)
[BSW12463] Combine and forward settings for register initialization	Not applicable (this module has no direct hardware access)
[BSW12062] Selection of static configuration sets	Not applicable (this module does not provide an initialization routine)
[BSW12068] MCAL initialization sequence	Not applicable (this module does not provide an initialization routine)
[BSW12069] Wake-up notification of ECU State Manager	Not applicable (this module does not provide any wakeup capabilities)
[BSW157] Notification mechanisms of drivers and handlers	Not applicable (this module does not support any notification mechanisms)
[BSW12155] Prototypes of callback functions	Not applicable (this module does not provide any callback routines)
[BSW12169] Control of operation mode	Chapter 8.3.1
[BSW12063] Raw value mode	Not applicable (this module does not handle any data)
[BSW12075] Use of application buffers	Not applicable (this module does not handle any data)
[BSW12129] Resetting of interrupt flags	Not applicable (this module does not implement any ISRs)
[BSW12064] Change of operation mode during running operation	Not applicable (this module is only an interface for underlying modules)
[BSW12448] Behavior after development error detection	<a href="#">MemIf023</a>

[BSW12067] Setting of wake-up conditions	Not applicable (this module does not provide any wakeup capabilities)
[BSW12077] Non-blocking implementation	Not applicable (this module does not provide any schedulable routines)
[BSW12078] Runtime and memory efficiency	<a href="#">MemIf019</a> , <a href="#">MemIf020</a> <a href="#">MAI018</a>
[BSW12092] Access to drivers	Not applicable (requirement on system architecture not for one module)
[BSW12265] Configuration data shall be kept constant	Not applicable (this module does not have post-compile time configuration data)
[BSW12264] Specification of configuration items	<a href="#">MemIf025</a> , <a href="#">MemIf026</a>
[BSW12081] Use HIS requirements as input	Not applicable (no corresponding HIS requirements available)

Document: Requirements on Memory Abstraction Interface

<b>Requirement</b>	<b>Satisfied by</b>
BSW14019 Provide uniform access to underlying memory abstraction modules	<a href="#">MemIf001</a> , <a href="#">MemIf017</a>
BSW14020 Selection of underlying memory abstraction modules	<a href="#">MemIf018</a>
BSW14021 Number of underlying memory abstraction modules	<a href="#">MemIf018</a> , <a href="#">MemIf019</a> , <a href="#">MemIf020</a> , <a href="#">MemIf022</a> , <a href="#">MemIf025</a>
BSW14022 Preserving of functionality	<a href="#">MemIf017</a>
BSW14023 Parameter checking	<a href="#">MemIf005</a> , <a href="#">MemIf022</a>
BSW14024 Preserving of timing behavior	Not applicable (requirement removed, see RfC 14746) <a href="#">MemIf004</a>
BSW14025 Efficient implementation	<a href="#">MemIf019</a> , <a href="#">MemIf020</a>

## 7 Functional specification

### 7.1 General behavior

**MemIf005:** All pre-compile time configuration parameters shall be checked statically (at least during compile time) for correctness. The version information in the module headers and source files shall be validated and consistent (e.g. by comparing the version information in the module headers and source files with a pre-processor macro).

### 7.2 Error classification

**MemIf029:** Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem\_IntErrId.h and included via Dem.h.

**MemIf030:** Development error values are of type uint8.

**MemIf006:** The following errors and exceptions shall be detectable by the Memory Abstraction Interface depending on its configuration (development/production):

Type or error	Relevance	Related error code	Value [hex]
API service called with wrong device index parameter	Development	MEMIF_E_PARAM_DEVICE	0x01

### 7.3 Error detection

**MemIf007:** The detection of development errors shall be configurable (on/off) at pre-compile time. The switch `MemIfDevErrorDetect` (see chapter 10) shall activate or deactivate the detection of all development errors.

**MemIf008:** A detection of errors not listed in the table above [\[MemIf006\]](#) shall not be implemented.

### 7.4 Error notification

**MemIf028:** Development errors shall be reported to the Development Error Tracer (DET) if the preprocessor switch `MemIfDevErrorDetect` is set. The error codes shall not be used as return values for the called function.

## 8 API specification

### 8.1 Imported types

#### 8.1.1 Standard types

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

##### MemIf037:

Header file	Imported Type
Std_Types.h	Std_VersionInfoType
	Std_ReturnType

### 8.2 Type definitions

**MemIf009:** The types specified in this chapter shall be located in the file MemIf\_Types.h.

**MemIf010:** The types specified in this chapter shall not be changed or extended for a specific memory abstraction module or hardware platform.

**MemIf011:** The data type for the memory device index shall be uint8. The lowest value to be used for this device index shall be 0. The allowed range of indices thus shall be 0..MemIfNumberOfDevices-1.

**MemIf036:** The symbolic name MEMIF\_BROADCAST\_ID shall be used to identify all underlying devices within one call. This special “broadcast” device ID shall only be allowed in the call to MemIf\_GetStatus to determine the status of all underlying abstraction modules and device drivers<sup>1</sup>.

#### 8.2.1 MemIf\_StatusType

<b>Name:</b>	MemIf_StatusType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	MEMIF_UNINIT	The underlying abstraction module or device driver has not been initialized (yet).
	MEMIF_IDLE	The underlying abstraction module or device driver is currently idle.
	MEMIF_BUSY	The underlying abstraction module or device driver is currently busy.
	MEMIF_BUSY_INTERNAL	The underlying abstraction module is busy with internal management operations. The underlying device driver can be busy or idle.

<sup>1</sup> I.e. used to query whether all devices are idle in order to shut down the ECU.

<b>Description:</b>	Denotes the current status of the underlying abstraction module and device drive.
---------------------	---

**MemIf015:** The type `MemIf_StatusType` denotes the current status of the underlying abstraction module and device driver. It shall be used as the return value of the corresponding driver's "GetStatus" function.

### 8.2.2 MemIf\_JobResultType

<b>Name:</b>	<code>MemIf_JobResultType</code>	
<b>Type:</b>	Enumeration	
<b>Range:</b>	<code>MEMIF_JOB_OK</code>	The job has been finished successfully.
	<code>MEMIF_JOB_FAILED</code>	The job has not been finished successfully.
	<code>MEMIF_JOB_PENDING</code>	The job has not yet been finished.
	<code>MEMIF_JOB_CANCELLED</code>	The job has been cancelled.
	<code>MEMIF_BLOCK_INCONSISTENT</code>	The requested block is inconsistent, it may contain corrupted data.
	<code>MEMIF_BLOCK_INVALID</code>	The requested block has been marked as invalid, the requested operation can not be performed.
<b>Description:</b>	Denotes the result of the last job.	

**MemIf016:** The type `MemIf_JobResultType` denotes the result of the last job.

### 8.2.3 MemIf\_ModeType

<b>Name:</b>	<code>MemIf_ModeType</code>	
<b>Type:</b>	Enumeration	
<b>Range:</b>	<code>MEMIF_MODE_SLOW</code>	The underlying memory abstraction modules and drivers are working in slow mode.
	<code>MEMIF_MODE_FAST</code>	The underlying memory abstraction modules and drivers are working in fast mode.
<b>Description:</b>	Denotes the operation mode of the underlying abstraction modules and device drivers.	

**MemIf021:** The type `MemIf_ModeType` denotes the operation mode of the underlying abstraction modules and device drivers.

## 8.3 Function definitions

**MemIf017:** The API specified in this chapter shall be mapped to the API of the underlying memory abstraction modules. For functional behavior refer to the specification of those modules respectively to that of the underlying memory drivers.

**MemIf018:** The parameter `DeviceIndex` shall be used for selection of memory abstraction modules (and thus memory devices). If only one memory abstraction module is configured, the parameter `DeviceIndex` shall be ignored.

**MemIf019:** If only one memory abstraction module is configured, the Memory Abstraction Interface shall be implemented as a set of macros mapping the Memory

Abstraction Interface API to the API of the corresponding memory abstraction module.

Example:

```
#define MemIf_Write(DeviceIndex, BlockNumber, DataPtr) \
    Fee_Write(BlockNumber, DataPtr)
```

**MemIf020:** If more than one memory abstraction module is configured, the Memory Abstraction Interface shall use efficient mechanisms to map the API calls to the appropriate memory abstraction module. One solution is to use tables of pointers to functions where the parameter `DeviceIndex` is used as array index.

Example:

```
#define MemIf_Write(DeviceIndex, BlockNumber, DataPtr) \
    MemIf_WriteFctPtr[DeviceIndex](BlockNumber, DataPtr)
```

*Note: The service IDs given in this interface specification are related to the service IDs of the underlying memory abstraction module(s). For that reason, they may not start with 0.*

**MemIf022:** If more than one memory abstraction module is configured and development error detection is enabled for this module, the parameter `DeviceIndex` shall be checked for being an existing device or the broadcast identifier within the module's services.

**MemIf023:** Detected errors shall be reported to the Development Error Tracer (DET) with the error code `MEMIF_E_PARAM_DEVICE` and the called service shall not be executed.

**MemIf024:** If the called function has a return value, it shall be set as follows:

MemIf_GetStatus:	MEMIF_UNINIT
MemIf_GetJobResult:	MEMIF_JOB_FAILED
All other functions:	E_NOT_OK.

**MemIf035:** If the function `MemIf_GetStatus` is called with the device index denoting a broadcast (`MEMIF_BROADCAST_ID`) to all configured devices (see [MemIf036](#)), this module shall call the "GetStatus" functions of all underlying devices in turn. It shall return the value

- `MEMIF_IDLE` – if all underlying devices have returned this state
- `MEMIF_UNINIT` – if at least one device returned this state, all other returned states shall be ignored
- `MEMIF_BUSY` – if at least one configured device returned this state and no other device returned `MEMIF_UNINIT`
- `MEMIF_BUSY_INTERNAL` – if at least one configured device returned this state and no other device returned `MEMIF_BUSY` or `MEMIF_UNINIT`

### 8.3.1 MemIf\_SetMode

**MemIf038:**

<b>Service name:</b>	MemIf_SetMode
<b>Syntax:</b>	void MemIf_SetMode( MemIf_ModeType Mode ) )
<b>Service ID[hex]:</b>	0x01
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	Mode --
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	None
<b>Return value:</b>	None
<b>Description:</b>	map function calls of MemIf_SetMode to service: Fee_SetMode respectively Ea_SetMode

**MemIf049:** The function MemIf\_SetMode shall be mapped to service: Fee\_SetMode respectively Ea\_SetMode.

Note: The device index was intentionally left out in the above function, that is the Memory Interface shall switch all underlying modules into the requested mode. An extra "broadcast" parameter is not needed in this case since the devices shall not be switched to different modes individually.

### 8.3.2 MemIf\_Read

**MemIf039:**

<b>Service name:</b>	MemIf_Read
<b>Syntax:</b>	Std_ReturnType MemIf_Read( uint8 DeviceIndex, uint16 BlockNumber, uint16 BlockOffset, uint8* DataBufferPtr, uint16 Length ) )
<b>Service ID[hex]:</b>	0x02
<b>Sync/Async:</b>	Synchronous
<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	DeviceIndex --
	BlockNumber --
	BlockOffset --
	Length --
<b>Parameters (inout):</b>	None
<b>Parameters (out):</b>	DataBufferPtr --
<b>Return value:</b>	Std_ReturnType --
<b>Description:</b>	map function calls of MemIf_Read to service: Fee_Read respectively Ea_Read

**MemIf050:** The function MemIf\_Read shall be mapped to service: Fee\_Read respectively Ea\_Read.

### 8.3.3 MemIf\_Write

#### MemIf040:

<b>Service name:</b>	MemIf_Write	
<b>Syntax:</b>	Std_ReturnType MemIf_Write( uint8 DeviceIndex, uint16 BlockNumber, uint8* DataBufferPtr )	
<b>Service ID[hex]:</b>	0x03	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	DeviceIndex	--
	BlockNumber	--
	DataBufferPtr	--
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	--
<b>Description:</b>	map function calls of MemIf_Write to service: Fee_Write respectively Ea_Write	

**MemIf051:** The function MemIf\_Write shall be mapped to service: Fee\_Write respectively Ea\_Write.

### 8.3.4 MemIf\_Cancel

#### MemIf041:

<b>Service name:</b>	MemIf_Cancel	
<b>Syntax:</b>	void MemIf_Cancel( uint8 DeviceIndex )	
<b>Service ID[hex]:</b>	0x04	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	DeviceIndex	--
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	None	
<b>Description:</b>	map function calls of MemIf_Cancel to service: Fee_Cancel respectively Ea_Cancel	

**MemIf052:** The function MemIf\_Cancel shall be mapped to service: Fee\_Cancel respectively Ea\_Cancel.

### 8.3.5 MemIf\_GetStatus

#### MemIf042:

<b>Service name:</b>	MemIf_GetStatus	
<b>Syntax:</b>	MemIf_StatusType MemIf_GetStatus( uint8 DeviceIndex )	



	)	
<b>Service ID[hex]:</b>	0x05	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	DeviceIndex	--
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	MemIf_StatusType	--
<b>Description:</b>	map function calls of MemIf_GetStatus to service: Fee_GetStatus respectively Ea_GetStatus	

**MemIf056:** The function MemIf\_GetStatus shall be mapped to service: Fee\_GetStatus respectively Ea\_GetStatus

*Note: In case the parameter given as device ID is MEMIF\_BROADCAST\_ID, the memory abstraction interface shall iterate over all underlying devices and return their combined statusn according to [MemIf035](#).*

### 8.3.6 MemIf\_GetJobResult

**MemIf043:**

<b>Service name:</b>	MemIf_GetJobResult	
<b>Syntax:</b>	MemIf_JobResultType MemIf_GetJobResult( uint8 DeviceIndex )	
<b>Service ID[hex]:</b>	0x06	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	DeviceIndex	--
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	MemIf_JobResultType	--
<b>Description:</b>	map function calls of MemIf_GetJobResult to service: Fee_GetJobResult respectively Ea_GetJobResult	

**MemIf053:** The function MemIf\_GetJobResult shall be mapped to service: Fee\_GetJobResult respectively Ea\_GetJobResult.

### 8.3.7 MemIf\_InvalidateBlock

**MemIf044:**

<b>Service name:</b>	MemIf_InvalidateBlock	
<b>Syntax:</b>	Std_ReturnType MemIf_InvalidateBlock( uint8 DeviceIndex, uint16 BlockNumber )	
<b>Service ID[hex]:</b>	0x07	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	DeviceIndex	--

	BlockNumber	--
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	--
<b>Description:</b>	map function calls of MemIf_InvalidateBlock to service: Fee_InvalidateBlock respectively Ea_InvalidateBlock	

**MemIf054:** The function MemIf\_InvalidateBlock shall be mapped to service: Fee\_InvalidateBlock respectively Ea\_InvalidateBlock.

### 8.3.8 MemIf\_GetVersionInfo

**MemIf045:**

<b>Service name:</b>	MemIf_GetVersionInfo	
<b>Syntax:</b>	void MemIf_GetVersionInfo( Std_VersionInfoType* VersionInfoPtr )	
<b>Service ID[hex]:</b>	0x08	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	None	
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	VersionInfoPtr	Pointer to standard version information structure.
<b>Return value:</b>	None	
<b>Description:</b>	Returns version information.	

**MemIf031:** The function MemIf\_GetVersionInfo shall return synchronously the version information on the Memory Abstraction Interface module in the structure provided by the caller.

**MemIf057:** Configurations: The function MemIf\_GetVersionInfo is only available if enabled by the pre-processor switch MemIfVersionInfoApi.

### 8.3.9 MemIf\_EraseImmediateBlock

**MemIf046:**

<b>Service name:</b>	MemIf_EraseImmediateBlock	
<b>Syntax:</b>	Std_ReturnType MemIf_EraseImmediateBlock( uint8 DeviceIndex, uint16 BlockNumber )	
<b>Service ID[hex]:</b>	0x09	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Non Reentrant	
<b>Parameters (in):</b>	DeviceIndex	--
	BlockNumber	--
<b>Parameters (inout):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	Std_ReturnType	--

<b>Description:</b>	map function calls of MemIf_EraseImmediateBlock to service: Fee_EraseImmediateBlock respectively Ea_EraseImmediateBlock
---------------------	---

**MemIf055:** The function MemIf\_EraseImmediateBlock shall be mapped to service: Fee\_EraseImmediateBlock respectively Ea\_EraseImmediateBlock.

## 8.4 Call-back notifications

None, the NVRAM manager shall provide the callback routines for the underlying memory abstraction modules.

## 8.5 Scheduled functions

None, there are no asynchronous functions in this module.

## 8.6 Expected Interfaces

### 8.6.1 Mandatory Interfaces

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

#### MemIf047:

<b>API function</b>	<b>Description</b>
Ea_Cancel	Cancels the ongoing asynchronous operation.
Fee_Cancel	Service to call the cancel function of the underlying flash driver.
Fee_GetStatus	Service to call the GetStatus function of the underlying flash driver.
Fee_Write	Service to initiate a write job.
Ea_SetMode	Sets the mode.
Ea_GetJobResult	Service to return the JobResult.
Fee_Read	Service to initiate a read job.
Ea_InvalidateBlock	Invalidates the block BlockNumber.
Fee_SetMode	Service to call the Fls_SetMode function of the underlying flash driver.
Ea_GetStatus	Service to return the Status.
Fee_GetJobResult	Service to call the GetJobResult function of the underlying flash driver.
Fee_InvalidateBlock	Service to invalidate a logical block.
Ea_Read	Reads Length bytes of block Blocknumber at offset BlockOffset into the buffer DataBufferPtr.
Ea_EraseImmediateBlock	Erases the block BlockNumber.
Fee_EraseImmediateBlock	Service to erase a logical block.
Ea_Write	Writes the contents of the DataBufferPtr to the block BlockNumber.

### 8.6.2 Optional Interfaces

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

**MemIf048:**

<i>API function</i>	<i>Description</i>
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 kind of interfaces is not fixed because they are configurable.

There are no configurable interfaces for this module.

## 9 Sequence diagrams

Refer to the specifications of the memory abstraction modules.

## 10 Configuration specification

### 10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture
  - AUTOSAR ECU Configuration Specification
- This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel 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.

#### 10.1.2 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

#### 10.1.3 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

<b>Label</b>	<b>Description</b>
x	The configuration parameter shall be of configuration class <i>Pre-compile time</i> .
--	The configuration parameter shall never be of configuration class <i>Pre-compile time</i> .

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

<b>Label</b>	<b>Description</b>
x	The configuration parameter shall be of configuration class <i>Link time</i> .
--	The configuration parameter shall never be of configuration class <i>Link time</i> .

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

<b>Label</b>	<b>Description</b>
x	The configuration parameter shall be of configuration class <i>Post Build</i> and no specific implementation is required.
L	<i>Loadable</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and only one configuration parameter set resides in the ECU.
M	<i>Multiple</i> - the configuration parameter shall be of configuration class <i>Post Build</i> and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class <i>Post Build</i> .

## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 0.

### 10.2.1 Variants

No variants specified.

### 10.2.2 MemIf

<b>Module Name</b>	<b>MemIf</b>
<b>Module Description</b>	Configuration of the MemIf (Memory Abstraction Interface) module.

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
MemIfGeneral	1	Configuration of the memory abstraction interface (Memif) module.

### 10.2.3 MemIfGeneral

<b>SWS Item</b>	<b>MemIf025 :</b>
<b>Container Name</b>	MemIfGeneral{MemIf_Configuration}
<b>Description</b>	Configuration of the memory abstraction interface (Memif) module.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	--
<b>Name</b>	MemIfDevErrorDetect {MEMIF_DEV_ERROR_DETECT}

<b>Description</b>	Pre-processor switch to enable and disable development error detection. true: Development error detection enabled. false: Development error detection disabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	--		
<b>Name</b>	MemIfNumberOfDevices {MEMIF_NUMBER_OF_DEVICES}		
<b>Description</b>	Concrete number of underlying memory abstraction modules.		
<b>Multiplicity</b>	1		
<b>Type</b>	DerivedIntegerParamDef		
<b>Range</b>	1 .. 255		
<b>Default value</b>	--		
<b>calculationFormula</b>	--		
<b>calculationLanguage</b>	informal		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	<b>MemIf032 :</b>		
<b>Name</b>	MemIfVersionInfoApi {MEMIF_VERSION_INFO_API}		
<b>Description</b>	Pre-processor switch to enable / disable the API to read out the modules version information. true: Version info API enabled. false: Version info API disabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

**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 [10] Figure 4.1 and Figure 7.1).

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

## 11 Changes during SWS Improvements by Technical Office

### 11.1 Deleted SWS Items

None

### 11.2 Replaced SWS Items

None

### 11.3 Changed SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">MemIf022</a>	RfC #17179: Requirement extended to encompass the broadcast identifier.
<a href="#">MemIf015</a> , <a href="#">MemIf016</a> , <a href="#">MemIf021</a> , <a href="#">MemIf031</a> , <a href="#">MemIf049</a> , <a href="#">MemIf050</a> , <a href="#">MemIf051</a> , <a href="#">MemIf052</a> , <a href="#">MemIf053</a> , <a href="#">MemIf054</a> , <a href="#">MemIf055</a> , <a href="#">MemIf056</a> , <a href="#">MemIf057</a>	RfC #19597: Wording of requirements adapted after chapter 8 has been generated.

### 11.4 Added SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">MemIf037</a>	UML Model linking of the imported types
<a href="#">MemIf038</a>	UML Model linking of MemIf_SetMode
<a href="#">MemIf039</a>	UML Model linking of MemIf_Read
<a href="#">MemIf040</a>	UML Model linking of MemIf_Write
<a href="#">MemIf041</a>	UML Model linking of MemIf_Cancel
<a href="#">MemIf042</a>	UML Model linking of MemIf_GetStatus
<a href="#">MemIf043</a>	UML Model linking of MemIf_GetJobResult
<a href="#">MemIf044</a>	UML Model linking of MemIf_InvalidateBlock
<a href="#">MemIf045</a>	UML Model linking of MemIf_GetVersionInfo
<a href="#">MemIf046</a>	UML Model linking of MemIf_EraseImmediateBlock
<a href="#">MemIf047</a>	UML Model linking of the mandatory interfaces
<a href="#">MemIf048</a>	UML Model linking of the optional interfaces
<a href="#">MemIf049</a>	Extracted during the UML model linking
<a href="#">MemIf050</a>	Extracted during the UML model linking
<a href="#">MemIf051</a>	Extracted during the UML model linking
<a href="#">MemIf052</a>	Extracted during the UML model linking
<a href="#">MemIf053</a>	Extracted during the UML model linking
<a href="#">MemIf054</a>	Extracted during the UML model linking
<a href="#">MemIf055</a>	Extracted during the UML model linking
<a href="#">MemIf056</a>	Extracted during the UML model linking
<a href="#">MemIf057</a>	Extracted during the UML model linking