

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

<b>Document Version</b>	1.4.0
<b>Document Status</b>	Final
<b>Part of Release</b>	4.0
<b>Revision</b>	3

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
03.11.2011	1.4.0	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Module short name changed</li><li>• Consistency checking reformulated</li></ul>
19.10.2010	1.3.0	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Check for NULL pointer added</li><li>• Inter module checks detailed</li></ul>
03.12.2009	1.2.0	AUTOSAR Administration	<ul style="list-style-type: none"><li>• Description of return values extended</li><li>• File include structure changed</li><li>• Variant requirement description added</li><li>• Legal disclaimer revised</li></ul>
23.06.2008	1.1.1	AUTOSAR Administration	Legal disclaimer revised
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

## Disclaimer

This specification and the material contained in it, as released by AUTOSAR is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only.

For any other purpose, no part of the specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Advice for users

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 .....	5
2	Acronyms and abbreviations .....	6
3	Related documentation.....	7
3.1	Input documents.....	7
3.2	Related standards and norms .....	7
4	Constraints and assumptions .....	8
4.1	Limitations .....	8
4.2	Applicability to car domains.....	8
5	Dependencies to other modules.....	9
5.1	File structure .....	9
5.1.1	Code file structure .....	9
5.1.2	Header file structure.....	9
6	Requirements traceability .....	11
7	Functional specification .....	22
7.1	Version check.....	22
7.2	Error classification .....	22
7.3	Error detection.....	22
7.4	Error notification .....	22
8	API specification .....	24
8.1	Imported types.....	24
8.1.1	Standard types .....	24
8.2	Type definitions .....	24
8.2.1	MemIf_StatusType .....	24
8.2.2	MemIf_JobResultType .....	24
8.2.3	MemIf_ModeType .....	25
8.3	Function definitions .....	25
8.3.1	MemIf_SetMode.....	26
8.3.2	MemIf_Read.....	26
8.3.3	MemIf_Write.....	27
8.3.4	MemIf_Cancel .....	27
8.3.5	MemIf_GetStatus .....	28
8.3.6	MemIf_GetJobResult .....	28
8.3.7	MemIf_InvalidateBlock .....	29
8.3.8	MemIf_GetVersionInfo .....	29
8.3.9	MemIf_EraseImmediateBlock .....	30
8.4	Call-back notifications .....	30
8.5	Scheduled functions .....	30
8.6	Expected Interfaces.....	31
8.6.1	Mandatory Interfaces .....	31
8.6.2	Optional Interfaces .....	31
8.6.3	Configurable interfaces .....	31

9	Sequence diagrams .....	32
10	Configuration specification .....	33
10.1	How to read this chapter .....	33
10.1.1	Configuration and configuration parameters .....	33
10.1.2	Containers .....	33
10.1.3	Specification template for configuration parameters .....	33
10.2	Containers and configuration parameters .....	34
10.2.1	Variants .....	34
10.2.2	MemIf .....	34
10.2.3	MemIfGeneral .....	34
10.3	Published Information .....	35
11	Changes w.r.t. Release 3.0 version .....	36
11.1	Deleted SWS Items .....	36
11.2	Replaced SWS Items .....	36
11.3	Changed SWS Items .....	36
11.4	Added SWS Items .....	36
12	Not applicable requirements .....	37

## 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).

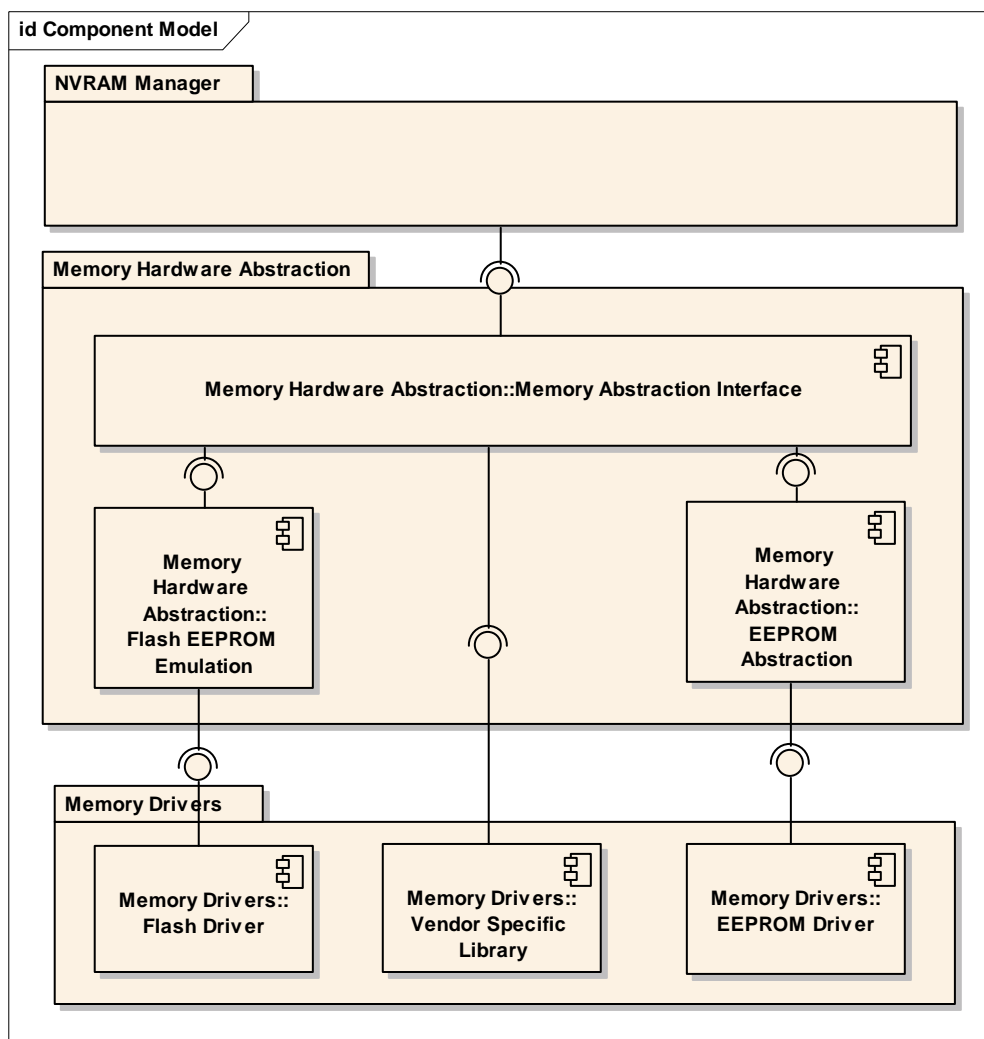


Figure 1: Module overview of memory hardware abstraction layer

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	<p>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.</p> <p><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></p>
Slow Mode	<p>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.</p> <p><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></p>
Vendor specific library	A vendor specific library is an ICC-2 implementation of the FEE/FLS and EA/EEP modules respectively. It provides the same upper layer interface (API) and functionality as the corresponding ICC-3 implementation.

## 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] General Requirements on SPAL  
AUTOSAR\_SRS\_SPALGeneral.pdf
- [5] Requirements on Memory Hardware Abstraction Layer  
AUTOSAR\_SRS\_MemoryHWAbstractionLayer.doc
- [6] Specification of Development Error Tracer  
AUTOSAR\_SWS\_DevelopmentErrorTracer.pdf

### 3.2 Related standards and norms

- [7] Specification of NVRAM Manager  
AUTOSAR\_SWS\_NVRAMManager.doc
- [8] Specification of Flash EEPROM Emulation  
AUTOSAR\_SWS\_FlashEEPROMEmulation.pdf
- [9] Specification of EEPROM Abstraction  
AUTOSAR\_SWS\_EEPROMAbstraction.pdf

## **4 Constraints and assumptions**

### **4.1 Limitations**

No limitations.

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

No restrictions.



## 5 Dependencies to other modules

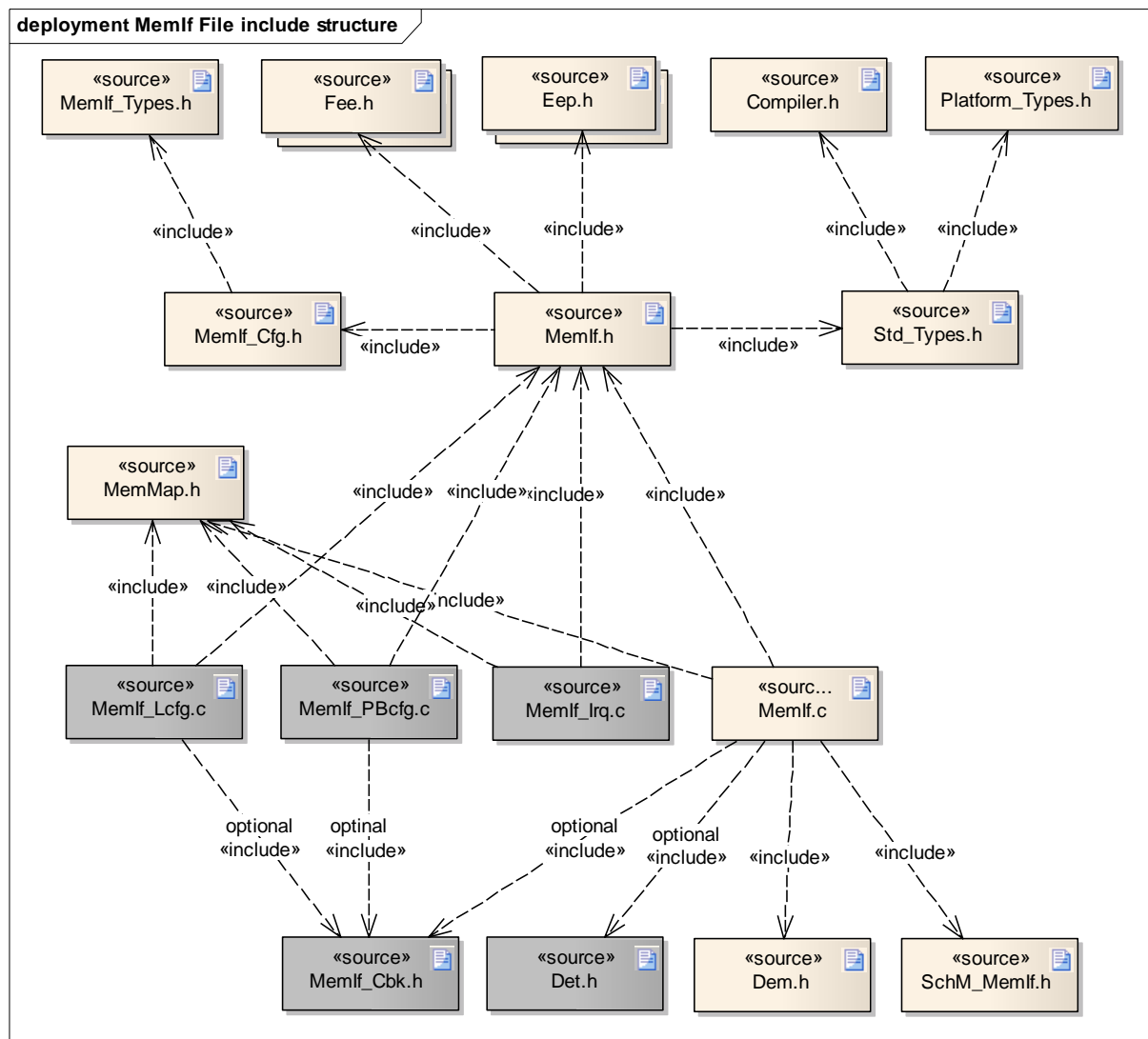
### 5.1 File structure

#### 5.1.1 Code file structure

**[MemIf033]** 「The code file structure shall not be defined within this specification. 」()

#### 5.1.2 Header file structure

**[MemIf002]** 「The file include structure shall be as follows: 」 (BSW00381, BSW00383; BSW00384, BSW00346, BSW158, BSW00435. BSW00436, BSW00301)



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

*Note: Files shown in grey are optional (depending on implementation / configuration), other files are mandatory.*

*Note: Only header files of such memory abstraction modules (FEE, EA) which are present in a specific implementation / configuration shall be included by the Memory Abstraction Interface.*

**[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

Requirement	Satisfied by
-	MemIf038
-	MemIf037
-	MemIf040
-	MemIf046
-	MemIf009
-	MemIf035
-	MemIf044
-	MemIf043
-	MemIf063
-	MemIf041
-	MemIf042
-	MemIf057
-	MemIf047
-	MemIf024
-	MemIf034
-	MemIf033
-	MemIf039
-	MemIf030
-	MemIf011
-	MemIf010
BSW00300	MemIf999
BSW00301	MemIf002
BSW00302	MemIf999
BSW00304	MemIf999
BSW00306	MemIf999
BSW00307	MemIf999
BSW00308	MemIf999
BSW00309	MemIf999
BSW00312	MemIf999
BSW00314	MemIf999
BSW00321	MemIf999
BSW00323	MemIf022
BSW00324	MemIf999
BSW00325	MemIf999
BSW00326	MemIf999
BSW00327	MemIf006
BSW00328	MemIf999

BSW00330	MemIf999
BSW00331	MemIf059
BSW00333	MemIf999
BSW00334	MemIf999
BSW00336	MemIf999
BSW00337	MemIf006
BSW00338	MemIf007, MemIf058
BSW00339	MemIf999
BSW0034	MemIf999
BSW00341	MemIf999
BSW00342	MemIf999
BSW00343	MemIf999
BSW00346	MemIf002
BSW00347	MemIf999
BSW00348	MemIf999
BSW00350	MemIf007, MemIf058
BSW00353	MemIf999
BSW00355	MemIf999
BSW00358	MemIf999
BSW00359	MemIf999
BSW00360	MemIf999
BSW00361	MemIf999
BSW00369	MemIf059
BSW00370	MemIf999
BSW00371	MemIf999
BSW00373	MemIf999
BSW00375	MemIf999
BSW00376	MemIf999
BSW00378	MemIf999
BSW00380	MemIf999
BSW00381	MemIf002
BSW00383;BSW00384	MemIf002
BSW00385	MemIf048
BSW00386	MemIf007, MemIf006, MemIf058, MemIf023
BSW00387	MemIf048
BSW00398	MemIf999
BSW00399	MemIf999
BSW004	MemIf061, MemIf062
BSW00400	MemIf999
BSW00401	MemIf999
BSW00404	MemIf999
BSW00405	MemIf999

BSW00406	MemIf999
BSW00407	MemIf045
BSW00409	MemIf029
BSW00412	MemIf999
BSW00413	MemIf999
BSW00414	MemIf999
BSW00415	MemIf999
BSW00416	MemIf999
BSW00417	MemIf999
BSW00420	MemIf999
BSW00421	MemIf999
BSW00422	MemIf999
BSW00423	MemIf999
BSW00424	MemIf999
BSW00425	MemIf999
BSW00426	MemIf999
BSW00427	MemIf999
BSW00428	MemIf999
BSW00429	MemIf999
BSW00431	MemIf999
BSW00432	MemIf999
BSW00433	MemIf999
BSW00434	MemIf999
BSW00435.BSW00436	MemIf002
BSW005	MemIf999
BSW006	MemIf999
BSW007	MemIf999
BSW009	MemIf999
BSW010	MemIf999
BSW101	MemIf999
BSW12056	MemIf999
BSW12057	MemIf999
BSW12058	MemIf999
BSW12059	MemIf999
BSW12060	MemIf999
BSW12062	MemIf999
BSW12063	MemIf999
BSW12064	MemIf999
BSW12067	MemIf999
BSW12068	MemIf999
BSW12069	MemIf999
BSW12075	MemIf999

BSW12077	MemIf999
BSW12078	MemIf019, MemIf020
BSW12081	MemIf999
BSW12092	MemIf999
BSW12125	MemIf999
BSW12129	MemIf999
BSW12155	MemIf999
BSW12163	MemIf999
BSW12263	MemIf999
BSW12265	MemIf999
BSW12267	MemIf999
BSW12448	MemIf023
BSW12461	MemIf999
BSW12462	MemIf999
BSW12463	MemIf999
BSW14019	MemIf017
BSW14020	MemIf018
BSW14021	MemIf018, MemIf019, MemIf020, MemIf022
BSW14022	MemIf017
BSW14023	MemIf061, MemIf062, MemIf022
BSW14025	MemIf019, MemIf020
BSW157	MemIf999
BSW158	MemIf002
BSW159	MemIf999
BSW160	MemIf999
BSW161	MemIf999
BSW162	MemIf999
BSW164	MemIf999
BSW167	MemIf061, MemIf062
BSW168	MemIf999
BSW170	MemIf999
BSW172	MemIf999

## 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="#">MemIf061</a> , <a href="#">MemIf062</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	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	<a href="#">MemIf002</a>
[BSW00387] Specify the configuration class of callback function	<a href="#">MemIf048</a>
[BSW00388] Introduce containers	<a href="#">MemIf025</a> , <a href="#">MemIf026</a>
[BSW00389] Containers shall have names	<a href="#">MemIf025</a> , <a href="#">MemIf026</a>
[BSW00390] Parameter content shall be unique within the module	<a href="#">MemIf025</a>
[BSW00391] Parameter shall have unique names	<a href="#">MemIf025</a>
[BSW00392] Parameters shall have a type	<a href="#">MemIf025</a>
[BSW00393] Parameters shall have a range	<a href="#">MemIf025</a>
[BSW00394] Specify the scope of the parameters	<a href="#">MemIf025</a>
[BSW00395] List the required parameters (per parameter)	<a href="#">MemIf025</a>
[BSW00396] Configuration classes	<a href="#">MemIf025</a>
[BSW00397] Pre-compile-time parameters	<a href="#">MemIf025</a>
[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	<a href="#">MemIf026</a>
[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	<a href="#">MemIf045</a> , <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="#">MemIf058</a>
[BSW00369] Do not return development error codes via API	<a href="#">MemIf059</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="#">MemIf061</a> , <a href="#">MemIf062</a>
[BSW00409] Header files for production code error IDs	<a href="#">MemIf029</a>
[BSW00385] List possible error notifications	<a href="#">MemIf048</a>
[BSW00386] Configuration for detecting an error	<a href="#">MemIf006</a> , <a href="#">MemIf007</a> , <a href="#">MemIf023</a> , <a href="#">MemIf058</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	Not applicable



within MCAL	(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	Not applicable (this module does not implement any ISRs)
[BSW00325] Runtime of interrupt service routines	Not 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	Not 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>
[BSW00335] Status values naming convention	Chapter 8.2.1
[BSW00350] Development error detection keyword	<a href="#">MemIf007</a> , <a href="#">MemIf058</a> , <a href="#">MemIf025</a>
[BSW00408] Configuration parameter naming convention	<a href="#">MemIf025</a>
[BSW00410] Compiler switches shall have defined values	<a href="#">MemIf025</a>
[BSW00411] Get version info keyword	<a href="#">MemIf025</a>
[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)
[BSW00435] Module Header File Structure for the	<a href="#">MemIf002</a>

Basic Software Scheduler	
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	<a href="#">MemIf002</a>
[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="#">MemIf059</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	Not applicable (this module has no direct hardware access)
[BSW12462] Provide settings for register initialization	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>
[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="#">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="#">MemIf061</a> , <a href="#">MemIf062</a> , <a href="#">MemIf022</a>
BSW14024 Preserving of timing behavior	Not applicable
BSW14025 Efficient implementation	<a href="#">MemIf019</a> , <a href="#">MemIf020</a>

## 7 Functional specification

### 7.1 Version check

**[MemIf061]** 「All pre-compile time configuration parameters shall be checked statically (at least during compile time) for correctness.」 (BSW167, BSW004, BSW14023)

**[MemIf062]** 「The MemIf module shall perform inter module checks to avoid integration of incompatible files: all included header files shall be checked by pre-processing directives. The MemIf module shall thereby verify that `<MODULENAME>_AR_RELEASE_MAJOR_VERSION` and `<MODULENAME>_AR_RELEASE_MINOR_VERSION` are identical to the expected values, where `<MODULENAME>` is the module abbreviation of the external module, which provides the included header file. If the values are not identical, an error shall be raised at compile time. 」 (BSW167, BSW004, BSW14023)

### 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`. 」 (BSW00409)

**[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
API service called with NULL pointer argument	Development	MEMIF_E_PARAM_POINTER	0x02

」 (BSW00337, BSW00386, BSW00327)

### 7.3 Error detection

**[MemIf007]** 「The detection of development errors shall be configurable (on/off) at pre-compile time. The switch `MEMIF_DEV_ERROR_DETECT` (see chapter 10) shall activate or deactivate the detection of all development errors.」 (BSW00338, BSW00386, BSW00350)

### 7.4 Error notification

**[MemIf058]** 「Detected development errors shall be reported to the *Det\_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch `MEMIF_DEV_ERROR_DETECT` is set (see chapter 10). 」  
(BSW00338, BSW00386, BSW00350)

**[MemIf059]** 「The error codes shall not be used as return values for the called function. 」(BSW00369, BSW00331)

## 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] ⌈

Module	Imported Type
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋()

### 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..MEMIF\_NUMBER\_OF\_DEVICES-1. ⌋()

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

#### 8.2.2 MemIf\_JobResultType

<b>Name:</b>	MemIf_JobResultType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	MEMIF_JOB_OK	The job has been finished successfully.
	MEMIF_JOB_FAILED	The job has not been finished successfully.
	MEMIF_JOB_PENDING	The job has not yet been finished.
	MEMIF_JOB_CANCELED	The job has been canceled.



	MEMIF_BLOCK_INCONSISTENT	The requested block is inconsistent, it may contain corrupted data.
	MEMIF_BLOCK_INVALID	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.	

### 8.2.3 MemIf\_ModeType

<b>Name:</b>	MemIf_ModeType	
<b>Type:</b>	Enumeration	
<b>Range:</b>	MEMIF_MODE_SLOW	The underlying memory abstraction modules and drivers are working in slow mode.
	MEMIF_MODE_FAST	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.	

## 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. 」 (BSW14019, BSW14022)

**[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. 」 (BSW14020, BSW14021)

**[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. 」 (BSW12078, BSW14021, BSW14025)

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. 」 (BSW12078, BSW14021, BSW14025)

*Note: 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 functions of the Memory Abstraction Interface API shall check the parameter `DeviceIndex` for being an existing device or the broadcast identifier within the module's services. ⌋ (BSW00323, BSW14021, BSW14023)

**[MemIf023]** ⌈ The functions of the Memory Abstraction Interface API shall report detected errors attributed to an illegal parameter `DeviceIndex` to the Development Error Tracer (DET) with the error code `MEMIF_E_PARAM_DEVICE` and the called service shall not be executed. ⌋ (BSW00386, BSW12448)

**[MemIf024]** ⌈ If a called function of the Memory Abstraction Interface API has detected an error attributed to an illegal parameter `DeviceIndex` and has a return value, it shall be set as follows:

<code>MemIf_GetStatus:</code>	<code>MEMIF_UNINIT</code>
<code>MemIf_GetJobResult:</code>	<code>MEMIF_JOB_FAILED</code>
All other functions:	<code>E_NOT_OK. ⌋()</code>

### 8.3.1 MemIf\_SetMode

**[MemIf038]** ⌈

<b>Service name:</b>	MemIf_SetMode		
<b>Syntax:</b>	<pre>void MemIf_SetMode(     MemIf_ModeType Mode )</pre>		
<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>	Invokes the "SetMode" functions of all underlying memory abstraction modules.		

⌋()

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 In case development error detection is enabled for the Memory Abstraction Interface and a development error is detected according to MemIf022 the function shall return E_NOT_OK else it shall return the value of the called function of the underlying module.
<b>Description:</b>	Invokes the "Read" function of the underlying memory abstraction module selected by the parameter DeviceIndex.

⌋()

### 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 In case development error detection is enabled for the Memory Abstraction Interface and a development error is detected according to MemIf022 the function shall return E_NOT_OK else it shall return the value of the called function of the underlying module.
<b>Description:</b>	Invokes the "Write" function of the underlying memory abstraction module selected by the parameter DeviceIndex.

⌋()

### 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>	Invokes the "Cancel" function of the underlying memory abstraction module selected by the parameter DeviceIndex.

]()

### 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>	Invokes the "GetStatus" function of the underlying memory abstraction module selected by the parameter DeviceIndex.

]()

**[MemIf035]** If the function MemIf\_GetStatus is called with the device index denoting a broadcast to all configured devices (MEMIF\_BROADCAST\_ID), the Memory Abstraction Interface 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 ]()

**Note:** The special "broadcast" device ID in the call to MemIf\_GetStatus is used to query whether all devices are idle in order to shut down the ECU.

### 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	In case development error detection is enabled for the Memory Abstraction Interface and a development error is detected according to MemIf022 the function shall return E_JOB_FAILED else it shall return the value of the called function of the underlying module.
<b>Description:</b>	Invokes the "GetJobResult" function of the underlying memory abstraction module selected by the parameter DeviceIndex.	

### 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	In case development error detection is enabled for the Memory Abstraction Interface and a development error is detected according to MemIf022 the function shall return E_NOT_OK else it shall return the value of the called function of the underlying module.
<b>Description:</b>	Invokes the "InvalidateBlock" function of the underlying memory abstraction module selected by the parameter DeviceIndex.	

### 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.	

⌋ (BSW00407)

**[MemIf057]** ⌈ Configurations: The function MemIf\_GetVersionInfo is only available if enabled by the pre-processor switch MEMIF\_VERSION\_INFO\_API. ⌋ ()

**[MemIf063]** ⌈ If development error detection for the module MemIf is enabled: the function MemIf\_GetVersionInfo shall raise the development error MemIf\_E\_PARAM\_POINTER if the argument is a NULL pointer and return without any action. ⌋ ()

### 8.3.9 MemIf\_EraseImmediateBlock

**[MemIf046]** ⌈

<b>Service name:</b>	MemIf_EraseImmediateBlock	
<b>Syntax:</b>	<pre>Std_ReturnType MemIf_EraseImmediateBlock(     uint8 DeviceIndex,     uint16 BlockNumber )</pre>	
<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	In case development error detection is enabled for the Memory Abstraction Interface and a development error is detected according to MemIf022 the function shall return E_NOT_OK else it shall return the value of the called function of the underlying module.
<b>Description:</b>	Invokes the "EraseImmediateBlock" function of the underlying memory abstraction module selected by the parameter DeviceIndex.	

⌋ ()

## 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]

API function	Description
Ea_Cancel	Cancels the ongoing asynchronous operation.
Ea_EraseImmediateBlock	Erases the block BlockNumber.
Ea_GetJobResult	Service to return the JobResult.
Ea_GetStatus	Service to return the Status.
Ea_InvalidateBlock	Invalidates the block BlockNumber.
Ea_Read	Reads Length bytes of block Blocknumber at offset BlockOffset into the buffer DataBufferPtr.
Ea_SetMode	Sets the mode.
Ea_Write	Writes the contents of the DataBufferPtr to the block BlockNumber.
Fee_Cancel	Service to call the cancel function of the underlying flash driver.
Fee_EraseImmediateBlock	Service to erase a logical block.
Fee_GetJobResult	Service to query the result of the last accepted job issued by the upper layer software.
Fee_GetStatus	Service to return the status.
Fee_InvalidateBlock	Service to invalidate a logical block.
Fee_Read	Service to initiate a read job.
Fee_SetMode	Service to call the Fls_SetMode function of the underlying flash driver.
Fee_Write	Service to initiate a write job.

()

### 8.6.2 Optional Interfaces

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

[MemIf048]

API function	Description
Det_ReportError	Service to report development errors.

(BSW00387, BSW00385)

### 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

Label	Description
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

Label	Description
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

Label	Description
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 8.

### 10.2.1 Variants

**[MemIf060]** **┌** VARIANT-PRE-COMPILE

Only parameters with "Pre-compile time" configuration are allowed in this variant. **└** ()

### 10.2.2 MemIf

<b>SWS Item</b>	<b>MemIf025_Conf :</b>
<b>Module Name</b>	<i>MemIf</i>
<b>Module Description</b>	Configuration of the MemIf (Memory Abstraction Interface) module.

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

### 10.2.3 MemIfGeneral

<b>SWS Item</b>	<b>MemIf034_Conf :</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>MemIf035_Conf :</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>	EcucBooleanParamDef		
<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>MemIf033_Conf :</b>		
<b>Name</b>	MemIfNumberOfDevices {MEMIF_NUMBER_OF_DEVICES}		
<b>Description</b>	Concrete number of underlying memory abstraction modules. Calculation Formula: Count number of configured EA and FEE modules.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	1 .. 255		
<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>MemIf032_Conf :</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>	EcucBooleanParamDef		
<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

**[MemIf64]** The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1]]. ()

Additional module-specific published parameters are listed below if applicable.

## 11 Changes w.r.t. Release 3.0 version

### 11.1 Deleted SWS Items

<b>SWS Item</b>	<b>Rationale</b>
MEMIF008	Requirement was confusing
MEMIF001	Requirement ID deleted, statement left in (information only).
MEMIF036	Requirement already covered by <a href="#">MemIf035</a>

### 11.2 Replaced SWS Items

<b>SWS Item</b>	<b>replaced by SWS Item</b>	<b>Rationale</b>
MEMIF005	<a href="#">MemIf061</a> , <a href="#">MemIf062</a>	Requirment split to make atomic

### 11.3 Changed SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">MEMIF038</a> , <a href="#">MEMIF039</a> , <a href="#">MEMIF040</a> , <a href="#">MEMIF041</a> , <a href="#">MEMIF042</a> , <a href="#">MEMIF043</a> , <a href="#">MEMIF044</a> , <a href="#">MEMIF046</a>	Formulation improved
MEMIF002	File include structure changed, second note added.
<a href="#">MEMIF039</a> , <a href="#">MEMIF040</a> , <a href="#">MEMIF043</a> , <a href="#">MEMIF044</a> , <a href="#">MEMIF046</a>	Description of return value improved.
<a href="#">MEMIF062</a>	Inter module checks detailed
<a href="#">MEMIF062</a>	Inter module checks detailed – not only if DET is active.
<a href="#">MEMIF062</a>	
<a href="#">MEMIF062</a>	

### 11.4 Added SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">MEMIF060</a>	Variant requirement description added
<a href="#">MEMIF001_PI</a>	Rework of Published Information
<a href="#">MEMIF063</a>	

## 12 Not applicable requirements

**[MemIf999]** [These requirements are not applicable to this specification. ]

(BSW0034, BSW00404, BSW00405, BSW159, BSW170, BSW00380, BSW00412, BSW00398, BSW00399, BSW00400, BSW00375, BSW101, BSW00416, BSW00406, BSW168, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW00336, BSW00339, BSW00421, BSW00422, BSW00420, BSW00417, BSW161, BSW162, BSW00324, BSW005, BSW00415, BSW164, BSW00325, BSW00326, BSW00342, BSW00343, BSW160, BSW007, BSW00300, BSW00413, BSW00347, BSW00307, BSW00373, BSW00314, BSW00370, BSW00348, BSW00353, BSW00361, BSW00302, BSW00328, BSW00312, BSW006, BSW00304, BSW00355, BSW00378, BSW00306, BSW00308, BSW00309, BSW00371, BSW00358, BSW00414, BSW00376, BSW00359, BSW00360, BSW00330, BSW009, BSW00401, BSW172, BSW010, BSW00333, BSW00321, BSW00341, BSW00334, BSW12263, BSW12056, BSW12267, BSW12057, BSW12125, BSW12163, BSW12058, BSW12059, BSW12060, BSW12461, BSW12462, BSW12463, BSW12062, BSW12068, BSW12069, BSW157, BSW12155, BSW12063, BSW12075, BSW12129, BSW12064, BSW12067, BSW12077, BSW12092, BSW12265, BSW12081)