

|                                   |                                 |
|-----------------------------------|---------------------------------|
| <b>Document Title</b>             | Specification of Memory Mapping |
| <b>Document Owner</b>             | AUTOSAR                         |
| <b>Document Responsibility</b>    | AUTOSAR                         |
| <b>Document Identification No</b> | 128                             |
| <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>   |
| 01.12.2011                     | 1.4.0          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Consistent naming pattern for memory allocation keywords is introduced</li> <li>• Refine definition the &lt;PREFIX&gt; part in memory allocation keywords</li> </ul>   |
| 03.11.2010                     | 1.3.0          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• ECU Configuration Parameters for MemMap defined</li> <li>• Define generation of MemMap header files</li> <li>• New standardised Memory Allocation Keywords for new initialisation policy CLEARED added</li> <li>• Refinement of &lt;SIZE&gt; suffix of Memory Allocation Keywords to &lt;ALIGNMENT&gt; suffix,</li> <li>• Clarify link MetaModel attribute values,               <ul style="list-style-type: none"> <li>○ Define MemorySectionType and SectionInitializationPolicy for the standardised Memory Allocation Keywords</li> <li>○ Define that &lt;NAME&gt; used for Memory Allocation Keywords is the MemorySection shortName</li> </ul> </li> <li>• Application hint for usage of INLINE and LOCAL_INLINE added</li> <li>• Handling structs, arrays and unions redefined</li> </ul> |

| <b>Document Change History</b> |                |                        |  |
|--------------------------------|----------------|------------------------|--|
| <b>Date</b>                    | <b>Version</b> | <b>Changed by</b>      | <b>Change Description</b>  |
| 04.12.2009                     | 1.2.0          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• Typo errors are corrected throughout the document</li> <li>• Memory Mapping section has been extended for application SWC</li> <li>• Common Published information has been updated</li> <li>• Legal disclaimer revised</li> </ul>   |
| 23.06.2008                     | 1.1.1          | AUTOSAR Administration | Legal disclaimer revised   |
| 12.12.2007                     | 1.1.0          | AUTOSAR Administration | <ul style="list-style-type: none"> <li>• In MEMMAP004, all size postfixes for memory segment names were listed, the keyword 'BOOLEAN' was added, taking into account the particular cases where boolean data need to be mapped in a particular segment.</li> <li>• In MEMMAP004 and MEMMAP021, tables are defining the mapping segments associated to #pragmas instructions, adding some new segments to take into account some implementation cases</li> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul> |
| 13.02.2006                     | 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 .....   | 6  |
| 2        | Acronyms and abbreviations .....   | 8  |
| 3        | Related documentation.....   | 9  |
| 3.1      | Input documents.....   | 9  |
| 3.2      | Related standards and norms .....  | 10 |
| 4        | Constraints and assumptions .....  | 11 |
| 4.1      | Limitations .....  | 11 |
| 4.2      | Applicability to car domains.....  | 11 |
| 4.3      | Applicability to safety related environments .....   | 11 |
| 5        | Dependencies to other modules.....   | 12 |
| 5.1      | File structure .....   | 12 |
| 5.1.1    | Code file structure .....  | 12 |
| 5.1.2    | Header file structure.....   | 12 |
| 6        | Requirements traceability .....  | 14 |
| 7        | Analysis .....   | 23 |
| 7.1      | Memory allocation of variables.....  | 23 |
| 7.2      | Memory allocation of constant variables .....  | 24 |
| 7.3      | Memory allocation of code .....  | 25 |
| 8        | Functional specification .....   | 27 |
| 8.1      | General issues .....   | 27 |
| 8.2      | Mapping of variables and code .....  | 28 |
| 8.2.1    | Requirements on implementations using memory mapping header files<br>for BSW Modules and Software Components ..... | 28 |
| 8.2.2    | Requirements on memory mapping header files.....   | 35 |
| 8.3      | Examples .....   | 39 |
| 8.3.1    | Code Section.....  | 39 |
| 8.3.2    | Fast Variable Section .....  | 42 |
| 8.3.3    | Code Section in ICC2 cluster .....   | 47 |
| 9        | API specification.....   | 50 |
| 10       | Sequence diagrams.....   | 51 |
| 11       | Configuration specification.....   | 52 |
| 11.1     | How to read this chapter .....   | 52 |
| 11.1.1   | Configuration and configuration parameters .....   | 52 |
| 11.1.2   | Variants.....  | 52 |
| 11.1.3   | Containers.....  | 53 |
| 11.1.4   | Specification template for configuration parameters .....  | 53 |
| 11.2     | Containers and configuration parameters .....  | 55 |
| 11.2.1   | Variants.....  | 55 |
| 11.2.1.1 | VARIANT-PRE-COMPILE.....   | 55 |
| 11.2.2   | MemMap .....   | 55 |

|        |                                    |    |
|--------|------------------------------------|----|
| 11.2.3 | MemMapAddressingModeSet .....      | 55 |
| 11.2.4 | MemMapAddressingMode .....         | 58 |
| 11.2.5 | MemMapAllocation.....              | 59 |
| 11.2.6 | MemMapGenericMapping.....          | 60 |
| 11.2.7 | MemMapSectionSpecificMapping ..... | 61 |
| 11.3   | Published Information.....         | 62 |
| 12     | Not applicable requirements .....  | 63 |

## 1 Introduction and functional overview

This document specifies mechanisms for the mapping of code and data to specific memory sections via memory mapping files. For many ECUs and microcontroller platforms it is of utmost necessity to be able to map code, variables and constants module wise to specific memory sections. Selection of important use cases:

### ***Avoidance of waste of RAM***

If different variables (8, 16 and 32 bit) are used within different modules on a 32 bit platform, the linker will leave gaps in RAM when allocating the variables in the RAM. This is because the microcontroller platform requires a specific alignment of variables and some linkers do not allow an optimization of variable allocation.

This wastage of memory can be circumvented if the variables are mapped to specific memory sections depending on their size. This minimizes unused space in RAM.

### ***Usage of specific RAM properties***

Some variables (e.g. the RAM mirrors of the NVRAM Manager) must not be initialized after a power-on reset. It shall be possible to map them to a RAM section that is not initialized after a reset.

For some variables (e.g. variables that are accessed via bit masks) it improves both performance and code size if they are located within a RAM section that allows for bit manipulation instructions of the compiler. Those RAM sections are usually known as 'Near Page' or 'Zero Page'.

### ***Usage of specific ROM properties***

In large ECUs with external flash memory there is the requirement to map modules with functions that are called very often to the internal flash memory that allows for fast access and thus higher performance. Modules with functions that are called rarely or that have lower performance requirements are mapped to external flash memory that has slower access.

***Usage of the same source code of a module for boot loader and application***

If a module shall be used both in boot loader and application, it is necessary to allow the mapping of code and data to different memory sections.

A mechanism for mapping of code and data to memory sections that is supported by all compilers listed in chapter 3.1 is the usage of pragmas. As pragmas are very compiler specific, a mechanism that makes use of those pragmas in a standardized way has to be specified.

***Support of Memory Protection***

The usage of hardware memory protection requires a separation of the modules variables into different memory areas. Internal variables are mapped into protected memory, buffers for data exchange are mapped into unprotected memory.

***Support of partitioning***

In some cases it is necessary to separate partition assigned memory.

Therefore an additional separation of the module variables into different memory (partition-)areas is needed if the BSW Module shall support a split over several Partitions.

## 2 Acronyms and abbreviations

| <b>Abbreviation /<br/>Acronym:</b> | <b>Description:</b>       |
|------------------------------------|---------------------------|
| BSW                                | Basic Software            |
| ISR                                | Interrupt Service Routine |
| NVRAM                              | Non-Volatile RAM          |



## 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] AUTOSAR Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Basic Software Module Description Template,  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [5] Software Component Template,  
AUTOSAR\_TPS\_SoftwareComponentTemplate.pdf
- [6] ECU Configuration Specification  
AUTOSAR\_SWS\_ECU\_StateManager.pdf
- [7] Methodology  
AUTOSAR\_TR\_Methodology.pdf
- [8] Cosmic C Cross Compiler User's Guide for Motorola MC68HC12, V4.5
- [9] ARM ADS compiler manual
- [10] GreenHills MULTI for V850 V4.0.5:  
Building Applications for Embedded V800, V4.0, 30.1.2004
- [11] TASKING for ST10 V8.5:  
C166/ST10 v8.5 C Cross-Compiler User's Manual, V5.16  
C166/ST10 v8.5 C Cross-Assembler, Linker/Locator, Utilities User's Manual,  
V5.16
- [12] Wind River (Diab Data) for PowerPC Version 5.2.1:  
Wind River Compiler for Power PC - Getting Started, Edition 2, 8.5.2004  
Wind River Compiler for Power PC - User's Guide, Edition 2, 11.5.2004
- [13] TASKING for TriCore TC1796 V2.0R1:  
TriCore v2.0 C Cross-Compiler, Assembler, Linker User's Guide, V1.2

- [14] Metrowerks CodeWarrior 4.0 for Freescale HC9S12X/XGATE (V5.0.25):  
Motorola HC12 Assembler, 2.6.2004  
Motorola HC12 Compiler, 2.6.2004  
Smart Linker, 2.4.2004

### **3.2 Related standards and norms**

Not applicable.

## **4 Constraints and assumptions**

### **4.1 Limitations**

During specification of abstraction and validation of concept the compilers listed in chapter 3.1 have been considered. If any other compiler requires keywords that cannot be mapped to the mechanisms described in this specification this compiler will not be supported by AUTOSAR. In this case, the compiler vendor has to adapt its compiler.

The concepts described in this document do only apply to C compilers. C++ is not in scope of this version.

A dedicated pack-control of structures is not supported. Hence global set-up passed via compiler / linker parameters has to be used.

A dedicated alignment control of code, variables and constants is not supported. Hence affected objects shall be assigned to different sections or a global setting passed via compiler / linker parameters has to be used.

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

No restrictions.

### **4.3 Applicability to safety related environments**

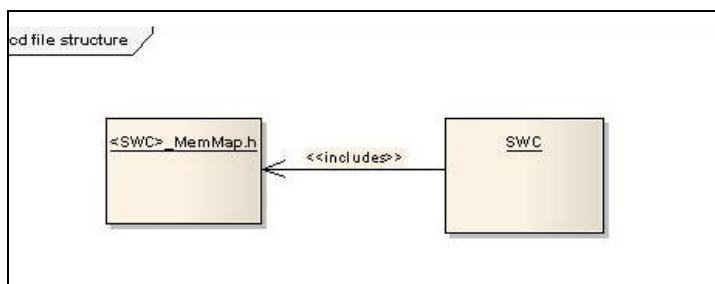
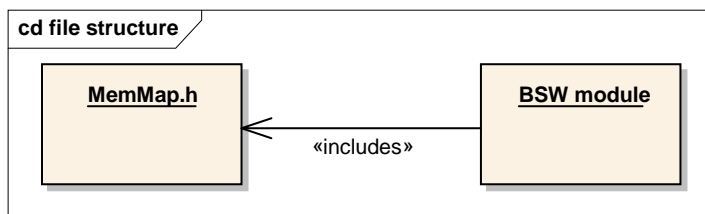
No restrictions. The memory mapping files do not implement any functionality, only symbols and macros.

## 5 Dependencies to other modules

**[MEMMAP020]** 「The SWS Memory Mapping is applicable for each AUTOSAR basic software module and software component. Therefore the implementation of memory mapping files shall fulfill the implementation and configuration specific needs of each software module in a specific build scenario. See also Recommendation A; [MEMMAP003](#), [MEMMAP018](#) and [MEMMAP001](#).」(BSW00384)

### 5.1 File structure

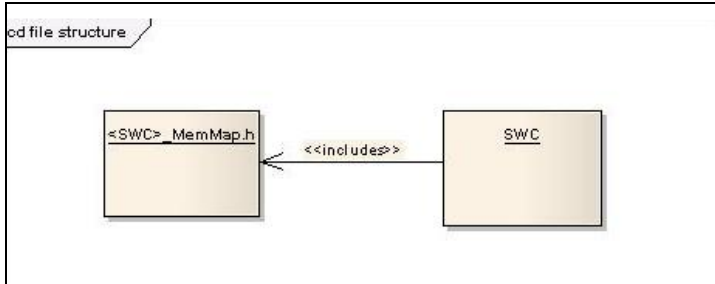
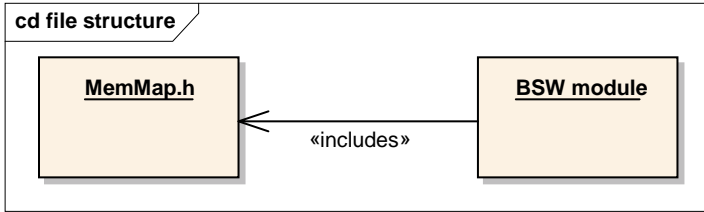
#### 5.1.1 Code file structure



#### 5.1.2 Header file structure

**[MEMMAP028]** 「The Memory Mapping shall provide one BSW memory mapping header file '<MemMap.h'.」()

**[MEMMAP029]** 「For each software component type which is part of the input configuration a software component type specific memory mapping header file '<SWC>\_MemMap.h' shall be provided by the Memory Mapping.」()



## 6 Requirements traceability

| Requirement | Satisfied by  |
|-------------|---|
| -           | MEMMAP021   |
| -           | MEMMAP029   |
| -           | MEMMAP027   |
| -           | MEMMAP022   |
| -           | MEMMAP018   |
| -           | MEMMAP016   |
| -           | MEMMAP023   |
| -           | MEMMAP015   |
| -           | MEMMAP026   |
| -           | MEMMAP030   |
| -           | MEMMAP028   |
| BSW00300    | MEMMAP999   |
| BSW00301    | MEMMAP999   |
| BSW00302    | MEMMAP999   |
| BSW00304    | MEMMAP999   |
| BSW00306    | MEMMAP010, MEMMAP011, MEMMAP013, MEMMAP006, MEMMAP005, MEMMAP003, MEMMAP007 |
| BSW00307    | MEMMAP999   |
| BSW00308    | MEMMAP999   |
| BSW00309    | MEMMAP999   |
| BSW00310    | MEMMAP999   |
| BSW00312    | MEMMAP999   |
| BSW00314    | MEMMAP999   |
| BSW00323    | MEMMAP999   |
| BSW00324    | MEMMAP999   |
| BSW00325    | MEMMAP999   |
| BSW00326    | MEMMAP999   |
| BSW00327    | MEMMAP999   |
| BSW00328    | MEMMAP005, MEMMAP001  |
| BSW00329    | MEMMAP999   |
| BSW00330    | MEMMAP999   |
| BSW00331    | MEMMAP999   |
| BSW00333    | MEMMAP999   |
| BSW00334    | MEMMAP999   |
| BSW00335    | MEMMAP999   |

|          |           |
|----------|-----------|
| BSW00336 | MEMMAP999 |
| BSW00337 | MEMMAP999 |
| BSW00338 | MEMMAP999 |
| BSW00339 | MEMMAP999 |
| BSW00341 | MEMMAP999 |
| BSW00342 | MEMMAP999 |
| BSW00343 | MEMMAP999 |
| BSW00344 | MEMMAP999 |
| BSW00345 | MEMMAP999 |
| BSW00346 | MEMMAP999 |
| BSW00347 | MEMMAP999 |
| BSW00348 | MEMMAP999 |
| BSW00350 | MEMMAP999 |
| BSW00353 | MEMMAP999 |
| BSW00355 | MEMMAP999 |
| BSW00357 | MEMMAP999 |
| BSW00358 | MEMMAP999 |
| BSW00359 | MEMMAP999 |
| BSW00360 | MEMMAP999 |
| BSW00361 | MEMMAP002 |
| BSW00369 | MEMMAP999 |
| BSW00370 | MEMMAP999 |
| BSW00371 | MEMMAP999 |
| BSW00373 | MEMMAP999 |
| BSW00375 | MEMMAP999 |
| BSW00377 | MEMMAP999 |
| BSW00378 | MEMMAP999 |
| BSW00380 | MEMMAP999 |
| BSW00381 | MEMMAP999 |
| BSW00383 | MEMMAP999 |
| BSW00384 | MEMMAP020 |
| BSW00385 | MEMMAP999 |
| BSW00386 | MEMMAP999 |
| BSW00387 | MEMMAP999 |
| BSW00388 | MEMMAP999 |
| BSW00389 | MEMMAP999 |
| BSW00390 | MEMMAP999 |
| BSW00391 | MEMMAP999 |
| BSW00392 | MEMMAP999 |
| BSW00393 | MEMMAP999 |

|          |   |
|----------|---|
| BSW00394 | MEMMAP999   |
| BSW00395 | MEMMAP999   |
| BSW00396 | MEMMAP999   |
| BSW00397 | MEMMAP999   |
| BSW00398 | MEMMAP999   |
| BSW00399 | MEMMAP999   |
| BSW004   | MEMMAP999   |
| BSW00400 | MEMMAP999   |
| BSW00401 | MEMMAP999   |
| BSW00404 | MEMMAP999   |
| BSW00405 | MEMMAP999   |
| BSW00406 | MEMMAP999   |
| BSW00407 | MEMMAP999   |
| BSW00408 | MEMMAP999   |
| BSW00409 | MEMMAP999   |
| BSW00410 | MEMMAP999   |
| BSW00411 | MEMMAP999   |
| BSW00412 | MEMMAP999   |
| BSW00413 | MEMMAP999   |
| BSW00414 | MEMMAP999   |
| BSW00415 | MEMMAP999   |
| BSW00416 | MEMMAP999   |
| BSW00417 | MEMMAP999   |
| BSW00419 | MEMMAP999   |
| BSW00420 | MEMMAP999   |
| BSW00421 | MEMMAP999   |
| BSW00422 | MEMMAP999   |
| BSW00423 | MEMMAP999   |
| BSW00424 | MEMMAP999   |
| BSW00425 | MEMMAP999   |
| BSW00426 | MEMMAP999   |
| BSW00427 | MEMMAP999   |
| BSW00428 | MEMMAP999   |
| BSW00429 | MEMMAP999   |
| BSW00431 | MEMMAP999   |
| BSW00432 | MEMMAP999   |
| BSW00433 | MEMMAP999   |
| BSW00434 | MEMMAP999   |
| BSW005   | MEMMAP999   |
| BSW006   | MEMMAP010, MEMMAP011, MEMMAP013, MEMMAP006, MEMMAP005, MEMMAP003, MEMMAP007 |



|          |           |
|----------|-----------|
| BSW00605 | MEMMAP999 |
| BSW007   | MEMMAP999 |
| BSW009   | MEMMAP999 |
| BSW010   | MEMMAP999 |
| BSW101   | MEMMAP999 |
| BSW158   | MEMMAP999 |
| BSW159   | MEMMAP999 |
| BSW160   | MEMMAP999 |
| BSW161   | MEMMAP999 |
| BSW162   | MEMMAP999 |
| BSW164   | MEMMAP999 |
| BSW167   | MEMMAP999 |
| BSW168   | MEMMAP999 |
| BSW170   | MEMMAP999 |
| BSW171   | MEMMAP999 |
| BSW172   | MEMMAP999 |

Document: AUTOSAR General Requirements on Basic Software Modules

| <b>Requirement</b>   | <b>Satisfied by</b>   |
|--|---|
| [BSW00344] Reference to link-time configuration            | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00404] Reference to post build time configuration      | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00405] Reference to multiple configuration sets        | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00345] Pre-compile-time configuration                  | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW159] Tool-based configuration                          | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW167] Static configuration checking                     | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW171] Configurability of optional functionality         | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW170] Data for reconfiguration of AUTOSAR SW-Components | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00380] Separate C-Files for configuration parameters   | Not applicable<br>(Memory Mapping is specific per build scenario) |

| <b>Requirement</b>  | <b>Satisfied by</b>   |
|---|---|
| [BSW00419] Separate C-Files for pre-compile time configuration parameters     | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00381] Separate configuration header file for pre-compile time parameters | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00412] Separate H-File for configuration parameters                       | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00383] List dependencies of configuration files                           | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00384] List dependencies to other modules                                 | <a href="#">MEMMAP020</a>   |
| [BSW00387] Specify the configuration class of callback function               | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00388] Introduce containers   | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00389] Containers shall have names  | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00390] Parameter content shall be unique within the module                | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00391] Parameter shall have unique names                                  | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00392] Parameters shall have a type                                       | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00393] Parameters shall have a range                                      | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00394] Specify the scope of the parameters                                | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00395] List the required parameters (per parameter)                       | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00396] Configuration classes  | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00397] Pre-compile-time parameters  | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00398] Link-time parameters   | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00399] Loadable Post-build time parameters                                | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00400] Selectable Post-build time parameters                              | Not applicable<br>(Memory Mapping is specific per build scenario) |
| [BSW00402] Published information  | <a href="#">MEMMAP019</a>   |
| [BSW00375] Notification of wake-up reason                                     | Not applicable<br>(Memory Mapping is not a BSW module)            |

| <b>Requirement</b>  | <b>Satisfied by</b>  |
|---|--|
| [BSW101] Initialization interface   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00416] Sequence of Initialization   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00406] Check module initialization  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW168] Diagnostic Interface of SW components  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00407] Function to read out published parameters  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces                               | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00424] BSW main processing function task allocation   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00425] Trigger conditions for schedulable objects   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00426] Exclusive areas in BSW modules   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00427] ISR description for BSW modules  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00428] Execution order dependencies of main processing functions  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00429] Restricted BSW OS functionality access   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00431] The BSW Scheduler module implements task bodies  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00433] Calling of main processing functions   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00434] The Schedule Module shall provide an API for exclusive areas   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00336] Shutdown interface   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00337] Classification of errors   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00338] Detection and Reporting of development errors  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00369] Do not return development error codes via API  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00339] Reporting of production relevant error status  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00421] Reporting of production relevant error events  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00422] Debouncing of production relevant error status   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00420] Production relevant error event rate detection   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00417] Reporting of Error Events by Non-Basic Software  | Not applicable,<br>(Memory Mapping does not report errors) |
| [BSW00323] API parameter checking   | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW004] Version check  | Not applicable<br>(Memory Mapping is not a BSW module)     |
| [BSW00409] Header files for production code error IDs   | Not applicable<br>(Memory Mapping is not a BSW module)     |

| <b>Requirement</b>   | <b>Satisfied by</b>  |
|--|--|
| [BSW00385] List possible error notifications                       | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00386] Configuration for detecting an error                    | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW161] Microcontroller abstraction                               | Not applicable<br>(non-functional requirement)                         |
| [BSW162] ECU layout abstraction                                    | Not applicable<br>(non-functional requirement)                         |
| [BSW00324] Do not use HIS I/O Library                              | Not applicable<br>(non-functional requirement)                         |
| [BSW005] No hard coded horizontal interfaces within MCAL           | Not applicable<br>(non-functional requirement)                         |
| [BSW00415] User dependent include files                            | Not applicable<br>(non-functional requirement)                         |
| [BSW164] Implementation of interrupt service routines              | Not applicable<br>(non-functional requirement)                         |
| [BSW00325] Runtime of interrupt service routines                   | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00326] Transition from ISRs to OS tasks                        | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00342] Usage of source code and object code                    | Not applicable<br>(non-functional requirement)                         |
| [BSW00343] Specification and configuration of time                 | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW160] Human-readable configuration data                         | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW007] HIS MISRA C   | Not applicable,<br>(Memory Mapping is the C-language extension header) |
| [BSW00300] Module naming convention                                | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00413] Accessing instances of BSW modules                      | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00347] Naming separation of different instances of BSW drivers | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00305] Self-defined data types naming convention               | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00307] Global variables naming convention                      | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00310] API naming convention                                   | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00373] Main processing function naming convention              | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00327] Error values naming convention                          | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00335] Status values naming convention                         | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00350] Development error detection keyword                     | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00408] Configuration parameter naming convention               | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00410] Compiler switches shall have defined values             | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00411] Get version info keyword                                | Not applicable<br>(Memory Mapping is not a BSW module)                 |
| [BSW00346] Basic set of module files                               | Not applicable<br>(Memory Mapping is not a BSW module)                 |

| <b>Requirement</b>   | <b>Satisfied by</b>   |
|--|---|
| [BSW158] Separation of configuration from implementation               | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00314] Separation of interrupt frames and service routines         | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00370] Separation of callback interface from API                   | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00348] Standard type header  | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00353] Platform specific type header                               | Not applicable<br>(Memory Mapping is a C-language extension header)   |
| [BSW00361] Compiler specific language extension header                 | <a href="#">MEMMAP002</a>   |
| [BSW00301] Limit imported information                                  | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00302] Limit exported information                                  | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00328] Avoid duplication of code                                   | supported by:<br><a href="#">MEMMAP001</a> , <a href="#">MEMMAP005</a>  |
| [BSW00312] Shared code shall be reentrant                              | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW006] Platform independency   | supported by:<br><a href="#">MEMMAP010</a> , <a href="#">MEMMAP004</a> ,<br><a href="#">MEMMAP003</a> , <a href="#">MEMMAP005</a> ,<br><a href="#">MEMMAP006</a> , <a href="#">MEMMAP007</a> ,<br><a href="#">MEMMAP011</a> , <a href="#">MEMMAP013</a> |
| [BSW00357] Standard API return type                                    | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00377] Module specific API return types                            | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00304] AUTOSAR integer data types                                  | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00355] Do not redefine AUTOSAR integer data types                  | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00378] AUTOSAR boolean type  | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00306] Avoid direct use of compiler and platform specific keywords | supported by:<br><a href="#">MEMMAP010</a> , <a href="#">MEMMAP004</a> ,<br><a href="#">MEMMAP003</a> , <a href="#">MEMMAP005</a> ,<br><a href="#">MEMMAP006</a> , <a href="#">MEMMAP007</a> ,<br><a href="#">MEMMAP011</a> , <a href="#">MEMMAP013</a> |
| [BSW00308] Definition of global data                                   | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00309] Global data with read-only constraint                       | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00371] Do not pass function pointers via API                       | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00358] Return type of <code>init()</code> functions                | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00414] Parameter of <code>init</code> function                     | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00414] Parameter of <code>init</code> function                     | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00359] Return type of callback functions                           | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00360] Parameters of callback functions                            | Not applicable<br>(Memory Mapping is not a BSW module)  |
| [BSW00329] Avoidance of generic interfaces                             | Not applicable  |

| <b>Requirement</b>   | <b>Satisfied by</b>                                    |
|--|--|
|  | (Memory Mapping is not a BSW module)                   |
| [BSW00330] Usage of macros / inline functions instead of functions         | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW00331] Separation of error and status values                           | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW009] Module User Documentation   | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW00401] Documentation of multiple instances of configuration parameters | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW172] Compatibility and documentation of scheduling strategy            | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW010] Memory resource documentation                                     | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW00333] Documentation of callback function context                      | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW00374] Module vendor identification                                    | <a href="#">MEMMAP019</a>                              |
| [BSW00379] Module identification   | <a href="#">MEMMAP019</a>                              |
| [BSW003] Version identification  | <a href="#">MEMMAP019</a>                              |
| [BSW00318] Format of module version numbers                                | <a href="#">MEMMAP019</a>                              |
| [BSW00321] Enumeration of module version numbers                           | <a href="#">MEMMAP019</a>                              |
| [BSW00341] Microcontroller compatibility documentation                     | Not applicable<br>(Memory Mapping is not a BSW module) |
| [BSW00334] Provision of XML file   | Not applicable<br>(Memory Mapping is not a BSW module) |



## 7 Analysis

This chapter does not contain requirements. It just gives an overview to used keywords and their syntax within different compilers. This analysis is required for a correct and complete specification of methods and keywords.

### 7.1 Memory allocation of variables

Compiler analysis for starting/stopping a memory section for variables:

| <b>Compiler</b>  | <b>Required syntax</b>   |
|------------------|--|
| Cosmic, S12X     | <p>Initialized variables:<br/>#pragma section {name}<br/>#pragma section {}</p> <p>Non Initialized variables:<br/>#pragma section [name]<br/>#pragma section []</p>  |
| Metrowerks, S12X | <p>#pragma DATA_SEG" (&lt;Modif&gt; &lt;Name&gt;   "DEFAULT")<br/>&lt;Modif&gt;: Some of the following strings may be used:<br/>SHORT, __SHORT_SEG,<br/>DIRECT, __DIRECT_SEG,<br/>NEAR, __NEAR_SEG,<br/>FAR, __FAR_SEG,<br/>DPAGE, __DPAGE_SEG,<br/>RPAGE, __RPAGE_SEG</p> <p>Pragma shall be used in definition and declaration.</p>  |
| Tasking, ST10    | <p>#pragma class mem=name<br/>#pragma combine mem=ctype<br/>#pragma align mem=atype<br/>#pragma noclear</p> <p>#pragma default_attributes<br/>#pragma clear</p> <p>atype is one of the following align types:<br/>B Byte alignment<br/>W Word alignment<br/>P Page alignment<br/>S Segment alignment<br/>C PEC addressable<br/>I IRAM addressable</p> <p>ctype is one of the following combine types:<br/>L private ('Local')<br/>P Public<br/>C Common<br/>G Global<br/>S Sysstack<br/>U Usrstack<br/>A address Absolute section AT constant address<br/>(decimal, octal or hexadecimal number)</p> |
| Tasking, TC1796  | <p>#pragma pack 0 / 2<br/>packing of structs. Shall be visible at type declaration</p>   |

| Compiler          | Required syntax   |
|-------------------|---|
|                   | <pre>#pragma section type "string" #pragma noclear  #pragma clear  #pragma for_extern_data_use_memory #pragma for_initialized_data_use_memory #pragma for_uninitialized_data_use_memory</pre>           |
| GreenHills, V850  | <pre>#pragma align (n) #pragma alignvar (n) #pragma ghs section sect="name" #pragma ghs section sect =default Section Keyword: data, sdata, tdata, zdata, bss, sbss, zbss</pre>                         |
| ADS, ST30         | <pre>#pragma arm section [sort_type[="name"]] [,sort_type="name"]* sort_type="rwdata, zidata alignment control via key words: __packed, __align()</pre>   |
| DIABDATA, MPC5554 | <pre>#pragma section class_name [init_name] [uninit_name] [address_mode] [access] #pragma section class_name Pragma shall be used before declaration.  class_name for variables: BSS, DATA, SDATA</pre> |

## 7.2 Memory allocation of constant variables

Compiler analysis for starting/stopping a memory section for constant variables:

| Compiler         | Required syntax  |
|------------------|--|
| Cosmic, S12X     | <pre>#pragma section const {name} #pragma section const {}</pre>   |
| Metrowerks, S12X | <pre>#pragma CONST_SEG" (&lt;Modif&gt; &lt;Name&gt;   "DEFAULT") &lt;Modif&gt;: Some of the following strings may be used: PPAGE, __PPAGE_SEG, GPAGE, __GPAGE_SEG Pragma shall be used in definition and declaration.</pre>  |
| Tasking, ST10    | <pre>#pragma class mem=name #pragma align mem=atype #pragma combine mem=ctype #pragma default_attributes  atype is one of the following align types: B Byte alignment W Word alignment P Page alignment S Segment alignment C PEC addressable I IRAM addressable  ctype is one of the following combine types: L private ('Local') P Public C Common</pre> |



| <b>Compiler</b>   | <b>Required syntax</b>   |
|-------------------|--|
|                   | G Global<br>S Sysstack<br>U Usrstack<br>A address Absolute section AT constant address<br>(decimal, octal or hexadecimal number)   |
| Tasking, TC1796   | <pre>#pragma pack 0 / 2</pre> Packing of structs. Shall be visible at type declaration<br><br><pre>#pragma section type "string"</pre> <pre>#pragma for_constant_data_use_memory</pre>   |
| GreenHills, V850  | <pre>#pragma ghs section sect="name"</pre> <pre>#pragma ghs section sect =default</pre> Section Keyword: rodata, rozdata, rosdata  |
| ADS, ST30         | <pre>#pragma arm section [sort_type[["name"]]</pre> <pre>[,sort_type="name"]*</pre> <pre>sort_type="rodata</pre><br>alignment control via key words:<br><pre>__packed, __align()</pre>   |
| DIABDATA, MPC5554 | <pre>#pragma section class_name [init_name]</pre> <pre>[uninit_name] [address_mode] [access]</pre> <pre>#pragma section class_name</pre> Pragma shall be used before declaration.<br><br>class_name for constant variables:<br>CONST, SCONST, STRING |

### 7.3 Memory allocation of code

Compiler analysis for starting/stopping a memory section for code::

| <b>Compiler</b>  | <b>Required syntax</b>  |
|------------------|---|
| Cosmic, S12X     | <pre>#pragma section ( name )</pre> <pre>#pragma section ( )</pre>  |
| Metrowerks, S12X | <pre>#pragma CODE_SEG" (&lt;Modif&gt; &lt;Name&gt;   "DEFAULT")</pre> <Modif>: Some of the following strings may be used:<br>DIRECT, __DIRECT_SEG,<br>NEAR, __NEAR_SEG,<br>CODE, __CODE_SEG,<br>FAR, __FAR_SEG,<br>PPAGE, __PPAGE_SEG,<br>PIC, __PIC_SEG<br>Pragma shall be used in definition and declaration. |
| Tasking, ST10    | <pre>#pragma class mem=name</pre> <pre>#pragma combine mem=ctype</pre> <pre>#pragma default_attributes</pre><br>ctype is one of the following combine types:<br>L private ('Local')<br>P Public<br>C Common<br>G Global<br>S Sysstack<br>U Usrstack<br>A address Absolute section AT constant address           |

| <b>Compiler</b>   | <b>Required syntax</b>  |
|-------------------|---|
| Tasking, TC1796   | <pre>#pragma section code "string" #pragma section code_init #pragma section const_init #pragma section vector_init #pragma section data_overlay #pragma section type[="name"] #pragma section all</pre>        |
| GreenHills, V850  | <pre>#pragma ghs section sect="name" #pragma ghs section sect =default</pre> <p><b>Section Keyword:</b> text</p>  |
| ADS, ST30         | <pre>#pragma arm section [sort_type[["name"]] [,sort_type="name"]*  sort_type="code"</pre>  |
| DIABDATA, MPC5554 | <pre>#pragma section class_name [init_name] [uninit_name] [address_mode] [access] #pragma section class_name</pre> <p><b>Pragma shall be used before declaration.</b></p> <p>class_name for code:<br/> CODE</p> |

## 8 Functional specification

### 8.1 General issues

The memory mapping files include the compiler and linker specific keywords for memory allocation into header and source files. These keywords control the assignment of variables and functions to specific sections. Thereby implementations are independent from compiler and microcontroller specific properties.

The assignment of the sections to dedicated memory areas / address ranges is not the scope of the memory mapping file and is typically done via linker control files.

**[MEMMAP001]** 「For each build scenario (e.g. Boot loader, ECU Application) an own set of memory mapping files has to be provided.」(BSW00328)

**[MEMMAP002]** 「The memory mapping file name shall be 'MemMap.h' for basic software modules and "<SWC>\_MemMap.h" for software components where <SWC> is the name of the software component type.」(BSW00361)

**[MEMMAP010]** 「If a compiler/linker does not require or support requisite functionality of SWS Memory Mapping, the Memory Allocation Keyword defines shall be undefined without further effect.」(BSW006, BSW00306)

For instance:

```
#ifdef EEP_START_SEC_VAR_CLEARED_16
    #undef EEP_START_SEC_VAR_CLEARED_16
#endif
```

As described in [MEMMAP029](#) the number of files depends on the number of `SwComponentTypes` in the input configuration. To determine the number of `MemorySections` the applicable `SwcImplementations` have to be known. These are described in an AUTOSAR environment with the `SwcToImplMapping` in the `SystemMapping` and / or via ECU Configuration values `RteImplementationRef` in a `RteSwComponentType` container.

Knowing the `SwcImplementations` provides as well the number of `MemorySections` which have to be identified for [MEMMAP027](#). For more details about the content of a `SwcImplementation` see document [5] and [4]

Further on the total number of used `MemorySections` depends as well on the number of used BSW modules. These can be determined by the M1 instance of the `EcucValueCollection` which refers to the `MemMap's EcucModuleConfigurationValues`. This `EcucValueCollection` refers as well to `EcucModuleConfigurationValues` of other Bsw Modules which refer again to `BswImplementations` via `moduleDescription` references.

Knowing the `BswImplementations` provides as well the number of `MemorySections` which have to be identified for [MEMMAP026](#). For more details about the content of a `BswImplementation` see document [4]

In [7] further information is provided how Memory Mapping is used in the AUTOSAR Methodology.

## 8.2 Mapping of variables and code

### 8.2.1 Requirements on implementations using memory mapping header files for BSW Modules and Software Components

Recommendation A:

Each AUTOSAR basic software module and software component shall support the configuration of at least the following different memory types as described in Tabelle 8-1 and Tabelle 8-2.

It is allowed to add module specific sections as they are mapped and thus are configurable within the module's configuration file.

The shortcut '<ALIGNMENT>' means the variable alignment. In order to avoid memory gaps in the allocation variables are allocated according their size. Possible ALIGNMENT postfixes are

BOOLEAN, used for variables and constants of size 1 bit

8, used for variables and constants which have to be aligned to 8 bit. For instance used for variables and constants of size 8 bit or used for composite data types: arrays, structs and unions containing elements of maximum 8 bits.

16, used for variables and constants which have to be aligned to 16 bit. For instance used for variables and constants of size 16 bit or used for composite data types: arrays, structs and unions containing elements of maximum 16 bits

32, used for variables and constants which have to be aligned to 32 bit. For instance used for variables and constants of size 32 bit or used for composite data types: arrays, structs and unions containing elements of maximum 32 bits.

UNSPECIFIED, used for variables, constants, structure, array and unions when SIZE (alignment) does not fit the criteria of 8,16 or 32 bit.

For instance used for variables and constants of unknown size

In case structures and unions, it shall be allowed to use an alignment larger than the bit size of the elements. For instance to facilitate copy instruction a structure may have minimum 2 byte alignment, even if members are byte aligned. In this case, it should be possible to use alignment 16 bit instead of 8 bit for this structure.

Please note that the values 8BIT, 16BIT, 32BIT are changed to 8, 16, 32 in order to reach a harmonization with Meta Model attributes. These values are classified as deprecated.

The shortcut '<INIT\_POLICY>' means the initialization policy of variables. Possible INIT\_POLICY postfixes are:

- NO\_INIT, used for variables that are never cleared and never initialized.
- CLEARED, used for variables that are cleared to zero after every reset.
- POWER\_ON\_CLEARED, used for variables that are cleared to zero only after power on reset.
- INIT, used for variables that are initialized with values after every reset.
- POWER\_ON\_INIT, used for variables that are initialized with values only after power on reset.

**[MEMMAP022]** The keywords to be used before inclusion of the memory mapping header file shall use the templates <PREFIX>\_START\_SEC\_<NAME> or <PREFIX>\_STOP\_SEC\_<NAME>

Where:

- <PREFIX> is composed according <snp>[\_<vi>\_<ai>] for basic software modules where
  - <snp> is the *Section Name Prefix* which shall be the Module Abbreviation from the BSW Module list (e.g. 'EEP' or 'CAN') in upper case letters of the BSW module. For the generation of the MemMap.h file following rules apply:
    - <snp> shall be the `BswModuleDescription`'s `shortName` converted in upper case letters if **no** `SectionNamePrefix` is defined for the `MemorySection`.
    - <snp> shall be the symbol of the `SectionNamePrefix` associated to the `MemorySection` if **a** `SectionNamePrefix` is defined for the `MemorySection`.
  - <vi> is the `vendorId` of the BSW module
  - <ai> is the `vendorApiInfix` of the BSW module

The sub part in squared brackets [\_<vi>\_<ai>] is omitted if no `vendorApiInfix` is defined for the Basic Software Module which indicates that it does not use multiple instantiation.
- <PREFIX> is the `shortName` of the software component type for software components (case sensitive)
- <NAME> is the `shortName` of the `MemorySection` described in Basic Software Module Description or a Software Component Description (case sensitive) if the `MemorySection` has **no** `symbol` attribute defined.
- <NAME> is the symbol of the `MemorySection` described in Basic Software Module Description or a Software Component Description (case sensitive) if the `MemorySection` has **a** `symbol` attribute defined. ]()

Please note if the Memory Allocation Keywords shall appear in capital letters in the code the related Memory Sections in the BSWMD or SWC Descriptions have to be named with capital letters.

| <b>Memory Section Type / Section Initialization Policy</b> | <b>Syntax of Memory Allocation Keyword</b>  | <b>Comments</b>   |
|--|---|---|
| VAR /<br><INIT_POLICY>                                     | <PREFIX>_START_SEC_VAR_<INIT_POLICY>_<ALIGNMENT><br><PREFIX>_STOP_SEC_VAR_<INIT_POLICY>_<ALIGNMENT>                   | To be used for all global or static variables.  |
| VAR /<br><INIT_POLICY>                                     | <PREFIX>_START_SEC_VAR_FAST_<INIT_POLICY>_<ALIGNMENT><br><PREFIX>_STOP_SEC_VAR_FAST_<INIT_POLICY>_<ALIGNMENT>         | To be used for all global or static variables that have at least one of the following properties: <ul style="list-style-type: none"> <li>• accessed bitwise</li> <li>• frequently used</li> <li>• high number of accesses in source code</li> </ul> Some platforms allow the use of bit instructions for variables located in this specific RAM area as well as shorter addressing instructions. This saves code and runtime. |
| VAR /<br><INIT_POLICY>                                     | <PREFIX>_START_SEC_VAR_SLOW_<INIT_POLICY>_<ALIGNMENT><br><PREFIX>_STOP_SEC_VAR_SLOW_<INIT_POLICY>_<ALIGNMENT>         | To be used for all infrequently accessed global or static variables.  |
| VAR /<br><INIT_POLICY>                                     | <PREFIX>_START_SEC_INTERNAL_VAR_<INIT_POLICY>_<ALIGNMENT><br><PREFIX>_STOP_SEC_INTERNAL_VAR_<INIT_POLICY>_<ALIGNMENT> | To be used for global or static variables those are accessible from a calibration tool.   |
|  |   |   |
| VAR /<br>NO-INIT   | <PREFIX>_START_SEC_VAR_NOINIT_<ALIGNMENT><br><PREFIX>_STOP_SEC_VAR_NOINIT_<ALIGNMENT>                                 | To be used for all global or static variables that are never initialized. This section is <b>DEPRECATED</b> and shall not be used in future development. <sup>1</sup>   |
| VAR /<br>POWER-ON-INIT                                     | <PREFIX>_START_SEC_VAR_POWER_ON_INIT_<ALIGNMENT><br><PREFIX>_STOP_SEC_VAR_POWER_ON_INIT_<ALIGNMENT>                   | To be used for all global or static variables that are initialized with values only after power on reset. This section is <b>DEPRECATED</b> and shall not be used in future development. <sup>1</sup>   |

<sup>1</sup> This section was replaced by the generic definition of <PREFIX>\_START\_SEC\_VAR\_<INIT\_POLICY>\_<ALIGNMENT>.

| Memory Section Type / Section Initialization Policy | Syntax of Memory Allocation Keyword  | Comments   |
|---|--|--|
| VAR / POWER-ON-CLEARED                              | <pre>&lt;PREFIX&gt;_START_SEC_VAR_POWER_ON_CLEARED_&lt;ALIGNMENT&gt;</pre> <pre>&lt;PREFIX&gt;_STOP_SEC_VAR_POWER_ON_CLEARED_&lt;ALIGNMENT&gt;</pre> | <p>To be used for all global or static variables that are cleared to zero only after power on reset.</p> <p>This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>1</sup></p>   |
| VAR / INIT  | <pre>&lt;PREFIX&gt;_START_SEC_VAR_&lt;ALIGNMENT&gt;</pre> <pre>&lt;PREFIX&gt;_STOP_SEC_VAR_&lt;ALIGNMENT&gt;</pre>                                   | <p>To be used for global or static variables that are initialized with values after every reset.</p> <p>This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>1</sup></p>   |
| VAR / CLEARED                                       | <pre>&lt;PREFIX&gt;_START_SEC_VAR_CLEARED_&lt;ALIGNMENT&gt;</pre> <pre>&lt;PREFIX&gt;_STOP_SEC_VAR_CLEARED_&lt;ALIGNMENT&gt;</pre>                   | <p>To be used for global or static variables that are cleared to zero after every reset (the normal case).</p> <p>This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>1</sup></p>   |
| VAR / INIT  | <pre>&lt;PREFIX&gt;_START_SEC_VAR_FAST_&lt;ALIGNMENT&gt;</pre> <pre>&lt;PREFIX&gt;_STOP_SEC_VAR_FAST_&lt;ALIGNMENT&gt;</pre>                         | <p>To be used for all global or static variables that have at least one of the following properties:</p> <ul style="list-style-type: none"> <li>• accessed bitwise</li> <li>• frequently used</li> <li>• high number of accesses in source code</li> </ul> <p>Some platforms allow the use of bit instructions for variables located in this specific RAM area as well as shorter addressing instructions. This saves code and runtime. Variables are initialized with values after every reset (the normal case).</p> <p>This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>2</sup></p> |
| VAR / CLEARED                                       | <pre>&lt;PREFIX&gt;_START_SEC_VAR_FAST_CLEARED_&lt;ALIGNMENT&gt;</pre>   | <p>To be used for all global or static variables that have at</p>  |

<sup>2</sup> This section was replaced by the generic definition of `<PREFIX>_START_SEC_VAR_FAST_<INIT_POLICY>_<ALIGNMENT>`.

| Memory Section Type / Section Initialization Policy | Syntax of Memory Allocation Keyword   | Comments   |
|---|---|--|
|   | <code>&lt;PREFIX&gt;_STOP_SEC_VAR_FAST_CLEARED_&lt;ALIGNMENT&gt;</code>   | <p>least one of the following properties:</p> <ul style="list-style-type: none"> <li>• accessed bitwise</li> <li>• frequently used</li> <li>• high number of accesses in source code</li> </ul> <p>Some platforms allow the use of bit instructions for variables located in this specific RAM area as well as shorter addressing instructions. This saves code and runtime. Variables are initialized to zero after every reset (the normal case). This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>2</sup></p> |
| VAR / INIT  | <code>&lt;PREFIX&gt;_START_SEC_INTERNAL_VAR_&lt;ALIGNMENT&gt;</code><br><code>&lt;PREFIX&gt;_STOP_SEC_INTERNAL_VAR_&lt;ALIGNMENT&gt;</code>                       | <p>To be used for global or static variables that are accessible from a calibration tool and initialized with values after every reset. This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>3</sup></p>   |
| VAR / CLEARED                                       | <code>&lt;PREFIX&gt;_START_SEC_INTERNAL_VAR_CLEARED_&lt;ALIGNMENT&gt;</code><br><code>&lt;PREFIX&gt;_STOP_SEC_INTERNAL_VAR_CLEARED_&lt;ALIGNMENT&gt;</code>       | <p>To be used for global or static variables that are accessible from a calibration tool and cleared to zero after every reset. This section is <b>DEPRECATED</b> and shall not be used in future development.<sup>3</sup></p>   |
| VAR / NO-INIT                                       | <code>&lt;PREFIX&gt;_START_SEC_VAR_SAVED_ZONE&lt;X&gt;_&lt;ALIGNMENT&gt;</code><br><code>&lt;PREFIX&gt;_STOP_SEC_VAR_SAVED_ZONE&lt;X&gt;_&lt;ALIGNMENT&gt;</code> | <p>To be used for RAM buffers of variables saved in non volatile memory.</p>   |
| CONST / --  | <code>&lt;PREFIX&gt;_START_SEC_CONST_SAVED_RECOVERY_ZONE&lt;X&gt;</code><br><code>&lt;PREFIX&gt;_STOP_SEC_CONST_SAVED_RECOVERY_ZONE&lt;X&gt;</code>               | <p>To be used for ROM buffers of variables saved in non volatile memory.</p>   |

<sup>3</sup> This section was replaced by the generic definition of `<PREFIX>_START_SEC_INTERNAL_VAR_<INIT_POLICY>_<ALIGNMENT>`.



| Memory Section Type / Section Initialization Policy | Syntax of Memory Allocation Keyword  | Comments  |
|---|--|---|
| CONST /<br>--                                       | <PREFIX>_START_SEC_VAR_SAVED_RECOVERY_ZONE<br><X><br><br><PREFIX>_STOP_SEC_VAR_SAVED_RECOVERY_ZONE<<br><X> | To be used for ROM buffers of variables saved in non volatile memory. This section is <b>DEPRECATED</b> and shall not be used in future development. <sup>4</sup> |
| CONST /<br>--                                       | <PREFIX>_START_SEC_CONST_<ALIGNMENT><br><br><PREFIX>_STOP_SEC_CONST_<ALIGNMENT>                            | To be used for global or static constants.  |
| CAL-PRM /<br>--                                     | <PREFIX>_START_SEC_CALIB_<ALIGNMENT><br><br><PREFIX>_STOP_SEC_CALIB_<ALIGNMENT>                            | To be used for calibration constants.   |
| CONST /<br>--                                       | <PREFIX>_START_SEC_CARTO_<ALIGNMENT><br><br><PREFIX>_STOP_SEC_CARTO_<ALIGNMENT>                            | To be used for cartography constants.   |
| CONFIG-<br>DATA /<br>--                             | <PREFIX>_START_SEC_CONFIG_DATA_<ALIGNMENT><br><br><PREFIX>_STOP_SEC_CONFIG_DATA_<ALIGNMENT>                | Constants with attributes that show that they reside in one segment for module configuration.   |

Tabelle 8-1 data sections

**[MEMMAP021]** There are different kinds of execution code sections. This code sections shall be identified with dedicated keywords. If a section is not supported by the integrator and micro controller then be aware that the keyword is ignored. The table below defines the keyword to be used for each code section:

| Memory Section Type / Section Initialization Policy | Syntax of Memory Allocation Keyword   | Comments  |
|---|---|---|
| CODE /<br>--  | <PREFIX>_START_SEC_CODE<br><br><PREFIX>_STOP_SEC_CODE                           | To be used for mapping code to application block, boot block, external flash etc.   |
| CODE /<br>--  | <PREFIX>_START_SEC_CALLOUT_CODE<br><br><PREFIX>_STOP_SEC_CALLOUT_CODE           | To be used for mapping callouts of the BSW Modules  |
| CODE /<br>--  | <PREFIX>_START_SEC_CODE_FAST[_<NUM>]<br><br><PREFIX>_STOP_SEC_CODE_FAST[_<NUM>] | To be used for code that shall go into fast code memory segments. The optional suffix [_<NUM>] can qualify the expected access commonness, e.g. typical period of code execution. |
| CODE /<br>--  | <PREFIX>_START_SEC_CODE_SLOW<br><br><PREFIX>_STOP_SEC_CODE_SLOW                 | To be used for code that shall go into slow code memory segments.   |
| CODE /<br>--  | <PREFIX>_START_SEC_CODE_LIB<br><br><PREFIX>_STOP_SEC_CODE_LIB                   | To be used for code that shall go into library segments for BSW module or Software Component.   |

<sup>4</sup> This section was replaced by the definition of <PREFIX>\_START\_SEC\_CONST\_SAVED\_RECOVERY\_ZONE<X>.

**Tabelle 8-2 code sections**」()

[MEMMAP003] 「Each AUTOSAR basic software module and software component shall wrap declaration and definition of code, variables and constants using the following mechanism:

1. Definition of start symbol for module memory section
2. Inclusion of the memory mapping header file
3. Declaration/definition of code, variables or constants belonging to the specified section
4. Definition of stop symbol for module memory section
5. Inclusion of the memory mapping header file

For code which is invariably implemented as inline function the wrapping with Memory Allocation Keywords is not required.」(BSW006, BSW00306)

Application hint:

For code which is implemented with the `LOCAL_INLINE` macro of the “Compiler.h” the wrapping with Memory Allocation Keywords is required. In the case that the `LOCAL_INLINE` is set to the inline keyword of the compiler the related Memory Allocation Keywords shall not define any linker section assignments or change the addressing behavior because this is already set by the environment of the calling function where the code is inlined. In the case that the `LOCAL_INLINE` is set to empty the related Memory Allocation Keywords shall be configured like for regular code.

For code which his implemented with the `INLINE` macro of the “Compiler.h” the wrapping with Memory Allocation Keywords is required at least for the code which is remaining if `INLINE` is set to empty.

Please note as well that in the Basic Software Module Description the `MemorySection` related to the used Memory Allocation Keywords has to document the usage of `INLINE` and `LOCAL_INLINE` in the option attribute. For further information see [4].

The inclusion of the memory mapping header files within the code is a MISRA violation. As neither executable code nor symbols are included (only pragmas) this violation is an approved exception without side effects.

The start and stop symbols for section control are configured with section identifiers defined in the inclusion of memory mapping header file. For details on configuring sections see “Configuration specification”

For example (BSW Module):

```
#define EEP_START_SEC_VAR_INIT_16
#include "MemMap.h"
static uint16 EepTimer = 100;
static uint16 EepRemainingBytes = 16;
#define EEP_STOP_SEC_VAR_INIT_16
#include "MemMap.h"
```

For example (SWC):

```
#define Abc_START_SEC_CODE
#include "Abc_MemMap.h"
/* --- Write a Code here */
#define Abc_STOP_SEC_CODE
#include "Abc_MemMap.h"
```

**[MEMMAP018]** Each AUTOSAR basic software module and software component shall support, for all C-objects, the configuration of the assignment to one of the memory types (code, variables and constants). 』()

Application hint:

An implicit assignment of objects to default sections is not allowed because properties of default sections are platform and tool dependent and therefore these implementations are not platform independent.

**[MEMMAP023]** Memory mapping header files shall not be included inside the body of a function. 』()

The goal of this requirement is to support compiler which do not support #pragma inside the body of a function. To force a special memory mapping of a function's static variable, this variable must be moved to file static

## 8.2.2 Requirements on memory mapping header files

**[MEMMAP005]** The memory mapping header files shall provide a mechanism to select different code, variable or constant sections by checking the definition of the module specific Memory Allocation Key Words for starting a section (see Recommendation A:). Code, variables or constants declared after this selection shall be mapped to this section. 』(BSW00328, BSW006, BSW00306)

**[MEMMAP026]** The BSW memory mapping header file 'MemMap.h' shall support the Memory Allocation Keywords to start and to stop a section for each MemorySection defined in a BswImplementation which is part of the input configuration. 』()

**[MEMMAP027]** 「The software component type specific memory mapping header file '<SWC>\_MemMap.h' shall support the Memory Allocation Keywords to start and to stop a section for each MemorySection defined in a SwcImplementation associated of this software component type.」()

**[MEMMAP015]** 「The selected section shall be activated, if the section macro is defined before include of the memory mapping header file.」()

**[MEMMAP016]** 「The selection of a section shall only influence the linker's behavior for one of the three different object types code, variables or constants concurrently.」()

Application hint:

On one side the creation of combined sections (for instance code and constants) is not allowed. For the other side the set-up of the compiler / linker must be done in a way, that only the settings of the selected section type is changed. For instance the set-up of the code section shall not influence the configuration of the constant section and other way around.

For instance:

```
#ifdef EEP_START_SEC_VAR_INIT_16
    #undef EEP_START_SEC_VAR_INIT_16
    #define START_SECTION_DATA_INIT_16
#elif
/*
    additional mappings of modules sections into project
    sections
*/
...
#endif

#ifdef START_SECTION_DATA_INIT_16
    #pragma section data "sect_data16"
    #undef START_SECTION_DATA_INIT_16
    #undef MEMMAP_ERROR
#elif
/*
    additional statements for switching the project sections
*/
...
#endif
```

Application hint:

Those code or variables sections can be used for the allocation of objects from more than one module.

Those code or variables sections can be used for the allocation of objects from different module specific code or variable sections of one module.

**[MEMMAP006]** The memory mapping header files shall provide a mechanism to deselect different code and variable sections by checking the definition of the module specific Memory Allocation Key Words for stopping a section (see Recommendation A:). Code or variables declared after this selection shall be mapped to default section. The selected section shall be deactivated if the section macro is defined before include of the memory mapping header file. (BSW006, BSW00306)

For instance:

```
#ifndef EEP_STOP_SEC_CODE
    #undef EEP_STOP_SEC_CODE
    #define STOP_SECTION_COMMON_CODE
#elif
/*
    additional mappings of modules sections into project
    sections
*/
...
#endif

/* additional module specific mappings */
...

#ifndef STOP_SECTION_COMMON_CODE
    #pragma section code restore
    #undef STOP_SECTION_COMMON_CODE
    #undef MEMMAP_ERROR
#elif
/*
    additional statements for switching the project sections
*/
#endif
```

**[MEMMAP007]** The memory mapping header files shall check if they have been included with a valid memory mapping symbol and in a valid sequence (no START preceded by a START, no STOP without the corresponding START). This shall be done by a preprocessor check. (BSW006, BSW00306)

For instance:

```
#define MEMMAP_ERROR
```

```

/*
  mappings of modules sections into project sections and
  statements for switching the project sections
*/

...
#elif STOP_SECTION_COMMON_CODE
  #pragma section code restore
  #undef STOP_SECTION_COMMON_CODE
  #undef MEMMAP_ERROR
#endif

#ifdef MEMMAP_ERROR
  #error "MemMap.h, wrong pragma command"
#endif

```

**[MEMMAP011]** 「The memory mapping header files shall undefine the module or software component specific Memory Allocation Key Words for starting or stopping a section.」(BSW006, BSW00306)

For instance:

```

#ifdef EEP_STOP_SEC_CODE
  #undef EEP_STOP_SEC_CODE

```

**[MEMMAP013]** 「The memory mapping header files shall use if-else structures to reduce the compilation effort.」(BSW006, BSW00306)

For instance:

```

#define MEMMAP_ERROR
...
/* module and ECU specific section mappings */
#if defined START_SECTION_COMMON_CODE
  #pragma section ftext
  #undef START_SECTION_COMMON_CODE
  #undef MEMMAP_ERROR
#elif defined START_SECTION_UNBANKED_CODE
  #pragma section code text
  #undef START_SECTION_UNBANKED_CODE
  #undef MEMMAP_ERROR
#elif defined ...
...
#endif

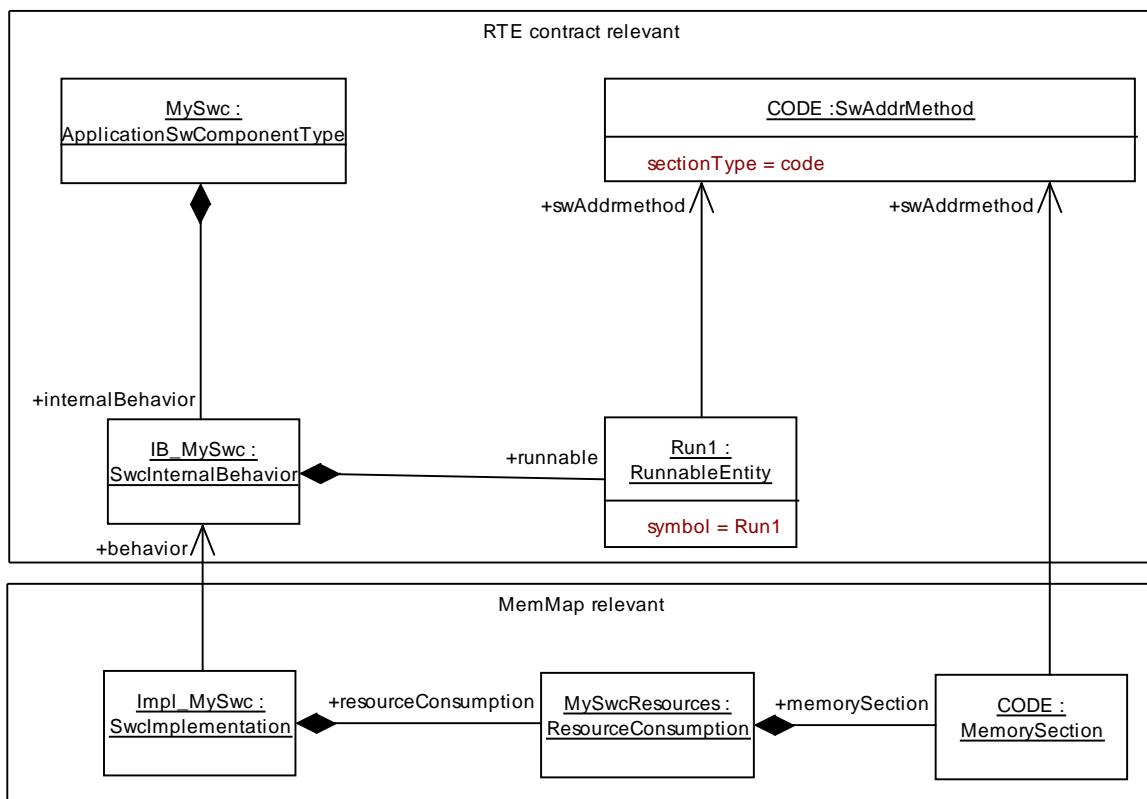
```

### 8.3 Examples

The examples in this section shall illustrate the relationship between the Basic Software Module Descriptions, Software Component Descriptions, the ECU configuration of the Memory Mapping and the Memory Mapping header files.

#### 8.3.1 Code Section

The following example shows `ApplicationSwComponentType` “MySwc” which contains in its `SwcInternalBehavior` a `RunnableEntity` “Run1”. The `RunnableEntity` “Run1” references the `SwAddrMethod` “CODE” which `sectionType` attribute is set to `code`. This expresses the request to allocate the `RunnableEntity` code into a code section with the name `CODE`.



According the SWS RTE the Runnable Entity prototype in the Application Header File of the software component is emitted as:

```
/* Runnable Entity prototype in Application Header File
   Rte_MySwc.h according rte_sws_7194 */
```

```
#define MySwc_START_SEC_CODE
```

```
#include "MySwc_MemMap.h"

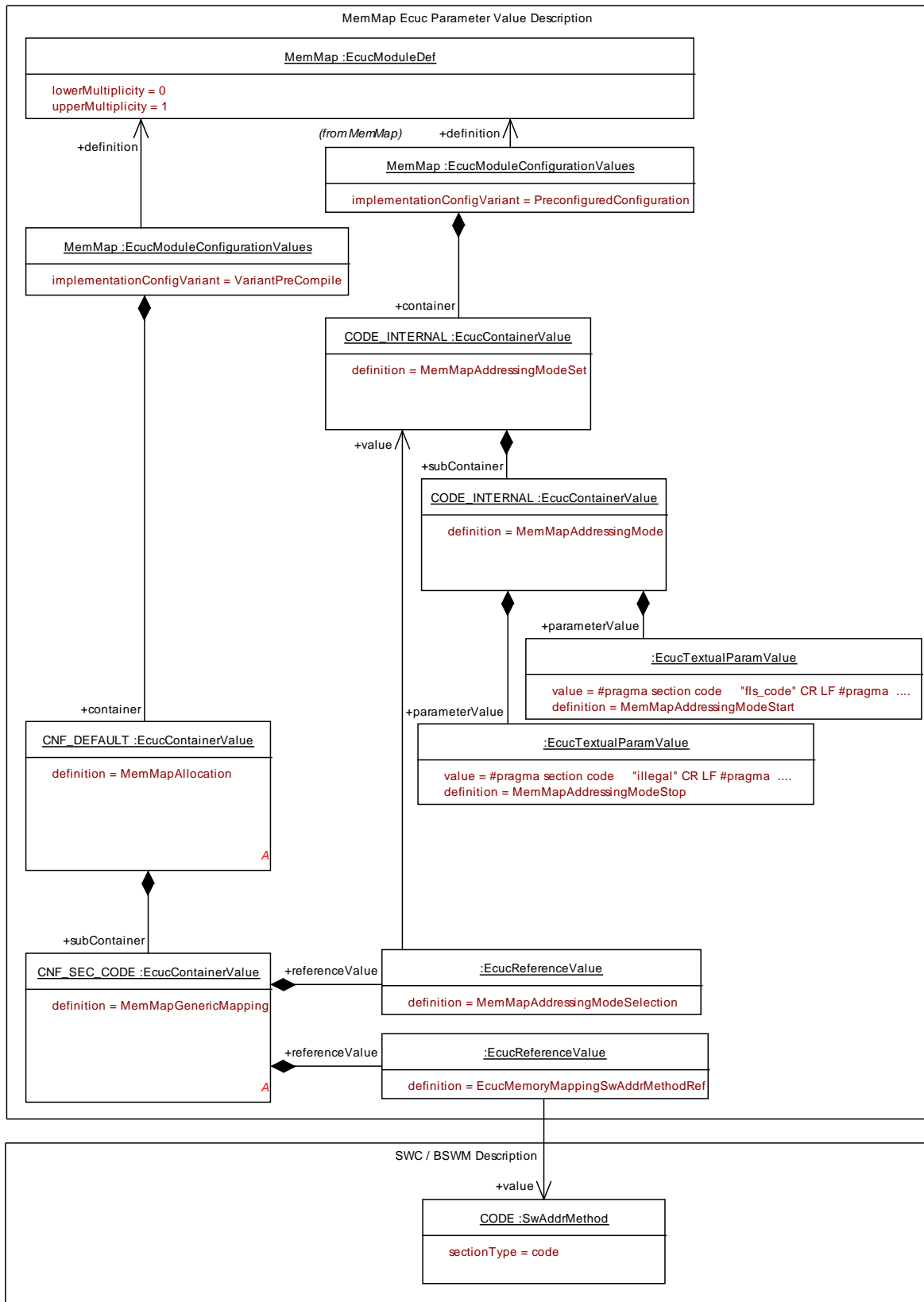
FUNC(void, MySwc_CODE) Run1 (void);

#define MySwc_STOP_SEC_CODE
#include "MySwc_MemMap.h"
```

Please note that the same Memory Allocation Keywords have to be used for the function definition of “Run1” and all other functions of the Software Component which shall be located to same MemorySection.

The SwcImplementation “Impl\_MySwc” associated with the ApplicationSwComponentType “MySwc” defines that it uses a MemorySection named CODE. The MemorySection “CODE” refers to SwAddrMethod “CODE”. This indicates that the module specific (abstract) memory section CODE share a common addressing strategy defined by SwAddrMethod “CODE”.





With the means of the MemMapGenericMapping “CNF\_SEC\_CODE” Memory Mapping is configured that all module specific (abstract) memory sections referring to SwAddrMethod “CODE” are using the MemMapAddressingModeSet

“CODE\_INTERNAL”. MemMapAddressingModeSet “CODE\_INTERNAL” defines the proper statements to start and to stop the mapping of code to the specific linker sections by the usage of the related Memory Allocation Keywords.

With this information of the Memory Allocation Header for the Software Component can be generated like:

```
/* MemMap Header file MySwc_MemMap.h*/  
/*MEMMAP022*/  
  
#ifdef MySwc_START_SEC_CODE  
#pragma section_code "fls_code"  
#pragma ...  
    #undef MySwc_START_SEC_CODE  
  
#ifdef MySwc_STOP_SEC_CODE  
#pragma section_code "illegal"  
    #undef MySwc_STOP_SEC_CODE
```

### 8.3.2 Fast Variable Section

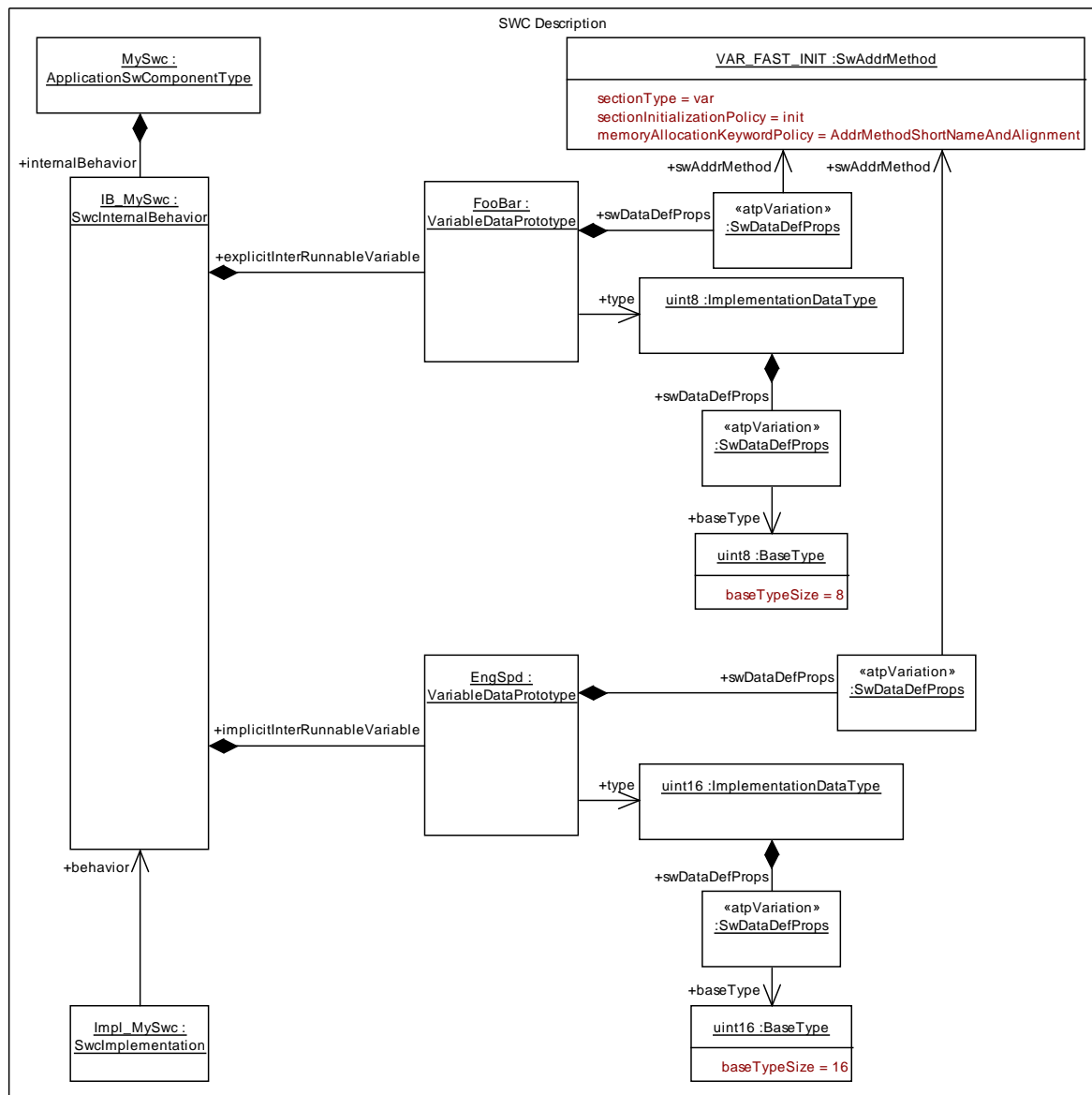
The following example shows ApplicationSwComponentType “MySwc” which contains in its SwcInternalBehavior two VariableDataPrototypes “FooBar” and “EngSpd”.

The VariableDataPrototype “FooBar” references a ImplementationDataType which is associated to a SwBaseType defining baseTypeSize = 8. This denotes a variable size of 8 bit for the data implementing “FooBar”.

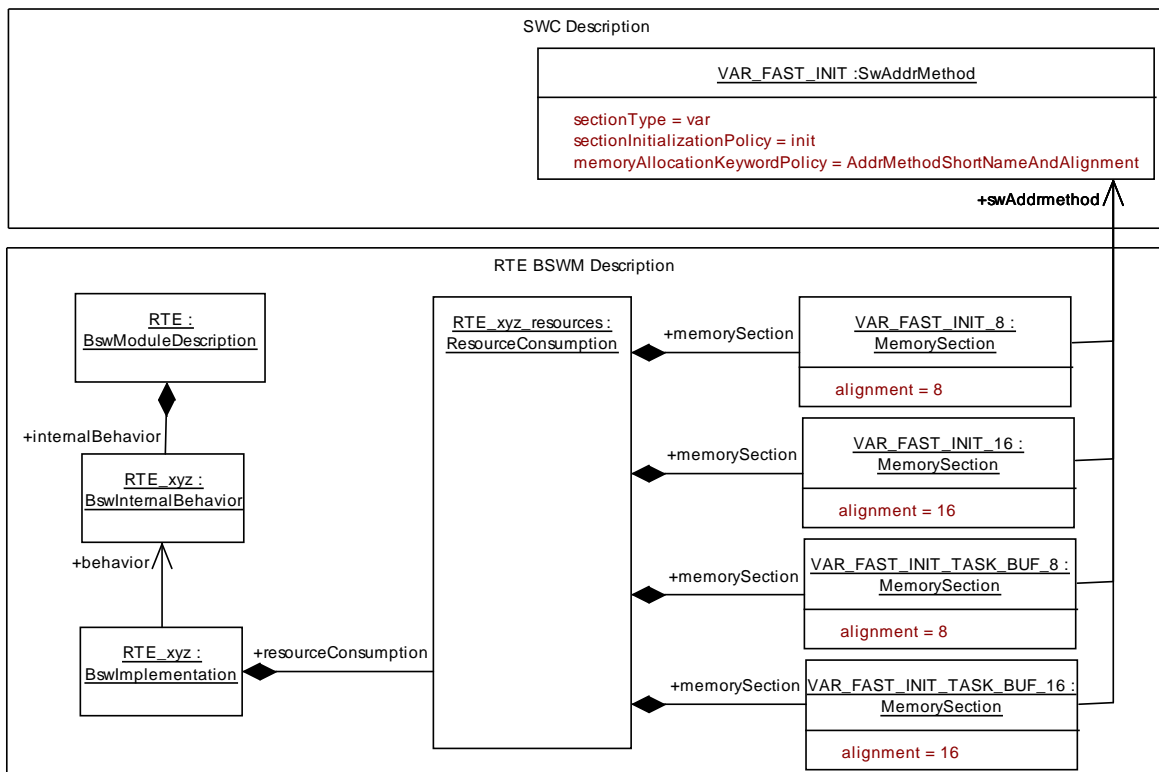
The VariableDataPrototype “EngSpd” references a ImplementationDataType which is associated to a SwBaseType defining baseTypeSize = 16. This denotes a variable size of 16 bit for the data implementing “EngSpd”.

Both VariableDataPrototypes references the SwAddrMethod “VAR\_FAST” which sectionType attribute is set to var and the memoryAllocationKeywordPolicy is set to AddrMethodShortNameAndAlignment.

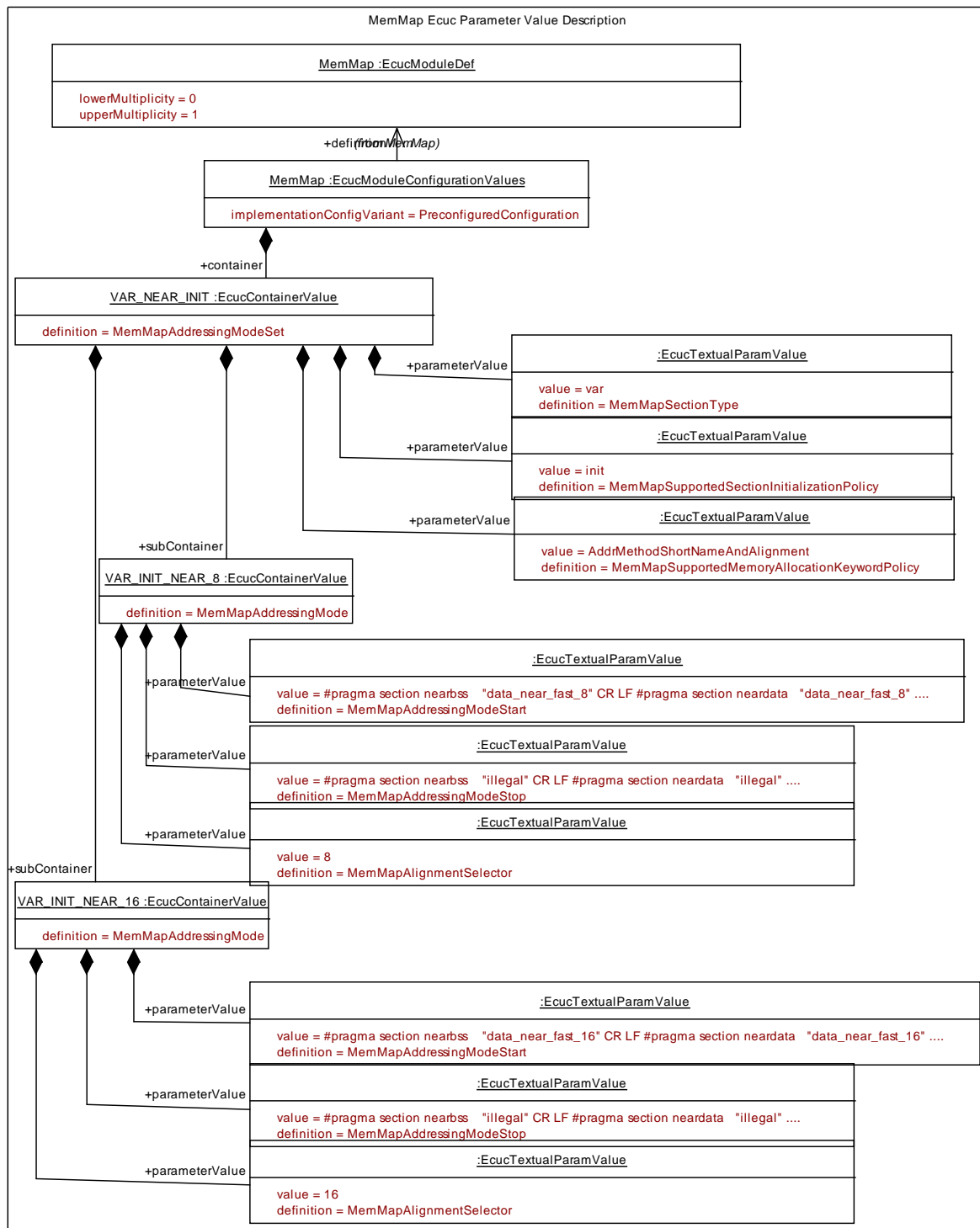
This denotes that the variables implementing the associated VariableDataPrototypes have to be sorted according their size into different MemorySections. The code section with the names shall contain “VAR\_FAST”



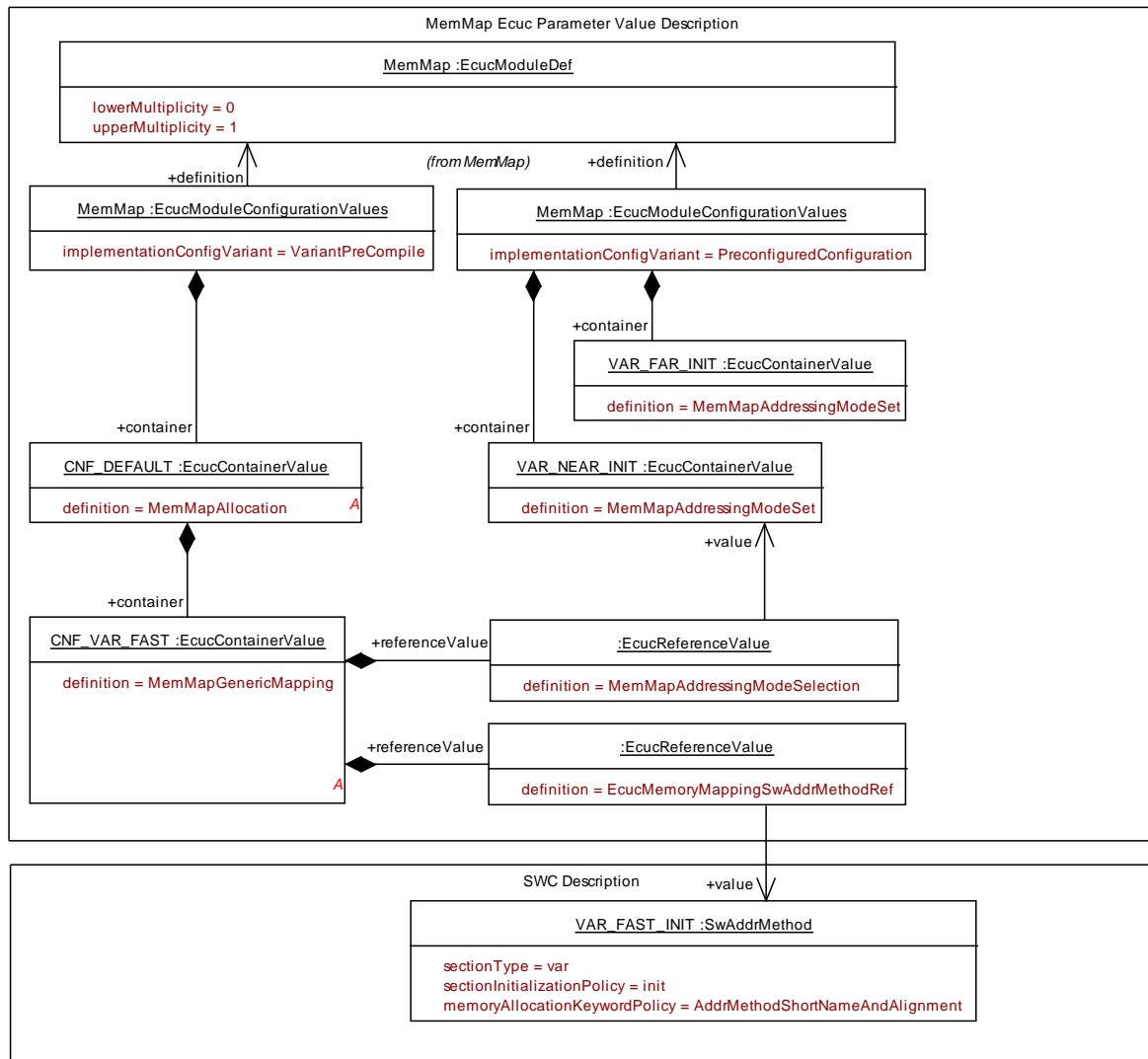
Please note that in this example both `VariableDataPrototypes` have to be implemented by RTE. The RTE again has to provide a BSW Module description defining the used `MemorySections`. Further on the RTE might allocate additional buffer for instance to implement implicit communication behavior. In this example the RTE uses four different `MemorySections` "VAR\_FAST\_8", "VAR\_FAST\_16", "VAR\_FAST\_TASK\_BUF\_8" and "VAR\_FAST\_TASK\_BUF\_8" to sort variables according their size and to allocate additional buffers.



All of these MemorySections are associated with the SwAddrMethod “VAR\_FAST”. This indicates that the module specific (abstract) memory sections “VAR\_FAST\_8”, “VAR\_FAST\_16”, “VAR\_FAST\_TASK\_BUF\_8” and “VAR\_FAST\_TASK\_BUF\_8” share a common addressing strategy defined by SwAddrMethod “VAR\_FAST”.



The ECU Configuration of Memory Mapping defines a MemMapAddressingModeSet “VAR\_NEAR”. This supports the sectionType var, sectionInitializationPolicy = init and memoryAllocationKeywordPolicy = AddrMethodShortNameAndAlignment. In this example MemMapAddressingModes are shown for the alignment 8 and 16 (MemMapAlignmentSelector = 8 and MemMapAlignmentSelector = 16).



With the means of the MemMapGenericMapping “CNF\_VAR\_FAST” Memory Mapping is configured that all module specific (abstract) memory sections referring to SwAddrMethod “VAR\_FAST” are using the MemMapAddressingModeSet “VAR\_NEAR”. MemMapAddressingModeSet “VAR\_NEAR” defines the proper statements to start and to stop the mapping of variables with different alignments (in this example 8 and 16) to the specific linker sections by the usage of the related Memory Allocation Keywords.

With this information of the Memory Allocation Header for the BSW can be generated like:

```

/* MemMap Header file MemMap.h*/

#ifdef RTE_START_SEC_VAR_FAST_8

```

```
#pragma section nearbss      "data_near_fast_8"
#pragma section neardata    "data_near_fast_8"
....
#pragma ...
    #undef RTE_START_SEC_VAR_FAST_8

#ifdef RTE_STOP_SEC_VAR_FAST_8
#pragma section_code "illegal"
    #undef RTE_STOP_SEC_VAR_FAST_8

#ifdef RTE_START_SEC_VAR_FAST_16
#pragma section nearbss      "data_near_fast_16"
#pragma section neardata    "data_near_fast_16"
....
#pragma ...
    #undef RTE_START_SEC_VAR_FAST_16

#ifdef RTE_STOP_SEC_VAR_FAST_16
#pragma section_code "illegal"
    #undef RTE_STOP_SEC_VAR_FAST_16

#ifdef RTE_START_SEC_VAR_FAST_TASK_BUF_8
#pragma section nearbss      "data_near_fast_8"
#pragma section neardata    "data_near_fast_8"
....
#pragma ...
    #undef RTE_START_SEC_VAR_FAST_TASK_BUF_8

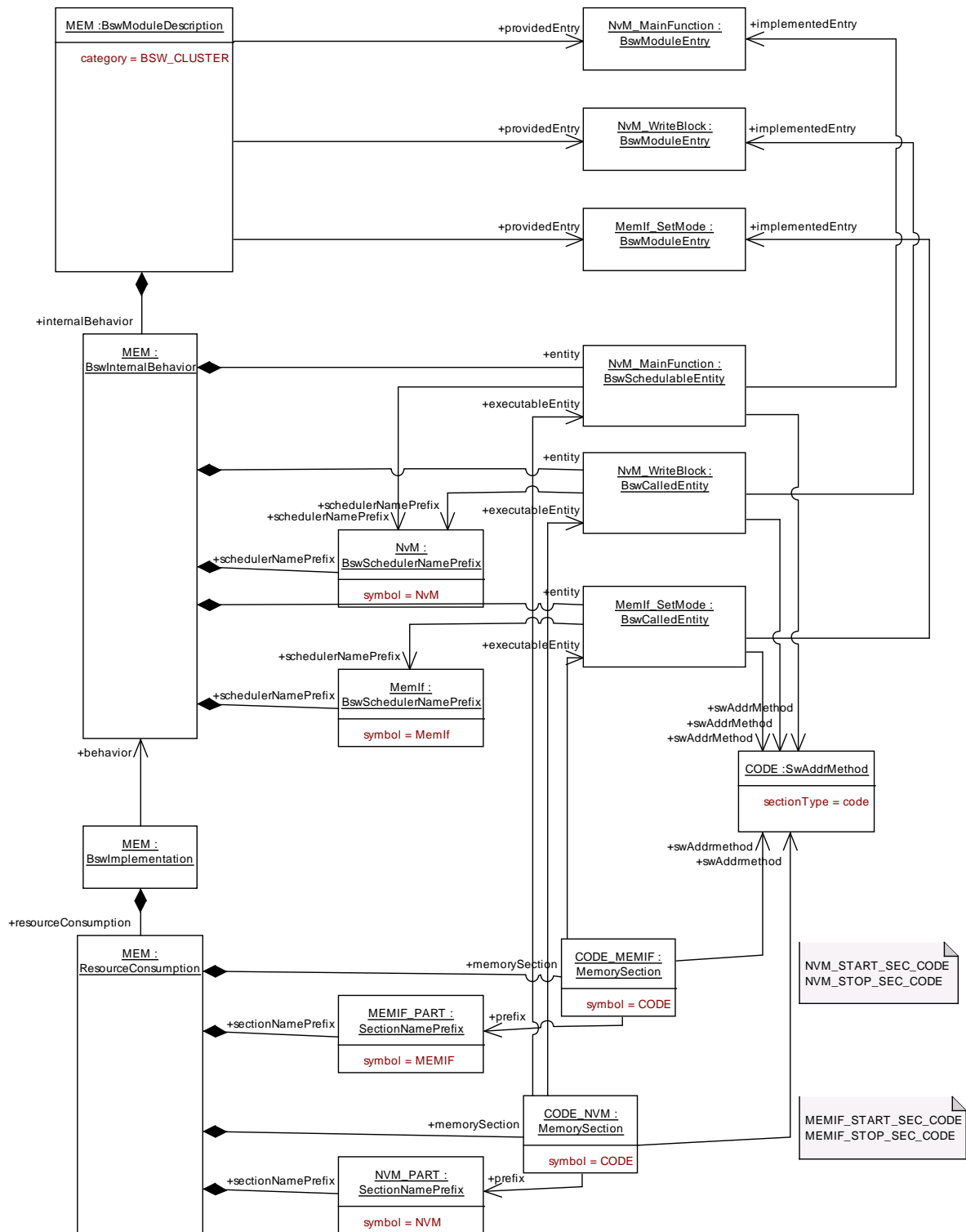
#ifdef RTE_STOP_SEC_VAR_FAST_TASK_BUF_8
#pragma section_code "illegal"
    #undef RTE_STOP_SEC_VAR_FAST_TASK_BUF_8

#ifdef RTE_START_SEC_VAR_FAST_TASK_BUF_16
#pragma section nearbss      "data_near_fast_16"
#pragma section neardata    "data_near_fast_16"
....
#pragma ...
    #undef RTE_START_SEC_VAR_FAST_TASK_BUF_16

#ifdef RTE_STOP_SEC_VAR_FAST_TASK_BUF_16
#pragma section_code "illegal"
    #undef RTE_STOP_SEC_VAR_FAST_TASK_BUF_16
```

### 8.3.3 Code Section in ICC2 cluster

The following Basic Software Module Description would result in the support of the Memory Allocation Keywords in the MemMap.h file:



```
/* MemMap Header file MemMap.h*/
```

```
#ifndef NVM_START_SEC_CODE
...
#endif NVM_STOP_SEC_CODE
```



```
...  
#ifdef MEMIF_START_SEC_CODE  
...  
#ifdef MEMIF_STOP_SEC_CODE
```

## 9 API specification

Not applicable.

## 10 Sequence diagrams

Not applicable.

## 11 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 11.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 11.1 in the specification to guarantee comprehension.

Chapter 11.2 specifies the structure (containers) and the parameters of the module Memory Mapping.

Chapter 11.3 specifies published information of the module Memory Mapping.

### 11.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [6]  
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.

#### 11.1.1 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.

#### 11.1.2 Variants

Thus describe the possible configuration variants of this module. Each Variant must have a unique name which could be referenced to in later chapters. The maximum number of allowed variants is 3. Each variant shall have its own requirement ID.

### 11.1.3 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.

### 11.1.4 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

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <[ReqXXX]>   |
| <b>Container Name</b>           | <Identifies the container by a name, e.g., CanDriverConfiguration> |
| <b>Description</b>              | <Explains the intention and the content of the container .>        |
| <b>Configuration Parameters</b> |  |

|                            |  |                                    |                                 |
|----------------------------|--|------------------------------------|---------------------------------|
| <b>Name</b>                | <Identifies the parameter by name. The naming convention shall follow BSW00408.>   |                                    |                                 |
| <b>Description</b>         | <Explains the intention of the configuration parameter.>   |                                    |                                 |
| <b>Type</b>                | <Specify the type of the parameter (e.g., uint8..uint32) if possible or mark it "--">  |                                    |                                 |
| <b>Unit</b>                | <Specify the unit of the parameter (e.g., ms) if possible or mark it "--" >  |                                    |                                 |
| <b>Range</b>               | <Specify the range (or possible values) of the parameter (e.g., 1..15, ON,OFF) if possible or mark it "--">  | <Describe the value(s) or ranges.> |                                 |
| <b>Configuration Class</b> | <b>Pre-compile</b>   | see <sup>5</sup>                   | <Refer here to (a) variant(s).> |
|                            | <b>Link time</b>   | see <sup>6</sup>                   | <Refer here to (a) variant(s).> |
|                            | <b>Post Build</b>  | see <sup>7</sup>                   | <Refer here to (a) variant(s).> |
| <b>Scope</b>               | <Describe the scope of the parameter if known or mark it as "- -".<br>The scope describes the impact of the configuration parameter: Does the setting affect only one instance of the module (instance), all instances of this module (module), the ECU or a network.<br><br>Possible values of scope :<br>instance, module, ECU, network> |                                    |                                 |
| <b>Dependency</b>          | <Describe the dependencies with respect to the scope if known or mark it as "- -".>  |                                    |                                 |

<sup>5</sup> see the explanation below this table - Pre-compile time

<sup>6</sup> see the explanation below this table - Link time

<sup>7</sup> see the explanation below this table - Post Build

| <b>Included Containers</b>  |  |  |
|---|--|--|
| <b>Container Name</b>   | <b>Multiplicity</b>  | <b>Scope / Dependency</b>  |
| <p><i>&lt;Reference a valid (sub)container by its name, e.g., CanController&gt;</i></p> | <p><i>&lt;Specifies the possible number of instances of the referenced container and its contained configuration parameters.</i></p> <p><i>Possible values:</i><br/> <i>&lt;multiplicity&gt;</i><br/> <i>&lt;min_multiplicity..max_multiplicity&gt;</i><br/> <i>&gt;</i></p> | <p><i>&lt;Describe the scope of the referenced sub-container if known or mark it as "-".</i></p> <p><i>The scope describes the impact of the configuration parameter: Does the setting affect only one instance of the module (instance), all instances of this module (module), the ECU or a network.</i></p> <p><i>Possible values of scope :</i><br/> <i>instance, module, ECU, network&gt;</i></p> <p><i>&lt;Describe the dependencies with respect to the scope if known or mark it as "-".&gt;</i></p> |

## 11.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 8.

### 11.2.1 Variants

The Memory Mapping provides one configuration variant.

#### 11.2.1.1 VARIANT-PRE-COMPILE

MEMMAP024:

Variant 1 – VARIANT-PRE-COMPILE: In this configuration variant all parameters need to be configured pre compile time.

### 11.2.2 MemMap

|                           |                                     |
|---------------------------|-------------------------------------|
| <b>SWS Item</b>           | <b>MemMap001_Conf :</b>             |
| <b>Module Name</b>        | <i>MemMap</i>                       |
| <b>Module Description</b> | Configuration of the MemMap module. |

| <b>Included Containers</b> |                     |  |
|----------------------------|---------------------|--|
| <b>Container Name</b>      | <b>Multiplicity</b> | <b>Scope / Dependency</b>  |
| MemMapAddressingModeSet    | 0..*                | Defines a set of addressing modes which might apply to a SwAddrMethod. |
| MemMapAllocation           | 0..*                | Defines a set of addressing modes which might apply to a SwAddrMethod. |

### 11.2.3 MemMapAddressingModeSet

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>MemMap002_Conf :</b>  |
| <b>Container Name</b>           | MemMapAddressingModeSet  |
| <b>Description</b>              | Defines a set of addressing modes which might apply to a SwAddrMethod. |
| <b>Configuration Parameters</b> |  |

|                      |  |
|----------------------|--|
| <b>SWS Item</b>      | <b>MemMap009_Conf :</b>  |
| <b>Name</b>          | MemMapSupportedAddressingMethodOption  |
| <b>Description</b>   | This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The attribute option of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedAddressMethodOption's |
| <b>Multiplicity</b>  | 0..*   |
| <b>Type</b>          | EcucStringParamDef   |
| <b>Default value</b> | --   |
| <b>maxLength</b>     | --   |

|                           |                                      |    |              |
|---------------------------|--------------------------------------|----|--------------|
| <b>minLength</b>          | --                                   |    |              |
| <b>regularExpression</b>  | [a-zA-Z]([a-zA-Z0-9]_[a-zA-Z0-9])*_? |    |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>              | X  | All Variants |
|                           | <b>Link time</b>                     | -- |              |
|                           | <b>Post-build time</b>               | -- |              |
| <b>Scope / Dependency</b> | scope: ECU                           |    |              |

|                           |   |  |              |
|---------------------------|---|--|--------------|
| <b>SWS Item</b>           | <b>MemMap017_Conf :</b>   |  |              |
| <b>Name</b>               | MemMapSupportedMemoryAllocationKeywordPolicy  |  |              |
| <b>Description</b>        | This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The attribute MemoryAllocationKeywordPolicy of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedMemoryAllocationKeywordPolicy's |  |              |
| <b>Multiplicity</b>       | 0..*  |  |              |
| <b>Type</b>               | EcucEnumerationParamDef   |  |              |
| <b>Range</b>              | MEMMAP_ALLOCATION_KEYWORD_POLICY_ADDR_METHOD_SHORT_NAME   | The Memory Allocation Keyword is build with the short name of the SwAddrMethod. This is the default value if the attribute does not exist in the SwAddrMethod.   |              |
|                           | MEMMAP_ALLOCATION_KEYWORD_POLICY_ADDR_METHOD_SHORT_NAME_AND_ALIGNMENT   | The Memory Allocation Keyword is build with the the short name of the SwAddrMethod and the alignment attribute of the MemorySection. This requests a separation of objects in memory dependent from the alignment and is not applicable for RunnableEntitys and BswSchedulableEntitys. |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                           | <b>Link time</b>  | --   |              |
|                           | <b>Post-build time</b>  | --   |              |
| <b>Scope Dependency</b>   | scope: ECU  |  |              |

|                    |   |  |  |
|--------------------|---|--|--|
| <b>SWS Item</b>    | <b>MemMap008_Conf :</b>   |  |  |
| <b>Name</b>        | MemMapSupportedSectionInitializationPolicy  |  |  |
| <b>Description</b> | This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The sectionIntializationPolicy attribute value |  |  |



|                           |   |    |              |
|---------------------------|---|----|--------------|
|                           | of a swAddrMethod mapped via MemMapGenericMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedSectionInitializationPolicy's. Please note that SectionInitializationPolicyType describes the intended initialization of MemorySections. The following values are standardized in AUTOSAR Methodology: * "NO-INIT": No initialization and no clearing is performed. Such data elements must not be read before one has written a value into it. * "INIT": To be used for data that are initialized by every reset to the specified value (initValue). * "POWER-ON-INIT": To be used for data that are initialized by "Power On" to the specified value (initValue). Note: there might be several resets between power on resets. * "CLEARED": To be used for data that are initialized by every reset to zero. * "POWER-ON-CLEARED": To be used for data that are initialized by "Power On" to zero. Note: there might be several resets between power on resets. Please note that the values are defined similar to the representation of enumeration types in the XML schema to ensure backward compatibility. |    |              |
| <b>Multiplicity</b>       | 0..*  |    |              |
| <b>Type</b>               | EcucStringParamDef  |    |              |
| <b>Default value</b>      | --  |    |              |
| <b>maxLength</b>          | --  |    |              |
| <b>minLength</b>          | --  |    |              |
| <b>regularExpression</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> | scope: ECU  |    |              |

|                     |  |   |
|---------------------|--|---|
| <b>SWS Item</b>     | <b>MemMap007_Conf :</b>  |   |
| <b>Name</b>         | MemMapSupportedSectionType   |   |
| <b>Description</b>  | This constrains the usage of this addressing mode set for Generic Mappings to swAddrMethods. The attribute sectionType of a swAddrMethod mapped via MemMapGenericMapping or MemMapSectionSpecificMapping to this MemMapAddressingModeSet shall be equal to one of the configured MemMapSupportedSectionType's. |   |
| <b>Multiplicity</b> | 0..*   |   |
| <b>Type</b>         | EcucEnumerationParamDef  |   |
| <b>Range</b>        | MEMMAP_SECTION_TYPE_CALIBRATION_OFFLINE  | Program data which can only be used for offline calibration. Note: This value is deprecated and shall be substituted by calPrm.   |
|                     | MEMMAP_SECTION_TYPE_CALIBRATION_ONLINE   | Program data which can be used for online calibration. Note: This value is deprecated and shall be substituted by calPrm.   |
|                     | MEMMAP_SECTION_TYPE_CAL_PRM  | To be used for calibratable constants of ECU-functions.   |
|                     | MEMMAP_SECTION_TYPE_CODE   | To be used for mapping code to application block, boot block, external flash etc.   |
|                     | MEMMAP_SECTION_TYPE_CONFIG_DATA  | Constants with attributes that show that they reside in one segment for module configuration.   |
|                     | MEMMAP_SECTION_TYPE_CONST  | To be used for global or static constants.  |
|                     | MEMMAP_SECTION_TYPE_EXCLUDE_FROM_FLASH   | Values existing in the ECU but not dropped down in the binary file. No upload should be needed to obtain access to the ECU data. The ECU will never be touched by the instrumentation tool, with the exception of upload. These are memory areas which are not overwritten by downloading the executable. |
|                     | MEMMAP_SECTION_TYPE_USER_DEFINED   | No specific categorization of sectionType possible. Note: This value is deprecated and shall be substituted by var,   |

|                             |                                       |  |              |
|-----------------------------|---------------------------------------|--|--------------|
|                             |                                       | code, const, calPrm, configData, excludeFromFlash and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option.   |              |
|                             | MEMMAP_SECTION_TYPE_VAR               | To be used for global or static variables. The expected initialization is specified with the attribute sectionInitializationPolicy.  |              |
|                             | MEMMAP_SECTION_TYPE_VAR_FAST          | To be used for all global or static variables that have at least one of the following properties: - accessed bit-wise - frequently used - high number of accesses in source code Some platforms allow the use of bit instructions for variables located in this specific RAM area as well as shorter addressing instructions. This saves code and runtime. Note: This value is deprecated and shall be substituted by var and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option. |              |
|                             | MEMMAP_SECTION_TYPE_VAR_NO_INIT       | To be used for all global or static variables that are never initialized. Note: This value is deprecated and shall be substituted by var and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option.  |              |
|                             | MEMMAP_SECTION_TYPE_VAR_POWER_ON_INIT | To be used for all global or static variables that are initialized only after power on reset. Note: This value is deprecated and shall be substituted by var and the appropriate values of the orthogonal attributes sectionInitializationPolicy, memoryAllocationKeywordPolicy and option.  |              |
| <b>ConfigurationClasses</b> | <b>Pre-compile time</b>               | X  | All Variants |
|                             | <b>Link time</b>                      | --   |              |
|                             | <b>Post-build time</b>                | --   |              |
| <b>Scope Dependency</b>     | scope: ECU                            |  |              |

| <b>Included Containers</b> |                     |  |
|----------------------------|---------------------|--|
| <b>Container Name</b>      | <b>Multiplicity</b> | <b>Scope / Dependency</b>  |
| MemMapAddressingMode       | 1..*                | Defines a addressing mode with a set of #pragma statements implementing the start and the stop of a section. |

### 11.2.4 MemMapAddressingMode

|                                 |  |
|---------------------------------|--|
| <b>SWS Item</b>                 | <b>MemMap003_Conf :</b>  |
| <b>Container Name</b>           | MemMapAddressingMode   |
| <b>Description</b>              | Defines a addressing mode with a set of #pragma statements implementing the start and the stop of a section. |
| <b>Configuration Parameters</b> |  |

|                     |  |
|---------------------|--|
| <b>SWS Item</b>     | <b>MemMap004_Conf :</b>  |
| <b>Name</b>         | MemMapAddressingModeStart  |
| <b>Description</b>  | Defines a set of #pragma statements implementing the start of a section. |
| <b>Multiplicity</b> | 1  |
| <b>Type</b>         | EcucMultilineStringParamDef  |

|                           |                         |    |              |
|---------------------------|-------------------------|----|--------------|
| <b>Default value</b>      | --                      |    |              |
| <b>maxLength</b>          | --                      |    |              |
| <b>minLength</b>          | --                      |    |              |
| <b>regularExpression</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> | scope: ECU              |    |              |

|                           |  |    |              |
|---------------------------|--|----|--------------|
| <b>SWS Item</b>           | <b>MemMap005_Conf :</b>  |    |              |
| <b>Name</b>               | MemMapAddressingModeStop   |    |              |
| <b>Description</b>        | Defines a set of #pragma statements implementing the start of a section. |    |              |
| <b>Multiplicity</b>       | 1  |    |              |
| <b>Type</b>               | EcucMultilineStringParamDef  |    |              |
| <b>Default value</b>      | --   |    |              |
| <b>maxLength</b>          | --   |    |              |
| <b>minLength</b>          | --   |    |              |
| <b>regularExpression</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> | scope: ECU   |    |              |

|                           |  |    |              |
|---------------------------|--|----|--------------|
| <b>SWS Item</b>           | <b>MemMap006_Conf :</b>  |    |              |
| <b>Name</b>               | MemMapAlignmentSelector  |    |              |
| <b>Description</b>        | Defines a the alignments for which the MemMapAddressingMode applies. The to be used alignment is defined in the alignment attribute of the MemorySection. If the MemMapAlignmentSelector fits to alignment attribute of the MemorySection the set of #pragmas of the related MemMapAddressingMode shall be used to implement the start and the stop of a section. Please note that the same MemMapAddressingMode can be applicable for several alignments, e.g. "8" bit and "UNSPECIFIED". |    |              |
| <b>Multiplicity</b>       | 1..*   |    |              |
| <b>Type</b>               | EcucStringParamDef   |    |              |
| <b>Default value</b>      | --   |    |              |
| <b>maxLength</b>          | --   |    |              |
| <b>minLength</b>          | --   |    |              |
| <b>regularExpression</b>  | [1-9][0-9]* 0x[0-9a-f]* 0[0-7]* 0b[0-1]* UNSPECIFIED UNKNOWN BOOLEAN   |    |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                           | <b>Link time</b>   | -- |              |
|                           | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b> | scope: ECU   |    |              |

**No Included Containers**

### 11.2.5 MemMapAllocation

|                                 |  |  |  |
|---------------------------------|--|--|--|
| <b>SWS Item</b>                 | <b>MemMap010_Conf :</b>  |  |  |
| <b>Container Name</b>           | MemMapAllocation   |  |  |
| <b>Description</b>              | Defines a set of addressing modes which might apply to a SwAddrMethod. |  |  |
| <b>Configuration Parameters</b> |  |  |  |

| <b>Included Containers</b>   |                     |   |
|------------------------------|---------------------|---|
| <b>Container Name</b>        | <b>Multiplicity</b> | <b>Scope / Dependency</b>   |
| MemMapGenericMapping         | 0..*                | Defines which SwAddrMethod is implemented with which MemMapAddressingModeSet. The pragmas for the implementation of the MemorySelectorKeywords are taken from the MemMapAddressingModeStart and MemMapAddressingModeStop parameters of the MemMapAddressingModeSet for the individual alignments. That this mapping becomes valid requires matching MemMapSupportedSectionType's, MemMapSupportedSectionInitializationPolicy's and MemMapSupportedAddressingMethodOption's. The MemMapGenericMapping applies only if it is not overruled by an MemMapSectionSpecificMapping |
| MemMapSectionSpecificMapping | 0..*                | Defines which MemorySection of a BSW Module or a Software Component is implemented with which MemMapAddressingModeSet. The pragmas for the implementation of the MemorySelectorKeywords are taken from the MemMapAddressingModeStart and MemMapAddressingModeStop parameters of the MemMapAddressingModeSet for the specific alignment of the MemorySection. The MemMapSectionSpecificMapping precedes a mapping defined by MemMapGenericMapping.   |

### 11.2.6 MemMapGenericMapping

| <b>SWS Item</b>                 | <b>MemMap011_Conf :</b>  |
|---------------------------------|--|
| <b>Container Name</b>           | MemMapGenericMapping   |
| <b>Description</b>              | Defines which SwAddrMethod is implemented with which MemMapAddressingModeSet.<br>The pragmas for the implementation of the MemorySelectorKeywords are taken from the MemMapAddressingModeStart and MemMapAddressingModeStop parameters of the MemMapAddressingModeSet for the individual alignments.<br>That this mapping becomes valid requires matching MemMapSupportedSectionType's, MemMapSupportedSectionInitializationPolicy's and MemMapSupportedAddressingMethodOption's.<br>The MemMapGenericMapping applies only if it is not overruled by an MemMapSectionSpecificMapping |
| <b>Configuration Parameters</b> |  |

| <b>SWS Item</b>           | <b>MemMap012_Conf :</b>   |    |              |
|---------------------------|---|----|--------------|
| <b>Name</b>               | MemMapAddressingModeSetRef  |    |              |
| <b>Description</b>        | Reference to the MemMapAddressingModeSet which applies to the MemMapGenericMapping. |    |              |
| <b>Multiplicity</b>       | 1   |    |              |
| <b>Type</b>               | Reference to [ MemMapAddressingModeSet ]  |    |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                           | <b>Link time</b>  | -- |              |
|                           | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b> | scope: ECU  |    |              |

|                           |  |    |              |
|---------------------------|--|----|--------------|
| <b>SWS Item</b>           | <b>MemMap013_Conf :</b>  |    |              |
| <b>Name</b>               | MemMapSwAddressMethodRef   |    |              |
| <b>Description</b>        | Reference to the SwAddrMethod which applies to the MemMapGenericMapping. |    |              |
| <b>Multiplicity</b>       | 1  |    |              |
| <b>Type</b>               | Foreign reference to [ SW-ADDR-METHOD ]                                  |    |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>  | X  | All Variants |
|                           | <b>Link time</b>   | -- |              |
|                           | <b>Post-build time</b>   | -- |              |
| <b>Scope / Dependency</b> | scope: ECU   |    |              |

**No Included Containers**

### 11.2.7 MemMapSectionSpecificMapping

|                                 |  |  |  |
|---------------------------------|--|--|--|
| <b>SWS Item</b>                 | <b>MemMap014_Conf :</b>  |  |  |
| <b>Container Name</b>           | MemMapSectionSpecificMapping   |  |  |
| <b>Description</b>              | <p>Defines which MemorySection of a BSW Module or a Software Component is implemented with which MemMapAddressingModeSet.</p> <p>The pragmas for the implementation of the MemorySelectorKeywords are taken from the MemMapAddressingModeStart and MemMapAddressingModeStop parameters of the MemMapAddressingModeSet for the specific alignment of the MemorySection.</p> <p>The MemMapSectionSpecificMapping precedes a mapping defined by MemMapGenericMapping.</p> |  |  |
| <b>Configuration Parameters</b> |  |  |  |

|                           |   |    |              |
|---------------------------|---|----|--------------|
| <b>SWS Item</b>           | <b>MemMap015_Conf :</b>   |    |              |
| <b>Name</b>               | MemMapAddressingModeSetRef  |    |              |
| <b>Description</b>        | Reference to the MemMapAddressingModeSet which applies to the MemMapModuleSectionSpecificMapping. |    |              |
| <b>Multiplicity</b>       | 1   |    |              |
| <b>Type</b>               | Reference to [ MemMapAddressingModeSet ]  |    |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                           | <b>Link time</b>  | -- |              |
|                           | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b> | scope: ECU  |    |              |

|                           |   |    |              |
|---------------------------|---|----|--------------|
| <b>SWS Item</b>           | <b>MemMap016_Conf :</b>   |    |              |
| <b>Name</b>               | MemMapMemorySectionRef  |    |              |
| <b>Description</b>        | Reference to the MemorySection which applies to the MemMapSectionSpecificMapping. |    |              |
| <b>Multiplicity</b>       | 1   |    |              |
| <b>Type</b>               | Foreign reference to [ MEMORY-SECTION ]   |    |              |
| <b>ConfigurationClass</b> | <b>Pre-compile time</b>   | X  | All Variants |
|                           | <b>Link time</b>  | -- |              |
|                           | <b>Post-build time</b>  | -- |              |
| <b>Scope / Dependency</b> | scope: ECU  |    |              |

**No Included 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> .  |

### 11.3 Published Information

[MEMMAP030] 「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.

## 12 Not applicable requirements

**[MEMMAP999]** 「 These requirements are not applicable to this specification. 」  
(BSW00344, BSW00404, BSW00405, BSW00345, BSW159, BSW167, BSW171, BSW170, BSW00380, BSW00419, BSW00381, BSW00412, BSW00383, BSW00387, BSW00388, BSW00389, BSW00390, BSW00391, BSW00392, BSW00393, BSW00394, BSW00395, BSW00396, BSW00397, BSW00398, BSW00399, BSW00400, BSW00375, BSW101, BSW00416, BSW00406, BSW168, BSW00407, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW00336, BSW00337, BSW00338, BSW00369, BSW00339, BSW00421, BSW00422, BSW00420, BSW00417, BSW00323, BSW004, BSW00409, BSW00385, BSW00386, BSW161, BSW162, BSW00324, BSW005, BSW00415, BSW164, BSW00325, BSW00326, BSW00342, BSW00343, BSW160, BSW007, BSW00300, BSW00413, BSW00347, BSW00605, BSW00307, BSW00310, BSW00373, BSW00327, BSW00335, BSW00350, BSW00408, BSW00410, BSW00411, BSW00346, BSW158, BSW00314, BSW00370, BSW00348, BSW00353, BSW00301, BSW00302, BSW00312, BSW00357, BSW00377, BSW00304, BSW00355, BSW00378, BSW00308, BSW00309, BSW00371, BSW00358, BSW00414, BSW00359, BSW00360, BSW00329, BSW00330, BSW00331, BSW009, BSW00401, BSW172, BSW010, BSW00333, BSW00341, BSW00334)