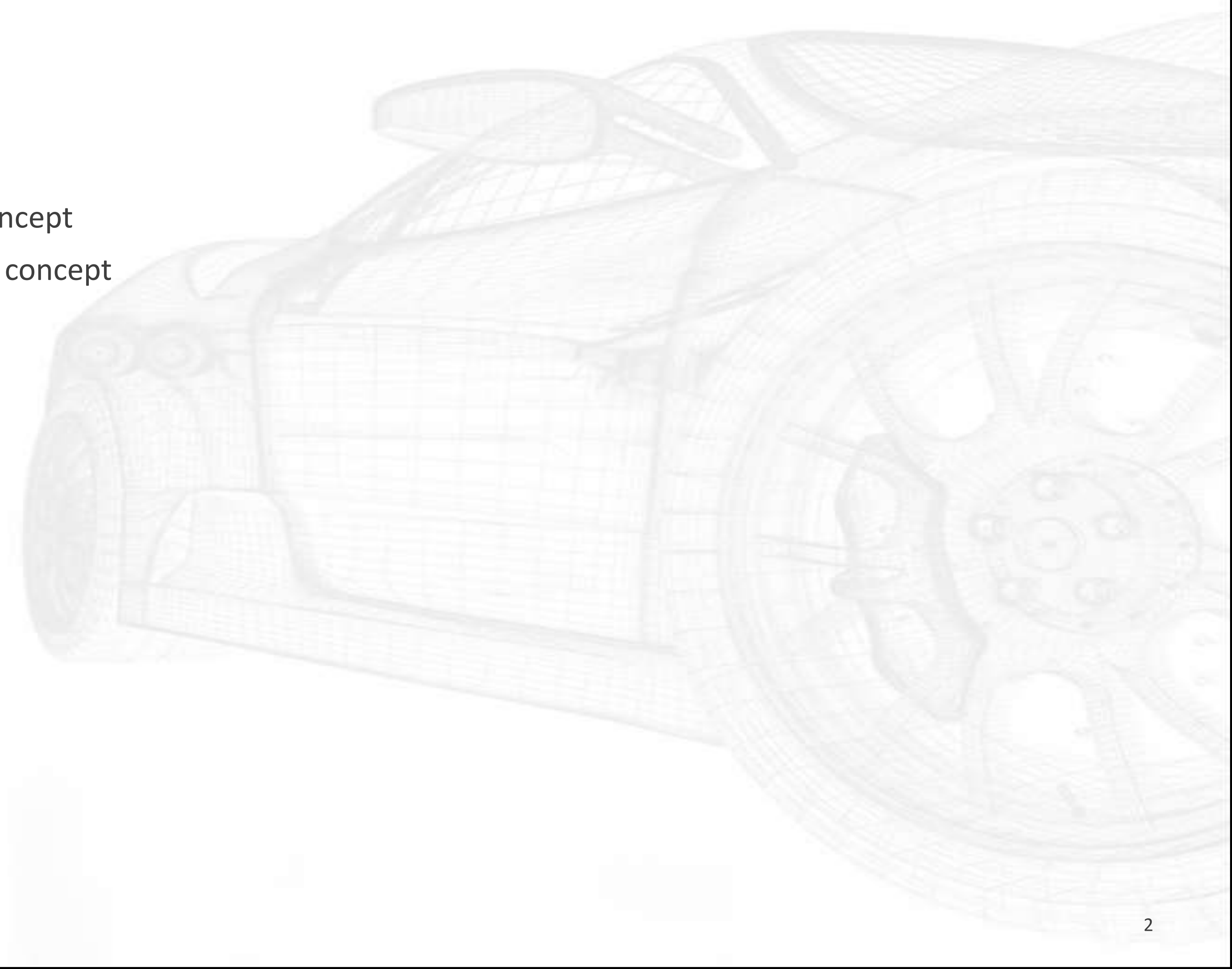
EB tresos Expert Training – Memory Mapping

Elektrobit

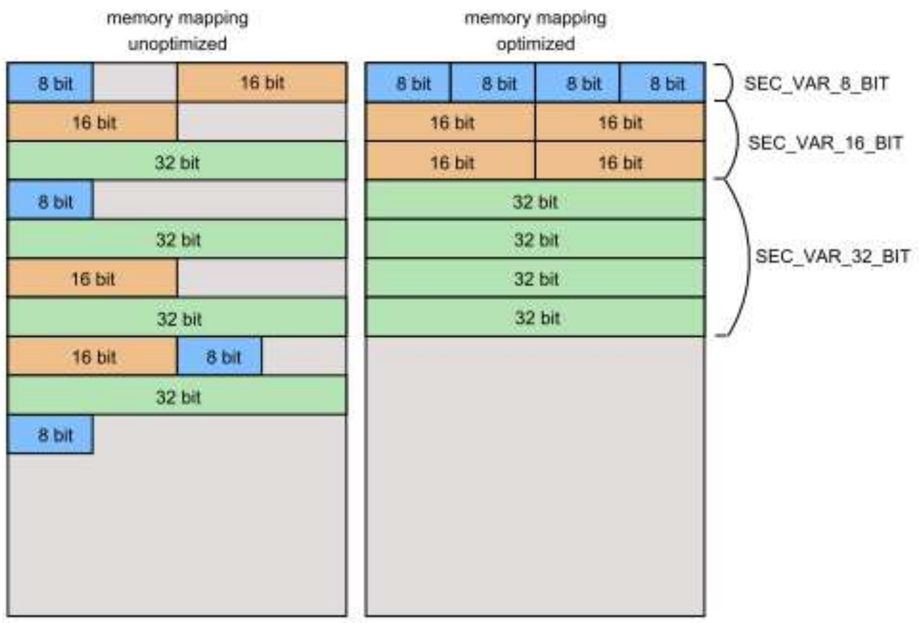
Agenda



- Objectives of the AUTOSAR memory mapping concept
- Overview about the AUTOSAR memory mapping concept
- What is the MemMap module?
- How to configure the MemMap module

Objectives of the AUTOSAR memory mapping concept

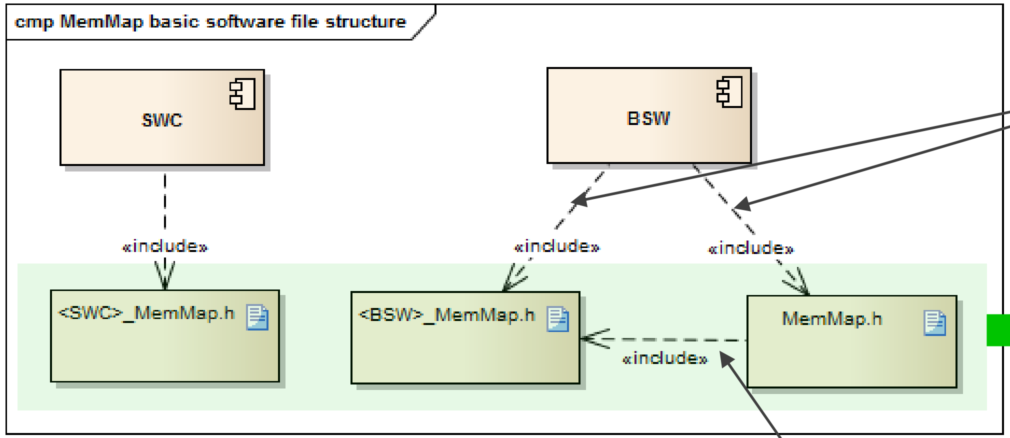
- Avoidance of waste of RAM
  - Avoid gaps in the RAM when the different variables (8, 16, 32 bit) are allocated
- Usage of specific RAM properties
  - RAM which is not initialized after a power-on-reset
  - Core local RAM
- Usage of specific ROM properties
  - Internal flash / external flash
- Usage of the same source code of a module for boot loader and application
- Support of Memory Protection
  - Separate module variables into different areas which are protected via the Memory Protection Unit
- Support of partitioning



Optimized memory usage without gaps in memory
Source: EB tresos AutoCore Generic 8 - documentation

Overview about the AUTOSAR memory mapping concept

- Concept is applicable for each AUTOSAR basic software module and software component
- The BSW /SWC includes the MemMap header files



Depends on the version of the AUTOSAR specification

Project specific files: Can be easily generated using the EB MemMap module

Ensure compatibility between the different version of the AUTOSAR specification

AUTOSAR memory mapping usage in a BSW module

- Example: CanIf module

```
#define CANIF_START_SEC_VAR_INIT_8
#include <CanIf_MemMap.h>

/** \brief Initialization state
 ** This variable shows the current state of the CAN interface.
 ** CANIF_UNINIT            CanIf is not initialized
 ** CANIF_INITIALIZED       CanIf is initialized
 */
STATIC VAR( uint8, CANIF_VAR ) CanIf_State = CANIF_UNINITIALIZED;

#define CANIF_STOP_SEC_VAR_INIT_8
#include <CanIf_MemMap.h>
```

Variables defined within the memory section can be mapped in a project specific way **without touching the CanIf implementation**

CanIf internal variable

Memory mapping „keywords“

- Used keywords
  - `<PREFIX>_START_SEC_<NAME>`
  - `<PREFIX>_STOP_SEC_<NAME>`

- `<PREFIX>`
  - Software component: Short name of the software component type (case sensitive)
  - BSW module: Composed according to <snp>[_<vi>_<ai>]
    - <snp>: BswModuleDescription’s short name (upper case letters)
    - <vi>: vendorId of the BSW module (optional)
    - <ai>: vendorApiInfix of the BSW module (optional)

- `<NAME>`
  - Short name of the memory section

Memory mapping „keywords“

- Usual patterns for the keywords
  - `{PREFIX}_START_SEC_CODE[_{safety}][_{coreScope}]`
  - `{PREFIX}_STOP_SEC_CODE[_{safety}][_{coreScope}]`

  - `{PREFIX}_START_SEC_VAR_{INIT_POLICY}[_{safety}][_{coreScope}]_{ALIGNMENT}`
  - `{PREFIX}_STOP_SEC_VAR_{INIT_POLICY}[_{safety}][_{coreScope}]_{ALIGNMENT}`

Memory mapping „keywords“

`safety`

- Optional tag
- Can be used to indicate restrictions
- Possible options:
  - `QM`
  - `ASIL_A`
  - `ASIL_B`
  - `ASIL_C`
  - `ASIL_D`

`coreScope`

- Optional tag
- Can be used for multi-core ECUs to indicate if the code / data is executed / accessed by any core or by a specific core
- Possible options:
  - `GLOBAL`
  - `LOCAL`

Memory mapping „keywords“

ALIGNMENT

- BOOLEAN
  - Used for variables / constants of size 1 bit
- 8
  - Used for variables / constants which have to be aligned to 8 bit
- 16
- 32
- PTR
- UNSPECIFIED
  - Used for variables / constants / structures / arrays / unions when exisiting size alignment does not fit

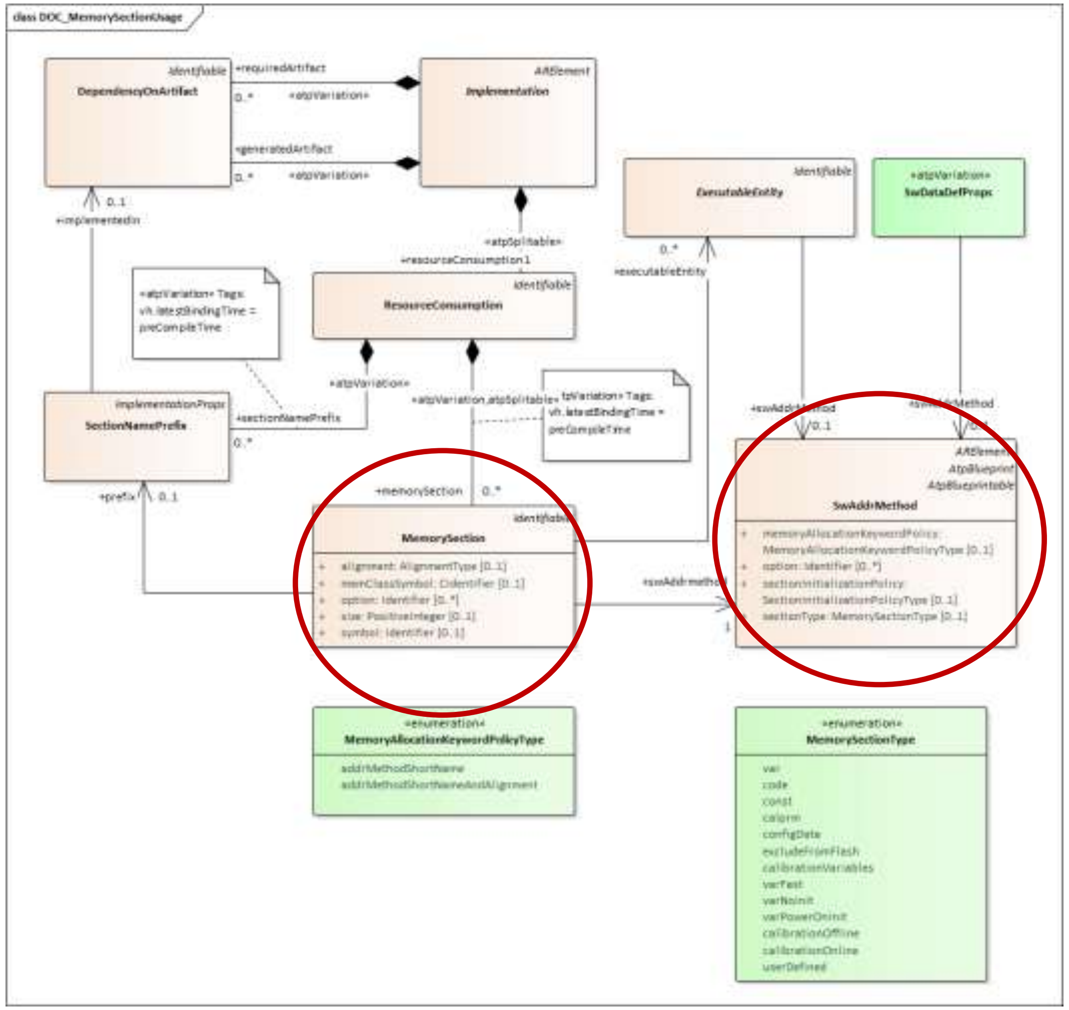
INIT_POLICY

- 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

Memory Mapping in the AUTOSAR Meta Model

- Each BSW module and the software components define the required MemorySections

- These MemorySections refer to SwAddrMethod elements



Source: UML diagram from the AUTOSAR 4.3 Meta Model

AUTOSAR description files

MemorySection

```
<MEMORY-SECTION>
       <SHORT-NAME>VAR_INIT</SHORT-NAME>
       <ALIGNMENT>8</ALIGNMENT>
       <OPTIONS>
                <OPTION>coreLocal</OPTION>
       </OPTIONS>
       <SW-ADDRMETHOD-REF DEST="SW-ADDR-METHOD">
                /AUTOSAR_MemMap/SwAddrMethods/VAR_INIT_LOCAL
       </SW-ADDRMETHOD-REF>
       <SYMBOL>VAR_INIT_LOCAL_8</SYMBOL>
</MEMORY-SECTION>
```

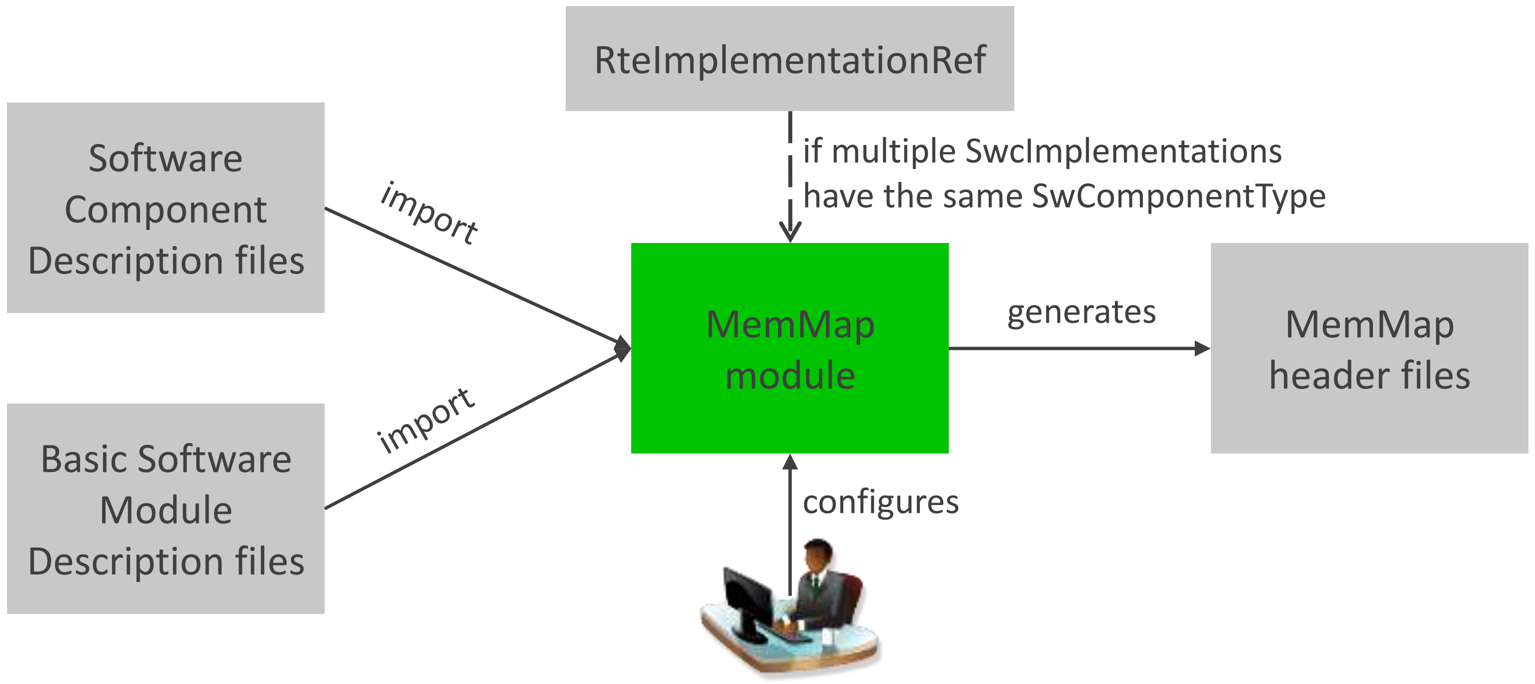
AUTOSAR description files

SwAddrMethod

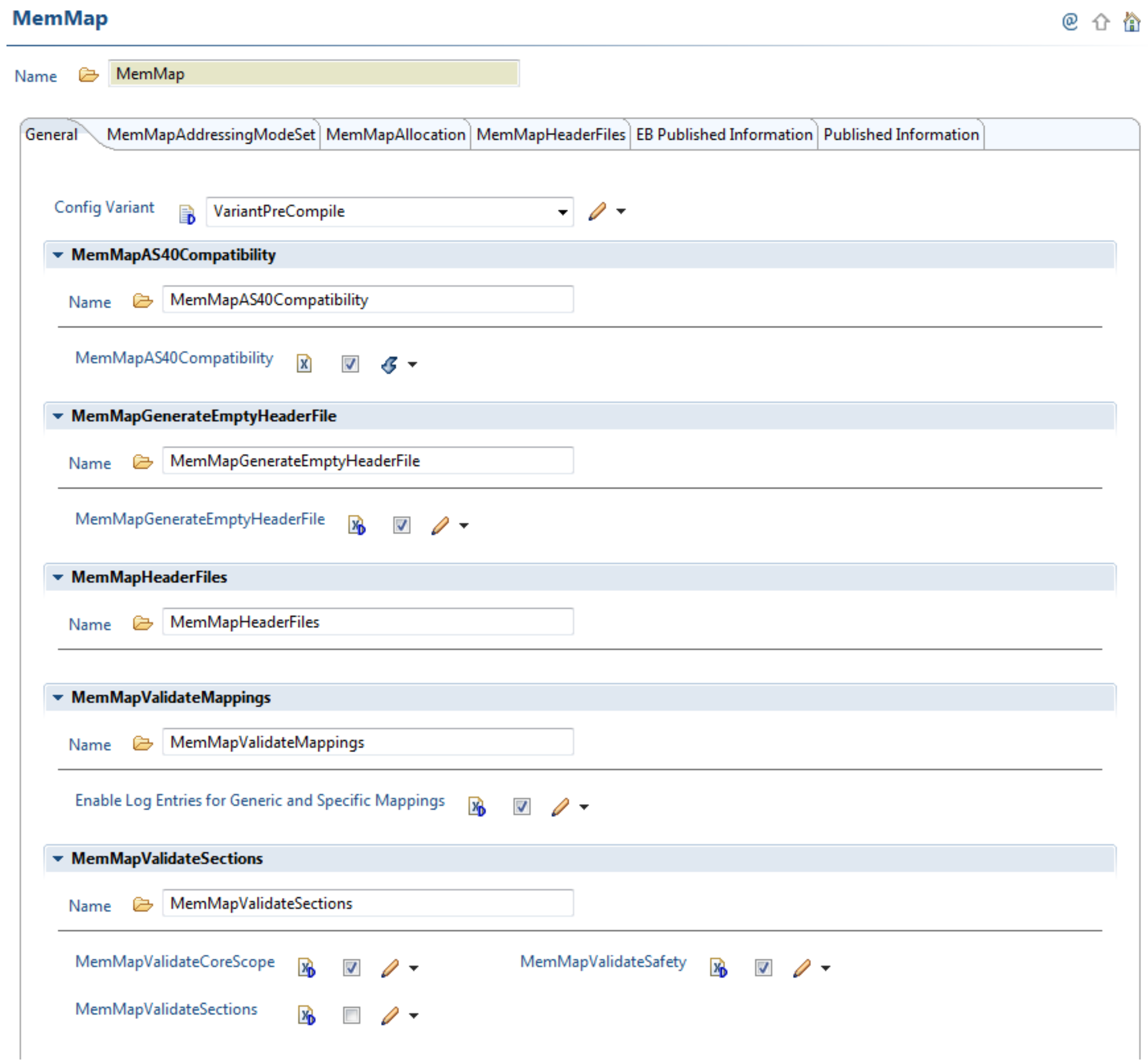
```
<SW-ADDR-METHOD>
        <SHORT-NAME>VAR_INIT_LOCAL</SHORT-NAME>
        <MEMORY-ALLOCATION-KEYWORD-POLICY>
                 ADDR-METHOD-SHORT-NAME-AND-ALIGNMENT
        </MEMORY-ALLOCATION-KEYWORD-POLICY>
        <OPTIONS>
                 <OPTION>coreLocal</OPTION>
        </OPTIONS>
        <SECTION-INITIALIZATION-POLICY>INIT<SECTION-INITIALIZATION-POLICY>
        <SECTION-TYPE>VAR</SECTION-TYPE>
</SW-ADDR-METHOD>
```

MemMap module

- The Memory Mapping (MemMap) module is used to map code and data to specific memory sections via memory mapping files
- The MemMap module generates specific header files which contain memory mapping preprocessor defines for MemorySections and compiler specific instructions


RteImplementationRef
if multiple SwcImplementations have the same SwComponentType
Software Component Description files
import
Basic Software Module Description files
import
MemMap module
generates
MemMap header files
configures

MemMap configuration



MemMap

Name: MemMap

General | MemMapAddressingModeSet | MemMapAllocation | MemMapHeaderFiles | EB Published Information | Published Information

Config Variant: VariantPreCompile

MemMapAS40Compatibility

Name: MemMapAS40Compatibility

MemMapAS40Compatibility ✓

MemMapGenerateEmptyHeaderFile

Name: MemMapGenerateEmptyHeaderFile

MemMapGenerateEmptyHeaderFile ✓

MemMapHeaderFiles

Name: MemMapHeaderFiles

MemMapValidateMappings

Name: MemMapValidateMappings

Enable Log Entries for Generic and Specific Mappings ✓

MemMapValidateSections

Name: MemMapValidateSections

MemMapValidateCoreScope ✓ | MemMapValidateSafety ✓

MemMapValidateSections ☐

MemMap configuration

MemMapAS40Compatibility

- If enabled, MemMap macros for the MemorySection CONFIG_DATA are defined as PREFIX_[START|STOP]_CONFIG_DATA_[ALIGNMENT] and [PREFIX]_[START|STOP]_SEC_CONFIG_DATA_[ALIGNMENT]

```
#elif (defined CANIF_START_CONFIG_DATA_UNSPECIFIED)
  #undef CANIF_START_CONFIG_DATA_UNSPECIFIED
  #undef MEMMAP_ERROR_CANIF
#elif (defined CANIF_STOP_CONFIG_DATA_UNSPECIFIED)
  #undef CANIF_STOP_CONFIG_DATA_UNSPECIFIED
  #undef MEMMAP_ERROR_CANIF
```

- If disabled, MemMap macros for the MemorySection CONFIG_DATA are defined as [PREFIX]_[START|STOP]_SEC_CONFIG_DATA_[ALIGNMENT]

```
#elif (defined CANIF_START_SEC_CONFIG_DATA_UNSPECIFIED)
  #undef CANIF_START_SEC_CONFIG_DATA_UNSPECIFIED
  #undef MEMMAP_ERROR_CANIF
#elif (defined CANIF_STOP_SEC_CONFIG_DATA_UNSPECIFIED)
  #undef CANIF_STOP_SEC_CONFIG_DATA_UNSPECIFIED
  #undef MEMMAP_ERROR_CANIF
```

MemMap configuration

MemMapGenerateEmptyHeaderFile

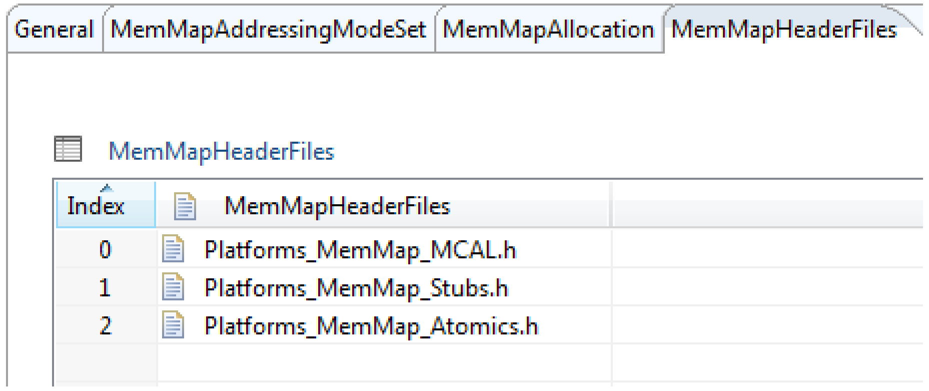
- If enabled, empty MemMap header files will be generated for the BSW and/or SWC implementations that do not have any memory sections defined
  - These files will report a MEMMAP_ERROR if they are included

- If disabled, empty MemMap header files will not be generated for the BSW and/or SWC implementations that do not have any memory sections defined

MemMap configuration

MemMapHeaderFiles

- A list of additional header files included by the generated MemMap.h (in alphabetical order)



Index	MemMapHeaderFiles
0	Platforms_MemMap_MCAL.h
1	Platforms_MemMap_Stubs.h
2	Platforms_MemMap_Atomics.h

```
#ifdef MEMMAP_ERROR
          #include <Platforms_MemMap_Atomics.h>
#endif
#ifdef MEMMAP_ERROR
           #include <Platforms_MemMap_MCAL.h>
#endif
#ifdef MEMMAP_ERROR
          #include <Platforms_MemMap_Stubs.h>
#endif
```

MemMap configuration

MemMapValidateMappings

- If enabled, warnings and errors are reported for the invalid MemMapGenericMappings and MemMapSectionSpecificMappings
- If disabled, invalid MemMapGenericMappings and MemMapSectionSpecificMappings will be silently ignored

Warnings are reported if:

- The SwAddrMethod referenced in MemMapGenericMapping has different attributes as the ones configured in MemMapAddressingModeSet
- The SwAddrMethod referenced in MemMapGenericMapping is not referenced by any of the MemorySection defined in the system description
- The SwAddrMethod referenced in MemMapGenericMapping is from a different package than the one referenced by the MemorySections
- The MemMapAlignmentSelector does not contain the same alignment as the one defined for the MemorySection, for which the memory mapping was created

Errors are reported if:

- More than one MemMapGenericMapping references the same MemMapSwAddressMethodRef
- More than one MemMapSectionSpecificMapping references the same MemMapMemorySectionRef

MemMap configuration

MemMapValidateCoreScope

- If enabled, the usage of coreScope is validated
- If disabled, the usage of coreScope is not validated

Errors are reported if:

- The coreScope is set multiple times
- CoreLocal is not set with the correct SwAddrMethod sectionInitializationPolicy (CLEARED or INIT)
- CoreLocal is not present in both the name and options of the MemorySection and the SwAddrMethod's options

MemMap configuration

MemMapValidateSafety

- If enabled, the usage of safety levels is validated
- If disabled, the usage of safety levels is not validated

Errors are reported if:

- The safety level is set multiple times
- The safety level is not present in both the name and options of the MemorySection and the SwAddrMethod's options

MemMap configuration

MemMapValidateSections

- If enabled, the memory sections will be checked that they are opened and closed in the right order

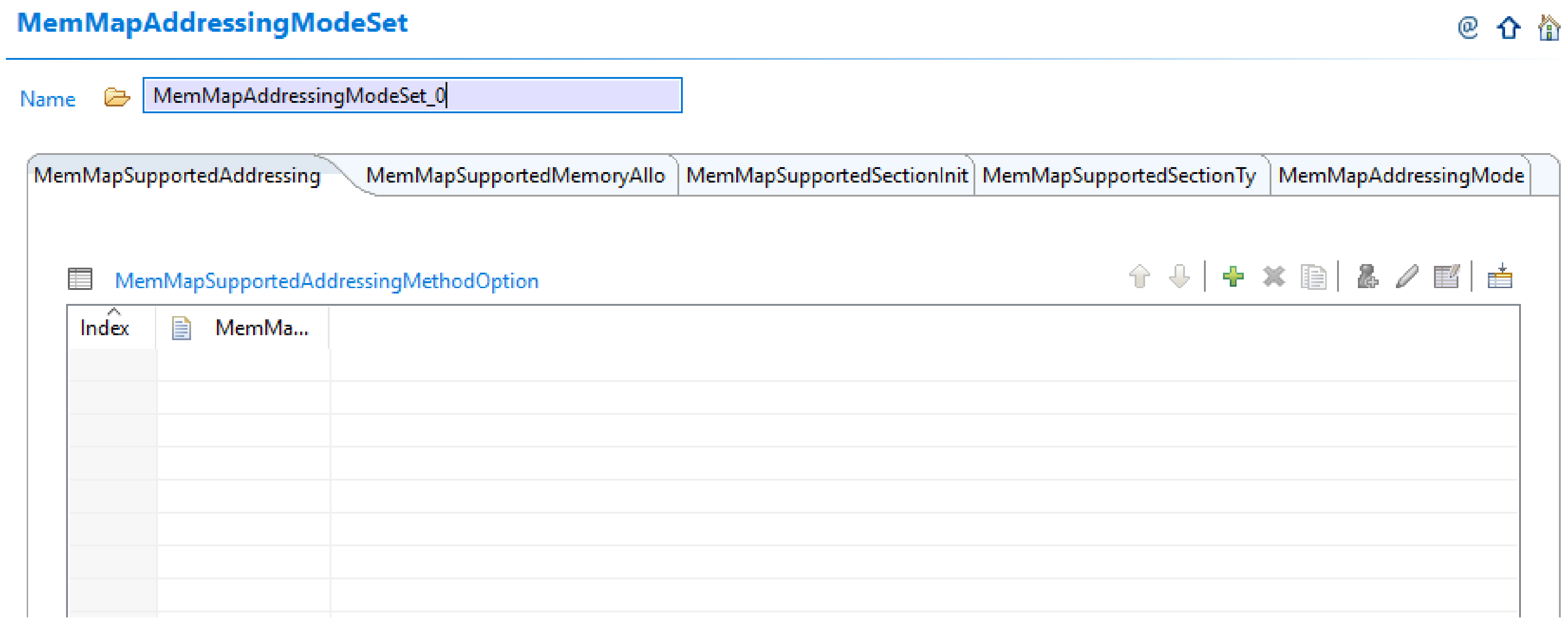
```
#elif (defined CANIF_START_SEC_VAR_INIT_8)
  #ifdef MEMMAP_SECTION_OPENED
    #undef MEMMAP_ERROR_CANIF
    #error Tried to open section CANIF_START_SEC_VAR_INIT_8 within an already open section.
  #else
    #define MEMMAP_SECTION_OPENED


#elif (defined CANIF_STOP_SEC_VAR_INIT_8)
  #if (defined MEMMAP_SECTION_OPENED) && (defined MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8)
    #undef MEMMAP_SECTION_OPENED
    #undef MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8
    #undef CANIF_STOP_SEC_VAR_INIT_8
    #undef MEMMAP_ERROR_CANIF
  #else
    #undef MEMMAP_ERROR_CANIF
    #error Tried to close section CANIF_STOP_SEC_VAR_INIT_8 without prior opening CANIF_START_SEC_VAR_INIT_8.
  #endif
```

MemMap configuration

MemMapAddressingModeSet

- Defines a set of addressing modes which might apply to a SwAddrMethod


MemMapAddressingModeSet

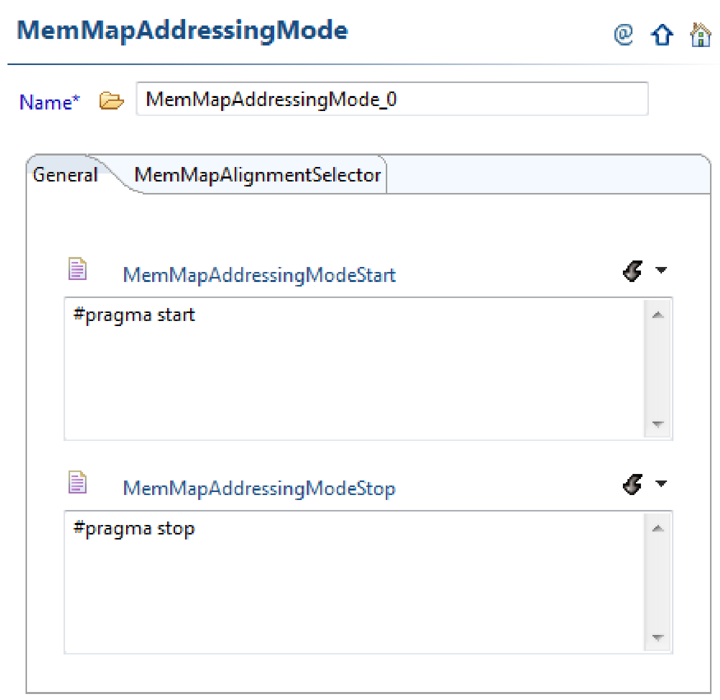
Name: MemMapAddressingModeSet_0

MemMapSupportedAddressing
MemMapSupportedMemoryAllo
MemMapSupportedSectionInit
MemMapSupportedSectionTy
MemMapAddressingMode
MemMapSupportedAddressingMethodOption
Index
MemMa...

MemMap configuration

MemMapAddressingMode

- Defines a addressing mode with a set of #pragma statements implementing for example the start and the stop of a section
- Defines the alignments for which the MemMapAddressingMode applies (mandatory)



MemMapAddressingMode

Name* MemMapAddressingMode_0

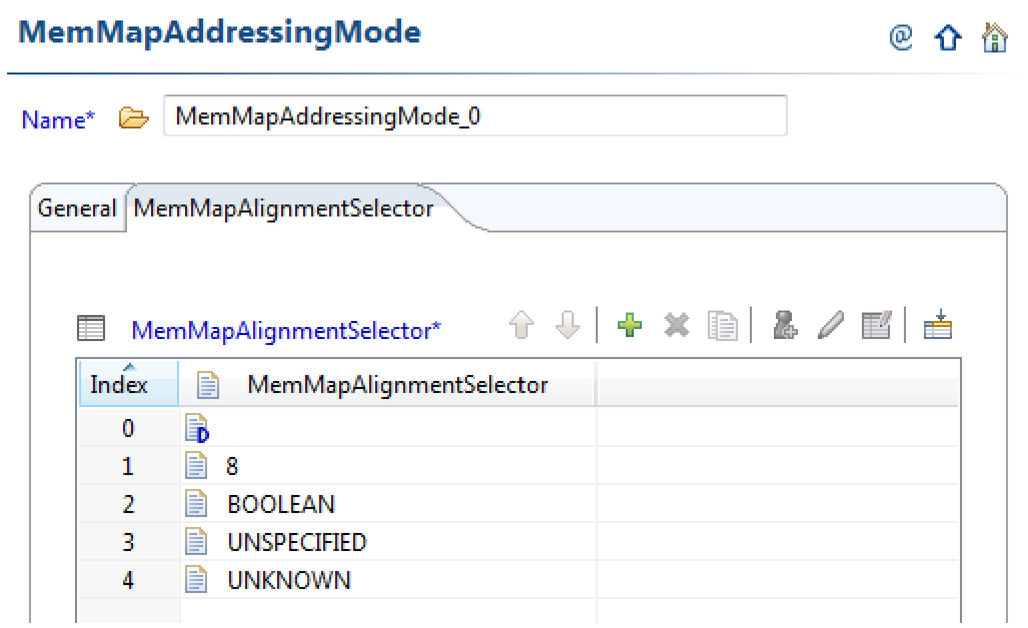
General | MemMapAlignmentSelector

MemMapAddressingModeStart

```
#pragma start
```

MemMapAddressingModeStop

```
#pragma stop
```



MemMapAddressingMode

Name* MemMapAddressingMode_0

General | MemMapAlignmentSelector

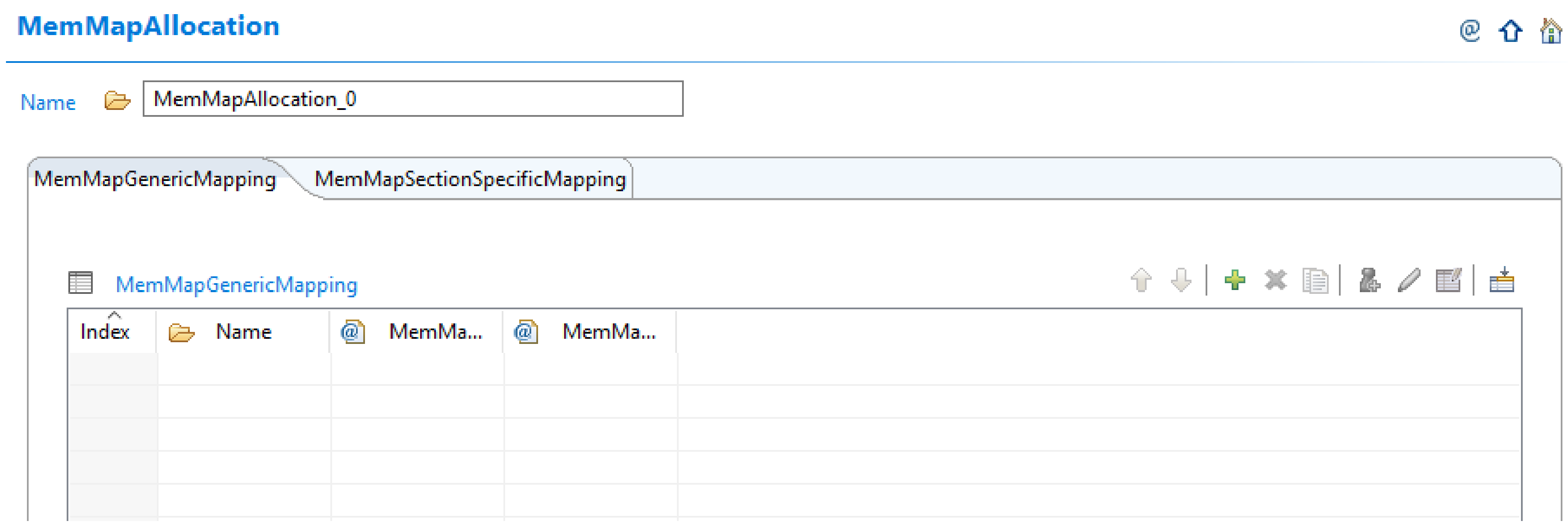
MemMapAlignmentSelector*

Index	MemMapAlignmentSelector
0	
1	8
2	BOOLEAN
3	UNSPECIFIED
4	UNKNOWN

MemMap configuration

MemMapAllocation

- Defines the generic or specific mappings of the MemMapAddressingModeSet to a SwAddrMethod



MemMapAllocation

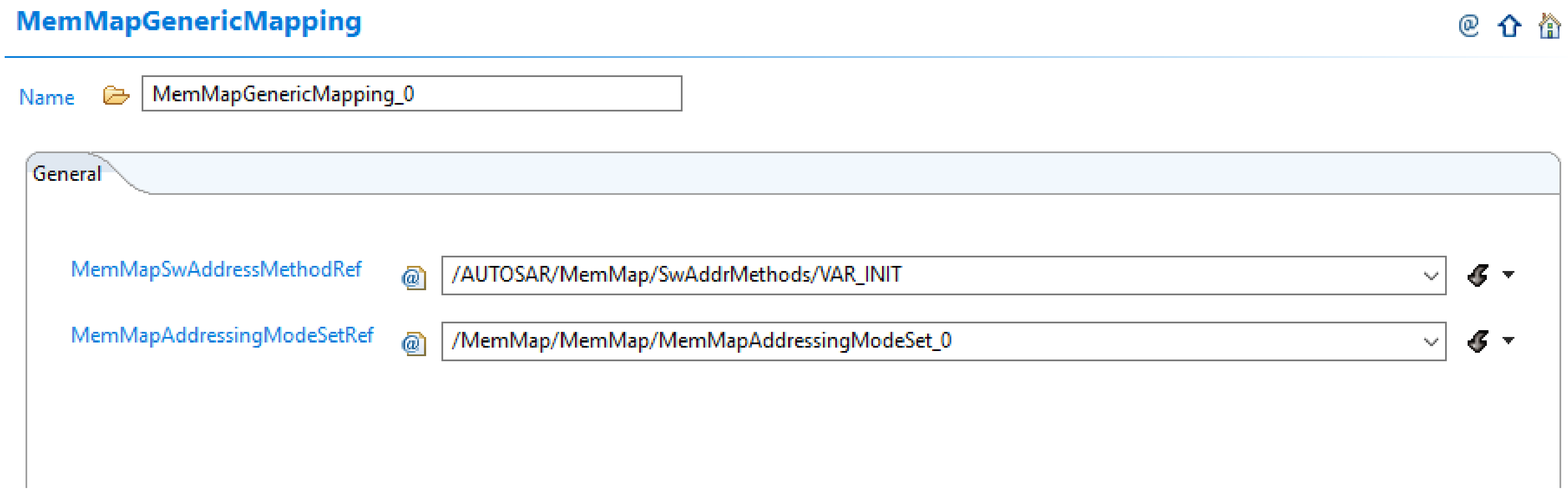
Name: MemMapAllocation_0

Index	Name	MemMa...	MemMa...

MemMap configuration

MemMapGenericMapping

- Defines which SwAddrMethod is implemented with which MemMapAddressingModeSet



MemMapGenericMapping

Name: MemMapGenericMapping_0

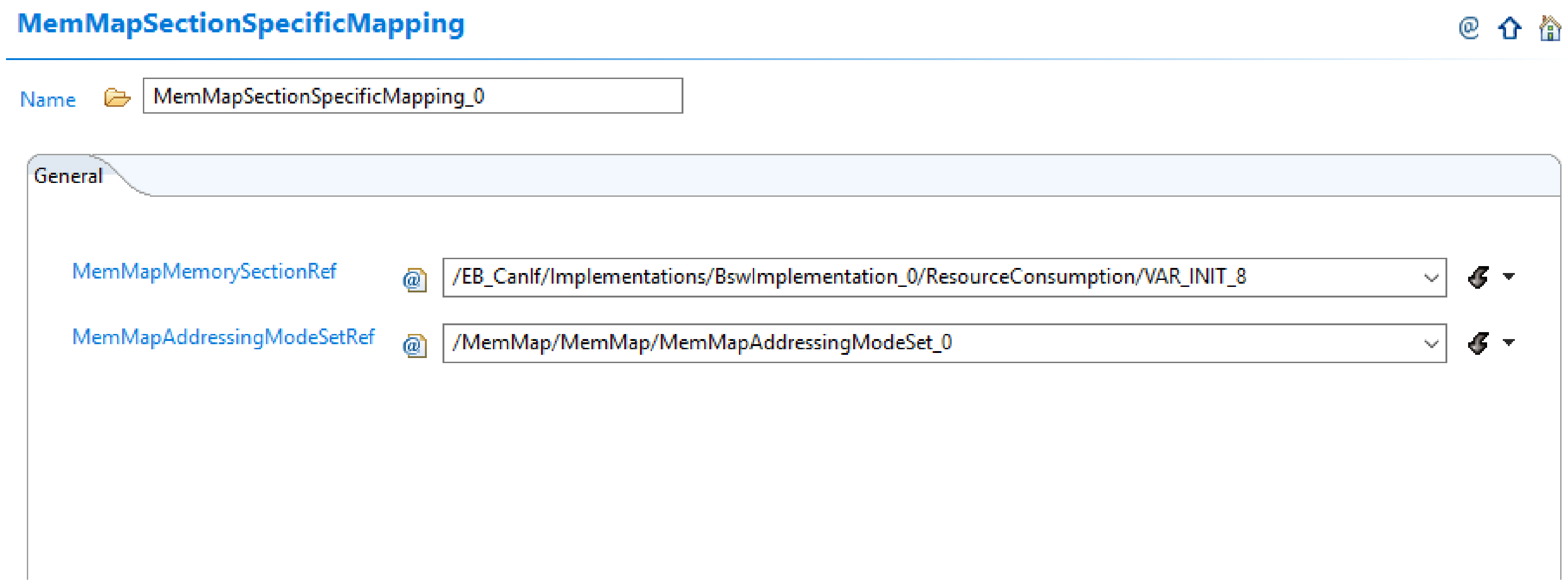
General

MemMapSwAddressMethodRef	/AUTOSAR/MemMap/SwAddrMethods/VAR_INIT
MemMapAddressingModeSetRef	/MemMap/MemMap/MemMapAddressingModeSet_0

MemMap configuration

MemMapSectionSpecificMapping

- Defines which MemorySection of a BSW Module or a Software Component is implemented with which MemMapAddressingModeSet



MemMapSectionSpecificMapping

Name: MemMapSectionSpecificMapping_0

General

MemMapMemorySectionRef: /EB_CanIf/Implementations/BswImplementation_0/ResourceConsumption/VAR_INIT_8

MemMapAddressingModeSetRef: /MemMap/MemMap/MemMapAddressingModeSet_0

Generated MemMap header files

MemMap.h

- The header file contains
  - Inclusion of all <BSW>_MemMap.h files (compatibility reasons)

- Code snippet:

```
#ifdef MEMMAP_ERROR
  #include <Atomics_MemMap.h>
#endif

#ifdef MEMMAP_ERROR
  #include <Base_MemMap.h>
#endif

#ifdef MEMMAP_ERROR
  #include <BswM_MemMap.h>
#endif

#ifdef MEMMAP_ERROR
  #include <Can_MemMap.h>
#endif

#ifdef MEMMAP_ERROR
  #include <CanIf_MemMap.h>
#endif
```

Generated MemMap header files

<BSW>_MemMap.h / <SWC>_MemMap.h

- The header file contains
  - The memory allocation keywords for the MemorySection
  - Compiler specific instructions, if valid generic or specific mappings are created

Generated MemMap header files

Code snippet CanIf_MemMap.h

- Without defining a generic or specific mapping

```
#elif (defined CANIF_START_SEC_VAR_INIT_8)
 #ifdef MEMMAP_SECTION_OPENED
  #undef MEMMAP_ERROR_CANIF
  #error Tried to open section CANIF_START_SEC_VAR_INIT_8 within an already open section.
 #else
  #define MEMMAP_SECTION_OPENED
  #define MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8
  #undef CANIF_START_SEC_VAR_INIT_8
  #undef MEMMAP_ERROR_CANIF
 #endif
#elif (defined CANIF_STOP_SEC_VAR_INIT_8)
 #if (defined MEMMAP_SECTION_OPENED) && (defined MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8)
  #undef MEMMAP_SECTION_OPENED
  #undef MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8
  #undef CANIF_STOP_SEC_VAR_INIT_8
  #undef MEMMAP_ERROR_CANIF
 #else
  #undef MEMMAP_ERROR_CANIF
  #error Tried to close section CANIF_STOP_SEC_VAR_INIT_8 without prior opening CANIF_START_SEC_VAR_INIT_8.
 #endif
```

Generated MemMap header files

Code snippet CanIf_MemMap.h

- With a defined generic or specific mapping

```
#elif (defined CANIF_START_SEC_VAR_INIT_8)
 #ifdef MEMMAP_SECTION_OPENED
  #undef MEMMAP_ERROR_CANIF
  #error Tried to open section CANIF_START_SEC_VAR_INIT_8 within an already open section.
 #else
  #pragma start
  #define MEMMAP_SECTION_OPENED
  #define MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8
  #undef CANIF_START_SEC_VAR_INIT_8
  #undef MEMMAP_ERROR_CANIF
 #endif
#elif (defined CANIF_STOP_SEC_VAR_INIT_8)
 #if (defined MEMMAP_SECTION_OPENED) && (defined MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8)
  #pragma stop
  #undef MEMMAP_SECTION_OPENED
  #undef MEMMAP_SECTION_OPENED_STARTSEC_VAR_INIT_8
  #undef CANIF_STOP_SEC_VAR_INIT_8
  #undef MEMMAP_ERROR_CANIF
 #else
  #undef MEMMAP_ERROR_CANIF
  #error Tried to close section CANIF_STOP_SEC_VAR_INIT_8 without prior opening CANIF_START_SEC_VAR_INIT_8.
 #endif
```

Get in touch!

sales@elektrobit.com
www.elektrobit.com