VECTOR

MICROSAR Memory Mapping

Technical Reference

Version 1.3.0

Authors	Eugen Stripling
Status	Released

Document Information

History

Author	Date	Version	Remarks
Eugen Stripling	2013-04-12	1.00.00	Creation - ESCAN00064655
Eugen Stripling	2013-05-29	1.00.01	Some typos corrected
Eugen Stripling	2016-09-27	1.01.00	FEATC-317 / FEAT-2002: 64-bit memory section keywords added
Eugen Stripling	2017-01-16	1.02.00	ESCAN00093572: 2.1.4 chapter added
Eugen Stripling	2017-07-18	1.03.00	Chapters 2.1 and 2.1.4 adapted due to ESCAN00095756

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_MemoryMapping.pdf	1.4.0
[2]	AUTOSAR	AUTOSAR_SWS_MemoryMapping.pdf	4.2.2
[3]	AUTOSAR	AUTOSAR_SWS_CompilerAbstraction.pdf	3.2.0

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

Illustrations

Tables

1 Introduction

This document gives an overview about the functionality of memory mapping as specified by AUTOSAR according to the Vector specific implementation (MICROSAR).

2 Functional Description

In the following chapters the following two expressions are used:

- **memory section keyword**

and

- **memory area**.

A **memory section keyword** describes a memory section according to AUTOSAR symbolically. All available corresponding keywords are described in chapter 3.2. By **memory area** a memory section in hardware at all is meant.

By using of expression **build-toolchain** the functionality covered by a compiler, linker and locator is meant.

In addition the abbreviations `MSN`[1] and `Msn`[1] respectively are used as placeholders for the short name of any BSW module. However AUTOSAR itself prefers to use the abbreviations `MIP`[2] and `Mip`[2] respectively. Both abbreviations mean the same.

2.1 Memory section keywords

The keywords of memory sections of all BSW modules which are implemented according to AUTOSAR 4.0.3 are located in the file `MemMap.h`. BSW modules implemented according to AUTOSAR 4.0.3 include this general file. However BSW modules which are implemented according to AUTOSAR 4.2.2 have their own specific `<Msn>_MemMap.h` file and include generally only their own file. For more information about usage of BSW modules which are implemented according to AUTOSAR 4.2.2 please see chapter 2.1.4.

The file `Compiler_Cfg.h` contains additional compiler specific keywords. These keywords are used to describe the memory location of variables, constants and pointers. In case of pointers additional keywords are defined to describe the memory location the pointer points to.

All files are provided as templates and may be modified and extended by the user if required. The files `MemMap.h` and `Compiler_Cfg.h` are named `_MemMap.h` and `_Compiler_Cfg.h` initially. Before using both files have to be renamed to `MemMap.h` and to `Compiler_Cfg.h` respectively.

The following figure shows descriptively the relationship between the mentioned files.

[1] Module Short Name
[2] Module Implementation Prefix

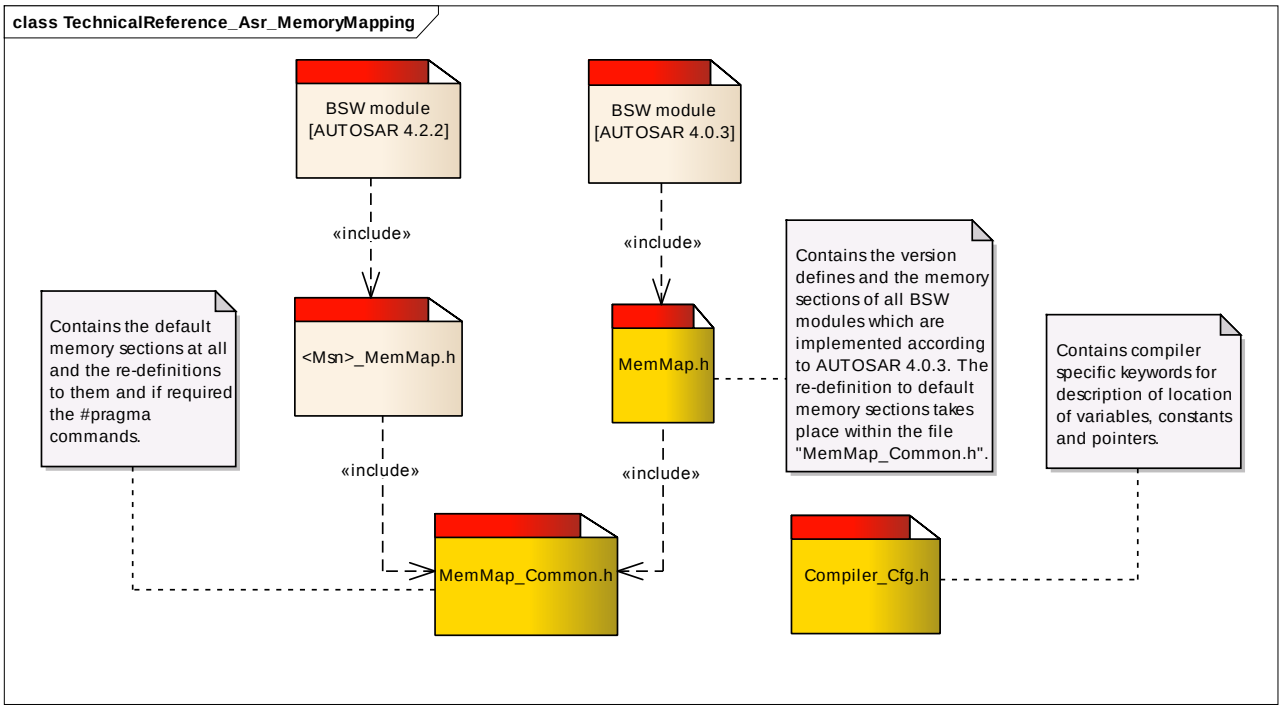

Figure 2-1 Files: MemMap.h, MemMap_Common.h and Compiler_Cfg.h

Each BSW module has its own set of compiler specific and memory section keywords with the pre-fix <MSN>. By default all the memory section keywords are mapped to the same default memory section keyword defined in file `MemMap_Common.h`. If required the default mapping can be modified by the BSW integrator. Detailed information concerning the provided memory section keywords can be found in the chapter 3.2. For detailed information about the compiler specific keywords please see the chapter 3.3.

2.1.1 Declaration of code and data segments in AUTOSAR

The declaration of all kinds of variables including constants, pointers and the code itself is performed in the following way.

1. **Opening of the memory section:** The memory section where e.g. a variable shall be stored in is opened by defining of the corresponding memory section keyword (s. chapter 3.2).
2. **Including of file `MemMap.h/<Msn>_MemMap.h`:** In this step the previously defined BSW module specific memory section keyword is redefined to the default keyword of the corresponding memory section. Additionally if specified the corresponding `#pragma`-command is activated. This command tells the build-toolchain to put the following data or code segments to intended memory area.
3. **Declaration of data and code segments:** In this step any kind of data or code itself is declared.
4. **Closing of the memory section:** In the last step the previously opened memory section is closed by defining of the STOP-keyword of the corresponding memory section keyword and by including of file `MemMap.h/<Msn>_MemMap.h` once again.

By this step the build-toolchain shall be told to switch back to the default memory section.

The following example shall clarify the mentioned steps.

Example

…

```
1. <MSN>_START_SEC_CONST_8BIT
2. #include "MemMap.h"

3. CONST(uint8, <MSN>_CONST) <Msn>_Constant = 0xFAu;

4. <MSN>_STOP_SEC_CONST_8BIT
4. #include "MemMap.h
```

…

2.1.2 Mapping of code and data segments to a dedicated memory area

In order to map a code segment or a data segment to a dedicated memory area you need to follow the following four steps:

1. **Selection of memory section keywords:** In this step the keywords of memory sections which are intended to be mapped to a dedicated memory area have to be picked out. All available memory section keywords are mentioned in chapter 3.2.
2. **Adaption of the linker / locator command file:** In this step the memory areas which are intended to be used for certain segments of code or data have to be defined in the linker / locator command file. By this way the defined memory areas are labeled with unique keywords. These keywords in combination with a `#pragma`-command can be used later to tell the build-toolchain to put code or data segments to a specific memory area.
3. **Adding of `#pragma`-commands to selected memory section keywords:** In this step `#pragma`-commands together with memory area keywords defined in step 2 are added to the corresponding memory section keywords picked out in step 1. The adaptions are performed in file `MemMap_Common.h` directly.
4. **Adaption of compiler specific keywords:** This step is optional. The applying or not applying of this step depends on the used compiler and the memory model of used hardware. If required, in this step the compiler specific keywords which are required for correct access of code and data have to be adapted according to the corresponding memory mapping (steps: 1 till 3). For more information see chapter 3.3.

Caution
In case of using a `#pragma`-command at any `START_SEC`-keyword ensure that at the corresponding `STOP_SEC`-keyword the `#pragma`-command is defined which switches back to the default memory section. Otherwise the once opened memory section won't be closed and all data segments which follow the `#pragma`-command will be mapped to the still opened memory section unintentionally.

Note
This depends on the used compiler e.g. a DiabData-compiler does not need any `#pragma`-command at the `STOP_SEC`-keyword to switch back to the default memory section. Therefrom please check the technical reference of used compiler.

For an example please see the following chapter.

2.1.3 Example: Mapping of data to a post-build memory section

There are three post-build sections: two for ROM and one for RAM. The related keywords are:

- `START_SEC_PBCFG_GLOBALROOT` and `START_SEC_CONST_PBCFG` (ROM)
- `START_SEC_VAR_PBCFG` (RAM)

The corresponding `STOP`-sections are

- `STOP_SEC_CONST` (ROM)
- `STOP_SEC_VAR` (RAM)

At the position of mentioned keywords in file `MemMap_Common.h` the appropriate compiler specific `#pragma`-commands have to be added in order to tell the compiler to put the related data segments to the intended memory area. In case of platform V850 and GHS compiler see the following example.

Example The following example applies on platform V850 and GHS-compiler.

1. Labels for memory areas which are intended to be used are defined in the linker / locator command file:

```
…
 $(ECHO) "  .pb_ecum_global_root 0x001C0000  :>.    /* Start post-build ROM area  */          " >> $@; \
 $(ECHO) "  .pb_tscpb_data align(4)                :>.    /* post-build ROM area */                 " >> $@; \
 $(ECHO) "  .pb_ram            0xfedfa000          :>.    /*  post-build RAM area */                " >> $@; \
…
```

2. The defined labels can now be used for the `#pragma`-commands at the corresponding memory section keywords in file `MemMap_Common.h`:

```
…
#elif defined START_SEC_CONST_PBCFG
# define CONST_SEC_OPEN
/* Enter here a #pragma command for opening the specified section */
# pragma ghs section rodata=".pb_tscpb_data"
# undef START_SEC_CONST_PBCFG
# undef MEMMAP_ERROR
# ifdef MEMMAP_DETECT_ANY_ERROR
#  error "The DEFINE: "START_SEC_CONST_PBCFG" is the obsolete one!"
# endif

#elif defined START_SEC_PBCFG_GLOBALROOT
# define CONST_SEC_OPEN
/* Enter here a #pragma command for opening the specified section */
# pragma ghs section rodata=".pb_ecum_global_root"
# undef START_SEC_PBCFG_GLOBALROOT
# undef MEMMAP_ERROR
# ifdef MEMMAP_DETECT_ANY_ERROR
#  error "The DEFINE: "START_SEC_PBCFG_GLOBALROOT" is the obsolete one!"
# endif
…
#elif defined STOP_SEC_CONST
# if defined CONST_SEC_OPEN
#  undef CONST_SEC_OPEN
/* Enter here a #pragma command for closing the specified section */
/* This should switch back to the default section */
#  pragma ghs section rodata=default
#  undef STOP_SEC_CONST
#  undef MEMMAP_ERROR
# else
#  error "MemMap Common.h: Const section closed but not opened one. Please close
a const section [== usage of *_STOP_SEC_CONST_*] only if you have opened one [==
usage of *_START_SEC_CONST_*]!"
# endif
…
#elif defined START_SEC_VAR_PBCFG
# define VAR_SEC_OPEN
/* Enter here a #pragma command for opening the specified section */
```

```
        # pragma ghs section bss=".pb_ram"
        # undef START_SEC_VAR_PBCFG
        # undef MEMMAP_ERROR
        # ifdef MEMMAP_DETECT_ANY_ERROR
        #  error "The DEFINE: "START_SEC_VAR_PBCFG" is the obsolete one!"
        # endif
…
        #elif defined STOP_SEC_VAR
        # if defined VAR_SEC_OPEN
        #  undef VAR_SEC_OPEN
        /* Enter here a #pragma command for closing the specified section */
        /* This should switch back to the default section */
        #  pragma ghs section bss=default
        #  undef STOP_SEC_VAR
        #  undef MEMMAP_ERROR
        # else
        #  error "MemMap Common.h: Var section closed but not opened one. Please close a
        var section [== usage of *_STOP_SEC_VAR_*] only if you have opened one [== usage
        of *_START_SEC_VAR_*]!"
        # endif
…
```

2.1.4 Usage of AUTOSAR 4.2.2 BSW module

The Vector specific implementation of memory mapping [MICROSAR] complies with both AUTOSAR 4.0.3 and AUTOSAR 4.2.2. Due to fact that the AUTOSAR 4.2.2 defines that each BSW module must include an own memory mapping header file [see SWS_MemMap_00032] one manual adaption is required for BSW modules which are implemented according to AUTOSAR 4.2.2 and are intended to be used in combination with the Vector specific implementation of memory mapping especially in case of using of default memory sections. The required adaption is described below.

- Add the following code at the very bottom to the BSW module specific memory mapping header file `<Msn>_MemMap.h`: `#include "MemMap_Common.h"` [see Figure 2-1].

3 Appendix

3.1 Sub-keywords used in the memory section and compiler specific keywords

The memory section and compiler specific keywords mentioned in the following chapters 3.2 and 3.3 generally have self-explanatory names. Despite this fact the following table gives more information about used sub-keywords.

Sub-keyword	Kind of memory section which is meant
CODE	Memory section for code (ROM)
CONST	Memory section for constant data (ROM)
VAR	Memory section for variables (RAM)
FAST	Memory section with fast read and write accesses (ROM / RAM)
ISR	Memory section for interrupt service routines (generally ROM)
8BIT (since ASR 4.0.3: 8)	Memory section for one byte data (ROM / RAM)
16BIT (since ASR 4.0.3: 16)	Memory section for two bytes data (ROM / RAM)
32BIT (since ASR 4.0.3: 32)	Memory section for four bytes data (ROM / RAM)
64BIT (since ASR 4.0.3: 64)	Memory section for eight bytes data (ROM / RAM)
UNSPECIFIED	Memory section for data (ROM / RAM) which type is variable, depends on configuration and is determined only at compile time (e.g. enumeration, boolean, structure)
PBCFG_GLOBALROOT	The top of the memory section for post-build ROM data. Only the table EcuM_GlobalConfigRoot shall be placed into this section.
(CONST_)PBCFG	Memory section for post-build ROM data. NOTE: The data which resides in this section may change at post-build time.
INIT	Memory section for pre-initialized variables (RAM) e.g. by the start-up code
NOINIT (since ASR 4.0.3: NO_INIT)	Memory section for not pre-initialized variables (RAM)
ZERO_INIT (since ASR 4.0.3: CLEARED)	Memory section for pre-initialized variables with zeros
VAR_PBCFG	Memory section for post-build RAM variables NOTE: The allocation of this memory area can change at post-build time.

Table 3-1 Explanation of sub-keywords used in the memory section and compiler specific keywords

3.2 Memory section keywords

The following tables listed below give an overview about pre-defined memory section keywords used by the BSW modules.

Note
The memory section keywords listed below are the default ones. Apart from these ones if required each BSW module may define further own ones. Please see the files `<Msn>_MemMap.h` [in case of an AUTOSAR 4.2.2 BSW module] or `MemMap.h` [in case of an AUTOSAR 4.0.3 BSW module] included in your delivery.

3.2.1 Memory section keywords for code

BSW specific memory section keyword	Default memory section keyword
<MSN>_START_SEC_CODE	START_SEC_CODE
<MSN>_START_SEC_CODE_FAST	START_SEC_CODE_FAST
<MSN>_START_SEC_CODE_ISR	START_SEC_CODE_ISR

Table 3-2 Memory sections for code

Note
For each `START`-section keyword a corresponding `STOP`-section keyword is defined e.g. `<MSN>_STOP_SEC_CODE`. By default all these `STOP`-section keywords are mapped to the same keyword `STOP_SEC_CODE`. Hence use a `#pragma` command to switch back to the default section for code.

3.2.2 Memory section keywords for constants

BSW specific memory section keyword	Default memory section keyword
<MSN>_START_SEC_CONST_8BIT	START_SEC_CONST_8BIT
<MSN>_START_SEC_CONST_16BIT	START_SEC_CONST_16BIT
<MSN>_START_SEC_CONST_32BIT	START_SEC_CONST_32BIT
<MSN>_START_SEC_CONST_64BIT	START_SEC_CONST_64BIT
<MSN>_START_SEC_CONST_UNSPECIFIED	START_SEC_CONST_UNSPECIFIED
<MSN>_START_SEC_FAST_CONST_8BIT	START_SEC_FAST_CONST_8BIT
<MSN>_START_SEC_FAST_CONST_16BIT	START_SEC_FAST_CONST_16BIT
<MSN>_START_SEC_FAST_CONST_32BIT	START_SEC_FAST_CONST_32BIT
<MSN>_START_SEC_FAST_CONST_64BIT	START_SEC_FAST_CONST_64BIT
<MSN>_START_SEC_FAST_CONST_UNSPECIFIED	START_SEC_FAST_CONST_UNSPECIFIED
<MSN>_START_SEC_PBCFG_GLOBALROOT	START_SEC_PBCFG_GLOBALROOT
<MSN>_START_SEC_PBCFG	START_SEC_CONST_PBCFG

Table 3-3 Memory sections for constants

Note
For each START-section keyword a corresponding STOP-section keyword is defined e.g. `<MSN>_STOP_SEC_CONST_32BIT`. By default all these STOP-section keywords are mapped to the same keyword `STOP_SEC_CONST`. Hence use a `#pragma` command to switch back to the default section for constants.

The mentioned BSW module specific keywords conform to AUTOSAR releases 3.x.x till 4.0.1. Since release 4.0.3 the naming convention of some keywords has changed. The following table shows the related changes.

Keyword in AUTOSAR 3.x.x till 4.0.1	Keyword in AUTOSAR 4.0.3
<MSN>_START_SEC_CONST_8BIT	<MSN>_START_SEC_CONST_8
<MSN>_START_SEC_CONST_16BIT	<MSN>_START_SEC_CONST_16
<MSN>_START_SEC_CONST_32BIT	<MSN>_START_SEC_CONST_32
<MSN>_START_SEC_CONST_64BIT	<MSN>_START_SEC_CONST_64
<MSN>_START_SEC_FAST_CONST_8BIT	<MSN>_START_SEC_FAST_CONST_8
<MSN>_START_SEC_FAST_CONST_16BIT	<MSN>_START_SEC_FAST_CONST_16
<MSN>_START_SEC_FAST_CONST_32BIT	<MSN>_START_SEC_FAST_CONST_32
<MSN>_START_SEC_FAST_CONST_64BIT	<MSN>_START_SEC_FAST_CONST_64

Table 3-4 Change of memory section keywords for constants in ASR 4.0.3

Note
Although names of some keywords have changed most of ASR 4.0.3 BSW modules provided by Vector Informatik GmbH still use the old naming convention of memory section keywords. The BSW module `MemMap` at all provided by Vector Informatik GmbH accepts both naming conventions.

3.2.3 Memory section keywords for variables

BSW specific memory section keyword	Default memory section keyword
<MSN>_START_SEC_VAR_INIT_8BIT	START_SEC_VAR_INIT_8BIT
<MSN>_START_SEC_VAR_INIT_16BIT	START_SEC_VAR_INIT_16BIT
<MSN>_START_SEC_VAR_INIT_32BIT	START_SEC_VAR_INIT_32BIT
<MSN>_START_SEC_VAR_INIT_64BIT	START_SEC_VAR_INIT_64BIT
<MSN>_START_SEC_VAR_INIT_UNSPECIFIED	START_SEC_VAR_INIT_UNSPECIFIED

BSW specific memory section keyword	Default memory section keyword
<MSN>_START_SEC_VAR_NOINIT_8BIT	START_SEC_VAR_NOINIT_8BIT
<MSN>_START_SEC_VAR_NOINIT_16BIT	START_SEC_VAR_NOINIT_16BIT
<MSN>_START_SEC_VAR_NOINIT_32BIT	START_SEC_VAR_NOINIT_32BIT
<MSN>_START_SEC_VAR_NOINIT_64BIT	START_SEC_VAR_NOINIT_64BIT
<MSN>_START_SEC_VAR_NOINIT_UNSPECIFIED	START_SEC_VAR_NOINIT_UNSPECIFIED
<MSN>_START_SEC_VAR_PBCFG	START_SEC_VAR_PBCFG
<MSN>_START_SEC_VAR_ZERO_INIT_8BIT	START_SEC_VAR_ZERO_INIT_8BIT
<MSN>_START_SEC_VAR_ZERO_INIT_16BIT	START_SEC_VAR_ZERO_INIT_16BIT
<MSN>_START_SEC_VAR_ZERO_INIT_32BIT	START_SEC_VAR_ZERO_INIT_32BIT
<MSN>_START_SEC_VAR_ZERO_INIT_64BIT	START_SEC_VAR_ZERO_INIT_64BIT
<MSN>_START_SEC_VAR_ZERO_INIT_UNSPECIFIED	START_SEC_VAR_ZERO_INIT_UNSPECIFIED
<MSN>_START_SEC_VAR_FAST_INIT_8BIT	START_SEC_VAR_FAST_INIT_8BIT
<MSN>_START_SEC_VAR_FAST_INIT_16BIT	START_SEC_VAR_FAST_INIT_16BIT
<MSN>_START_SEC_VAR_FAST_INIT_32BIT	START_SEC_VAR_FAST_INIT_32BIT
<MSN>_START_SEC_VAR_FAST_INIT_64BIT	START_SEC_VAR_FAST_INIT_64BIT
<MSN>_START_SEC_VAR_FAST_INIT_UNSPECIFIED	START_SEC_VAR_FAST_INIT_UNSPECIFIED
<MSN>_START_SEC_VAR_FAST_NOINIT_8BIT	START_SEC_VAR_FAST_NOINIT_8BIT
<MSN>_START_SEC_VAR_FAST_NOINIT_16BIT	START_SEC_VAR_FAST_NOINIT_16BIT
<MSN>_START_SEC_VAR_FAST_NOINIT_32BIT	START_SEC_VAR_FAST_NOINIT_32BIT
<MSN>_START_SEC_VAR_FAST_NOINIT_64BIT	START_SEC_VAR_FAST_NOINIT_64BIT
<MSN>_START_SEC_VAR_FAST_NOINIT_UNSPECIFIED	START_SEC_VAR_FAST_NOINIT_UNSPECIFIED
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_8BIT	START_SEC_VAR_FAST_ZERO_INIT_8BIT
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_16BIT	START_SEC_VAR_FAST_ZERO_INIT_16BIT
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_32BIT	START_SEC_VAR_FAST_ZERO_INIT_32BIT
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_64BIT	START_SEC_VAR_FAST_ZERO_INIT_64BIT
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_UNSPECIFIED	START_SEC_VAR_FAST_ZERO_INIT_UNSPECIFIED

Table 3-5 Memory sections for variables

Note
For each START-section keyword a corresponding `STOP`-section keyword is defined e.g. `<MSN>_STOP_SEC_VAR_INIT_32BIT`. By default all these `STOP`-section keywords are mapped to the same keyword `STOP_SEC_VAR`. Hence use a `#pragma` command to switch back to the default section for variables.

The mentioned BSW module specific keywords conform to AUTOSAR releases 3.x.x till 4.0.1. Since release 4.0.3 the naming convention of some keywords has changed. The following table shows the related changes.

Keyword in AUTOSAR 3.x.x till 4.0.1	Keyword in AUTOSAR 4.0.3
<MSN>_START_SEC_VAR_INIT_8BIT	<MSN>_START_SEC_VAR_INIT_8
<MSN>_START_SEC_VAR_INIT_16BIT	<MSN>_START_SEC_VAR_INIT_16
<MSN>_START_SEC_VAR_INIT_32BIT	<MSN>_START_SEC_VAR_INIT_32
<MSN>_START_SEC_VAR_INIT_64BIT	<MSN>_START_SEC_VAR_INIT_64
<MSN>_START_SEC_VAR_NOINIT_8BIT	<MSN>_START_SEC_VAR_NO_INIT_8
<MSN>_START_SEC_VAR_NOINIT_16BIT	<MSN>_START_SEC_VAR_NO_INIT_16
<MSN>_START_SEC_VAR_NOINIT_32BIT	<MSN>_START_SEC_VAR_NO_INIT_32
<MSN>_START_SEC_VAR_NOINIT_64BIT	<MSN>_START_SEC_VAR_NO_INIT_64
<MSN>_START_SEC_VAR_NOINIT_UNSPECIFIED	<MSN>_START_SEC_VAR_NO_INIT_UNSPECIFIED
<MSN>_START_SEC_VAR_ZERO_INIT_8BIT	<MSN>_START_SEC_VAR_CLEARED_8
<MSN>_START_SEC_VAR_ZERO_INIT_16BIT	<MSN>_START_SEC_VAR_CLEARED_16
<MSN>_START_SEC_VAR_ZERO_INIT_32BIT	<MSN>_START_SEC_VAR_CLEARED_32
<MSN>_START_SEC_VAR_ZERO_INIT_64BIT	<MSN>_START_SEC_VAR_CLEARED_64
<MSN>_START_SEC_VAR_ZERO_INIT_UNSPECIFIED	<MSN>_START_SEC_VAR_CLEARED_UNSPECIFIED
<MSN>_START_SEC_VAR_FAST_INIT_8BIT	<MSN>_START_SEC_VAR_FAST_INIT_8
<MSN>_START_SEC_VAR_FAST_INIT_16BIT	<MSN>_START_SEC_VAR_FAST_INIT_16
<MSN>_START_SEC_VAR_FAST_INIT_32BIT	<MSN>_START_SEC_VAR_FAST_INIT_32
<MSN>_START_SEC_VAR_FAST_INIT_64BIT	<MSN>_START_SEC_VAR_FAST_INIT_64
<MSN>_START_SEC_VAR_FAST_INIT_UNSPECIFIED	<MSN>_START_SEC_VAR_FAST_INIT_UNSPECIFIED
<MSN>_START_SEC_VAR_FAST_NOINIT_8BIT	<MSN>_START_SEC_VAR_FAST_NO_INIT_8
<MSN>_START_SEC_VAR_FAST_NOINIT_16BIT	<MSN>_START_SEC_VAR_FAST_NO_INIT_16
<MSN>_START_SEC_VAR_FAST_NOINIT_32BIT	<MSN>_START_SEC_VAR_FAST_NO_INIT_32
<MSN>_START_SEC_VAR_FAST_NOINIT_64BIT	<MSN>_START_SEC_VAR_FAST_NO_INIT_64
<MSN>_START_SEC_VAR_FAST_NOINIT_UNSPECIFIED	<MSN>_START_SEC_VAR_FAST_NO_INIT_UNSPECIFIED

Keyword in AUTOSAR 3.x.x till 4.0.1	Keyword in AUTOSAR 4.0.3
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_8BIT	<MSN>_START_SEC_VAR_FAST_ CLEARED_8
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_16BIT	<MSN>_START_SEC_VAR_FAST_CLEARED_16
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_32BIT	<MSN>_START_SEC_VAR_FAST_CLEARED_32
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_64BIT	<MSN>_START_SEC_VAR_FAST_CLEARED_64
<MSN>_START_SEC_VAR_FAST_ZERO_INIT_UNSPECIFIED	<MSN>_START_SEC_VAR_FAST_CLEARED_UNSPECIFIED

Table 3-6 Change of memory section keywords for variables in ASR 4.0.3

Note
Although names of some keywords have changed most of AUTOSAR 4.0.3 BSW modules provided by Vector Informatik GmbH still use the old naming convention of memory section keywords. The BSW module `MemMap` at all provided by Vector Informatik GmbH accepts both naming conventions.

3.3 Compiler specific keywords

The following table gives an overview about available compiler specific keywords. These keywords are required to tell the compiler the memory location of the corresponding code and data segments. This information allows the compiler to create correct access instructions for corresponding data and code segments. The need of usage of these keywords depends on the used compiler and the memory model of used hardware.

Note
The compiler specific keywords listed below are the default ones. Apart from these ones if required each BSW module may define further own ones. Please see file `Compiler_Cfg.h` included in your delivery.

Compiler specific keyword	Memory section the compiler specific keyword used for
<MSN>_CODE	<MSN>_START_SEC_CODE
<MSN>_CODE_FAST	<MSN>_START_SEC_CODE_FAST
<MSN>_CODE_ISR	<MSN>_START_SEC_CODE_ISR
<MSN>_CONST	<MSN>_START_SEC_CONST_8BIT <MSN>_START_SEC_CONST_16BIT <MSN>_START_SEC_CONST_32BIT <MSN>_START_SEC_CONST_64BIT <MSN>_START_SEC_CONST_UNSPECIFIED
<MSN>_CONST_FAST	<MSN>_START_SEC_FAST_CONST_8BIT <MSN>_START_SEC_FAST_CONST_16BIT <MSN>_START_SEC_FAST_CONST_32BIT <MSN>_START_SEC_FAST_CONST_64BIT <MSN>_START_SEC_FAST_CONST_UNSPECIFIED
<MSN>_PBCFG	<MSN>_START_SEC_PBCFG_GLOBALROOT

	<MSN>_START_SEC_CONST_PBCFG
<MSN>_VAR_PBCFG	<MSN>_START_SEC_VAR_PBCFG
<MSN>_VAR_INIT	<MSN>_START_SEC_VAR_INIT_8BIT <MSN>_START_SEC_VAR_INIT_16BIT <MSN>_START_SEC_VAR_INIT_32BIT <MSN>_START_SEC_VAR_INIT_64BIT <MSN>_START_SEC_VAR_INIT_UNSPECIFIED
<MSN>_VAR_NOINIT	<MSN>_START_SEC_VAR_NOINIT_8BIT <MSN>_START_SEC_VAR_NOINIT_16BIT <MSN>_START_SEC_VAR_NOINIT_32BIT <MSN>_START_SEC_VAR_NOINIT_64BIT <MSN>_START_SEC_VAR_NOINIT_UNSPECIFIED
<MSN>_VAR_ZERO_INIT	<MSN>_START_SEC_VAR_ZERO_INIT_8BIT <MSN>_START_SEC_VAR_ZERO_INIT_16BIT <MSN>_START_SEC_VAR_ZERO_INIT_32BIT <MSN>_START_SEC_VAR_ZERO_INIT_64BIT <MSN>_START_SEC_VAR_ZERO_INIT_UNSPECIFIED
<MSN>_VAR_INIT_FAST	<MSN>_START_SEC_VAR_FAST_INIT_8BIT <MSN>_START_SEC_VAR_FAST_INIT_16BIT <MSN>_START_SEC_VAR_FAST_INIT_32BIT <MSN>_START_SEC_VAR_FAST_INIT_64BIT <MSN>_START_SEC_VAR_FAST_INIT_UNSPECIFIED
<MSN>_VAR_NOINIT_FAST	<MSN>_START_SEC_VAR_FAST_NOINIT_8BIT <MSN>_START_SEC_VAR_FAST_NOINIT_16BIT <MSN>_START_SEC_VAR_FAST_NOINIT_32BIT <MSN>_START_SEC_VAR_FAST_NOINIT_64BIT <MSN>_START_SEC_VAR_FAST_NOINIT_UNSPECIFIED
<MSN>_VAR_ZERO_INIT_FAST	<MSN>_START_SEC_VAR_FAST_ZERO_INIT_8BIT <MSN>_START_SEC_VAR_FAST_ZERO_INIT_16BIT <MSN>_START_SEC_VAR_FAST_ZERO_INIT_32BIT <MSN>_START_SEC_VAR_FAST_ZERO_INIT_64BIT <MSN>_START_SEC_VAR_FAST_ZERO_INIT_UNSPECIFIED

Table 3-7 Compiler specific keywords and the related memory sections they used for

4 Glossary and Abbreviations

4.1 Abbreviations

Abbreviation	Description
API	Application Programming Interface
ASR	AUTOSAR
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
ECU	Electronic Control Unit
GHS	Greenhills
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
MIP/Mip	Module Implementation Prefix
MSN/Msn	Module Short Name
RAM	Random Access Memory
ROM	Read Only Memory
SWS	Software Specification

Table 4-1 Abbreviations

5 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com