Document Title	Specification of Platform Types
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	048
Document Classification	Standard

Document Version	2.5.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
14.11.2011	2.5.0	AUTOSAR Administration	• Clarified use of operators for boolean variables • Implemented new traceability mechanism
26.10.2010	2.4.0	AUTOSAR Administration	• Detailed published parameter names (module names) in chapter 10. The previous definition was ambiguous across several releases • Changed "Module Short Name" (MSN) to "Module Abbreviation" (MAB) for the use of API service prefixes such as "CanIf"
04.12.2009	2.3.0	AUTOSAR Administration	• Restored PLATFORM012 • Clarified endian support • Clarified support for variable register width architectures • Legal disclaimer revised
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
13.11.2007	2.2.0	AUTOSAR Administration	• Chapter 8.2: "AUTOSAR supports for compiler and target implementation only 2 complement arithmetic" • Chapter 12.10: changed the basic type for *_least types (optimized types) from 'int' to 'long' for SHx processors • Removal the explicit cast to boolean in the precompile definition (#define) for macros TRUE and FALSE ("#define TRUE ((boolean) 1)" has become "#define TRUE 1") • Document meta information extended • Small layout adaptations made

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.01.2007	2.1.0	AUTOSAR Administration	• Boolean type has been defined as an eight bit long unsigned integer • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
12.07.2006	2.0.0	AUTOSAR Administration	Second release
30.06.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

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

This document specifies the AUTOSAR platform types header file. It contains all platform dependent types and symbols. Those types must be abstracted in order to become platform and compiler independent.

It is required that all platform types files are unique within the AUTOSAR community to guarantee unique types per platform and to avoid type changes when moving a software module from platform A to B.

2 Acronyms and abbreviations

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

Acronym:	***Description:***
Rollover mechanism	The following example sequence is called ‘rollover’: • An unsigned char has the value of 255 • It is incremented by 1 • The result is 0
SDU	Service Data Unit (payload)

Abbreviation:	***Description:***
int	Integer

3 Related documentation

3.1 Input documents

[1] General Requirements on Basic Software Modules,
AUTOSAR_SRS_BSWGeneral.pdf

[2] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[3] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[4] Cosmic C Cross Compiler User's Guide for Motorola MC68HC12, V4.5

[5] ARM ADS compiler manual

[6] Greenhills MULTI for V850 V4.0.5:
Building Applications for Embedded V800, V4.0, 30.1.2004

[7] 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

[8] 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

[9] TASKING for TriCore TC1796 V2.1R1:
TriCore v2.0 C Cross-Compiler, Assembler, Linker User's Guide, V1.2

[10] 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

[11] ISO/IEC 9899:1990 Programming Language – C

[12] MISRA-C 2004: Guidelines for the use of the C language in critical systems, October 2004

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

4.3 Applicability to safety related environments

The AUTOSAR `boolean` type may be used if the correct usage (see PLATFORM027) is proven by a formal code review or a static analysis by a validated static analysis tool.

The optimized AUTOSAR integer data types (`*_least`) may be used if the correct usage (see PLATFORM005) is proven by a formal code review or a static analysis by a validated static analysis tool.

5 Dependencies to other modules

None.

5.1 File structure

5.1.1 Code file structure

None

5.1.2 Header file structure

Two header file structures are applicable. One is depending on communication related basic software modules and the second is depending on non-communication related basic software modules.

5.1.2.1 Communication related basic software modules

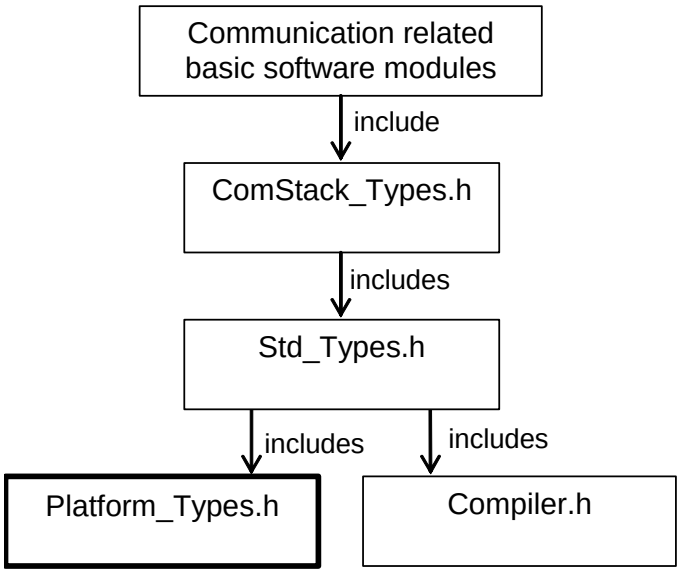


Figure 1: Include File Structure for communication related basic software modules

- If existing, <mab>_Types.h shall include ComStack_Types.h where <mab> (module abbreviation) is a communication related basic software module (e.g. Com, PduR, Can…).
- ComStack_Types.h shall include Std_Types.h
- Std_Types.h shall include Platform_Types.h
- Std_Types.h shall include Compiler.h

The existence and purpose of <mab>_Types.h is specified in the module specific SWS document.

5.1.3 Non-communication related basic software modules

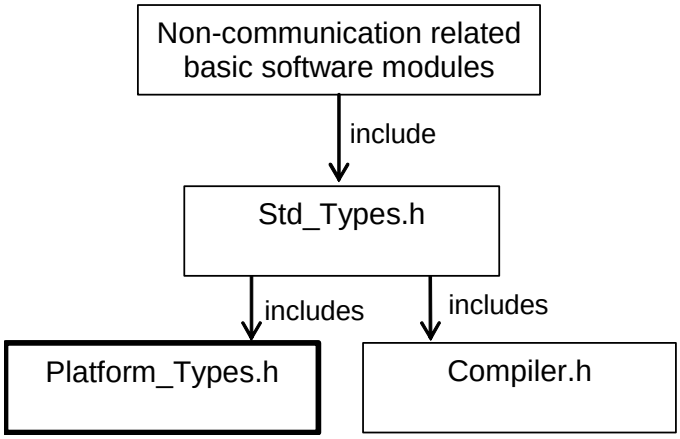


Figure 2: Include File Structure for non-communication related basic software modules

- <mab>_Types.h shall include Std_Types.h where <mab> (module abbreviation) is a non-communication related basic software module (e.g. Mcu, WdgM ...)
- Std_Types.h shall include Platform_Types.h
- Std_Types.h shall include Compiler.h

6 Requirements traceability

Requirement	Satisfied by
-	PLATFORM045
-	PLATFORM059
-	PLATFORM058
-	PLATFORM032
-	PLATFORM031
-	PLATFORM002
-	PLATFORM049
-	PLATFORM044
-	PLATFORM039
-	PLATFORM007
-	PLATFORM019
-	PLATFORM048
-	PLATFORM011
-	PLATFORM043
-	PLATFORM046
-	PLATFORM033
-	PLATFORM009
-	PLATFORM008
-	PLATFORM006
-	PLATFORM038
-	PLATFORM061
-	PLATFORM050
-	PLATFORM057
-	PLATFORM010
-	PLATFORM051
32bit	PLATFORM041
64bit	PLATFORM042
BSW00300	PLATFORM063
BSW00301	PLATFORM063
BSW00302	PLATFORM063
BSW00304	PLATFORM003, PLATFORM005, PLATFORM025, PLATFORM013, PLATFORM020, PLATFORM022, PLATFORM021, PLATFORM024, PLATFORM023, PLATFORM001, PLATFORM014, PLATFORM015, PLATFORM016, PLATFORM017, PLATFORM018
BSW00305	PLATFORM063
BSW00306	PLATFORM063
BSW00307	PLATFORM063
BSW00308	PLATFORM063
BSW00309	PLATFORM063
BSW00310	PLATFORM063

BSW00312	PLATFORM063
BSW00314	PLATFORM063
BSW00321	PLATFORM063
BSW00323	PLATFORM063
BSW00325	PLATFORM063
BSW00326	PLATFORM063
BSW00327	PLATFORM063
BSW00328	PLATFORM063
BSW00329	PLATFORM063
BSW00330	PLATFORM063
BSW00331	PLATFORM063
BSW00333	PLATFORM063
BSW00334	PLATFORM063
BSW00335	PLATFORM063
BSW00336	PLATFORM063
BSW00337	PLATFORM063
BSW00338	PLATFORM063
BSW00339	PLATFORM063
BSW00341	PLATFORM063
BSW00342	PLATFORM063
BSW00343	PLATFORM063
BSW00344	PLATFORM063
BSW00345	PLATFORM063
BSW00346	PLATFORM063
BSW00347	PLATFORM063
BSW00348	PLATFORM063
BSW00350	PLATFORM063
BSW00353	PLATFORM003, PLATFORM001
BSW00355	PLATFORM063
BSW00357	PLATFORM063
BSW00358	PLATFORM063
BSW00359	PLATFORM063
BSW00360	PLATFORM063
BSW00361	PLATFORM063
BSW00369	PLATFORM063
BSW00370	PLATFORM063
BSW00371	PLATFORM063
BSW00373	PLATFORM063
BSW00374	PLATFORM063
BSW00375	PLATFORM063
BSW00376	PLATFORM063
BSW00377	PLATFORM063

BSW00378	PLATFORM034, PLATFORM027
BSW00379	PLATFORM063
BSW00380	PLATFORM063
BSW00381	PLATFORM063
BSW00383	PLATFORM063
BSW00384	PLATFORM063
BSW00385	PLATFORM063
BSW00386	PLATFORM063
BSW00387	PLATFORM063
BSW00388	PLATFORM063
BSW00389	PLATFORM063
BSW00390	PLATFORM063
BSW00391	PLATFORM063
BSW00392	PLATFORM063
BSW00393	PLATFORM063
BSW00394	PLATFORM063
BSW00395	PLATFORM063
BSW00396	PLATFORM063
BSW00397	PLATFORM063
BSW00398	PLATFORM063
BSW00399	PLATFORM063
BSW00400	PLATFORM063
BSW00401	PLATFORM063
BSW00404	PLATFORM063
BSW00405	PLATFORM063
BSW00406	PLATFORM063
BSW00407	PLATFORM063
BSW00408	PLATFORM063
BSW00409	PLATFORM063
BSW00410	PLATFORM063
BSW00411	PLATFORM063
BSW00412	PLATFORM063
BSW00413	PLATFORM063
BSW00414	PLATFORM063
BSW00415	PLATFORM063
BSW00416	PLATFORM063
BSW00417	PLATFORM063
BSW00419	PLATFORM063
BSW00420	PLATFORM063
BSW00422	PLATFORM063
BSW00423	PLATFORM063
BSW00429	PLATFORM063

BSW00432	PLATFORM063
BSW005	PLATFORM063
BSW007	PLATFORM063
BSW009	PLATFORM063
BSW010	PLATFORM063
BSW101	PLATFORM063
BSW158	PLATFORM063
BSW159	PLATFORM063
BSW160	PLATFORM063
BSW161	PLATFORM063
BSW162	PLATFORM063
BSW164	PLATFORM063
BSW167	PLATFORM063
BSW168	PLATFORM063
BSW170	PLATFORM063
BSW171	PLATFORM063
BSW172	PLATFORM063

7 Functional specification

7.1 General issues

[PLATFORM001] ⌈For each platform an own platform types header file has to be provided. ⌋(BSW00353, BSW00304)

Here, the term “platform” refers to both the microcontroller type and, if applicable, the specific mode of the microcontroller with regard to instruction set, register size etc. For example, for a microcontroller that can run both 16-bit and 32-bit code (e.g. a x86 based CPU), two different platform types header files for each of these two instruction sets shall be created.

[PLATFORM031] ⌈If a specific compiler (not listed in this specification) requires a different mapping of ANSI C types to the AUTOSAR standard integer types, an own platform types header file for this compiler has to be provided. ⌋( )

[PLATFORM003] ⌈The file name of the platform types header file shall be for all platforms ‘`Platform_Types.h`’. ⌋(BSW00353, BSW00304)

[PLATFORM002] ⌈It is not allowed to add any extension to this file. Any extension invalidates the AUTOSAR conformity. ⌋( )

7.2 CPU Type

[PLATFORM044] ⌈For each platform the register width of the CPU used shall be indicated by defining `CPU_TYPE`. ⌋( )

[PLATFORM045] ⌈According to the register width of the CPU used, CPU_TYPE shall be assigned to one of the symbols `CPU_TYPE_8`, `CPU_TYPE_16` or `CPU_TYPE_32`. ⌋( )

7.3 Endianess

The pattern for bit, byte and word ordering in native types, such as integers, is called endianess.

[PLATFORM043] ⌈For each platform the appropriate bit order on register level shall be indicated in the platform types header file using the symbol `CPU_BIT_ORDER`. ⌋( )

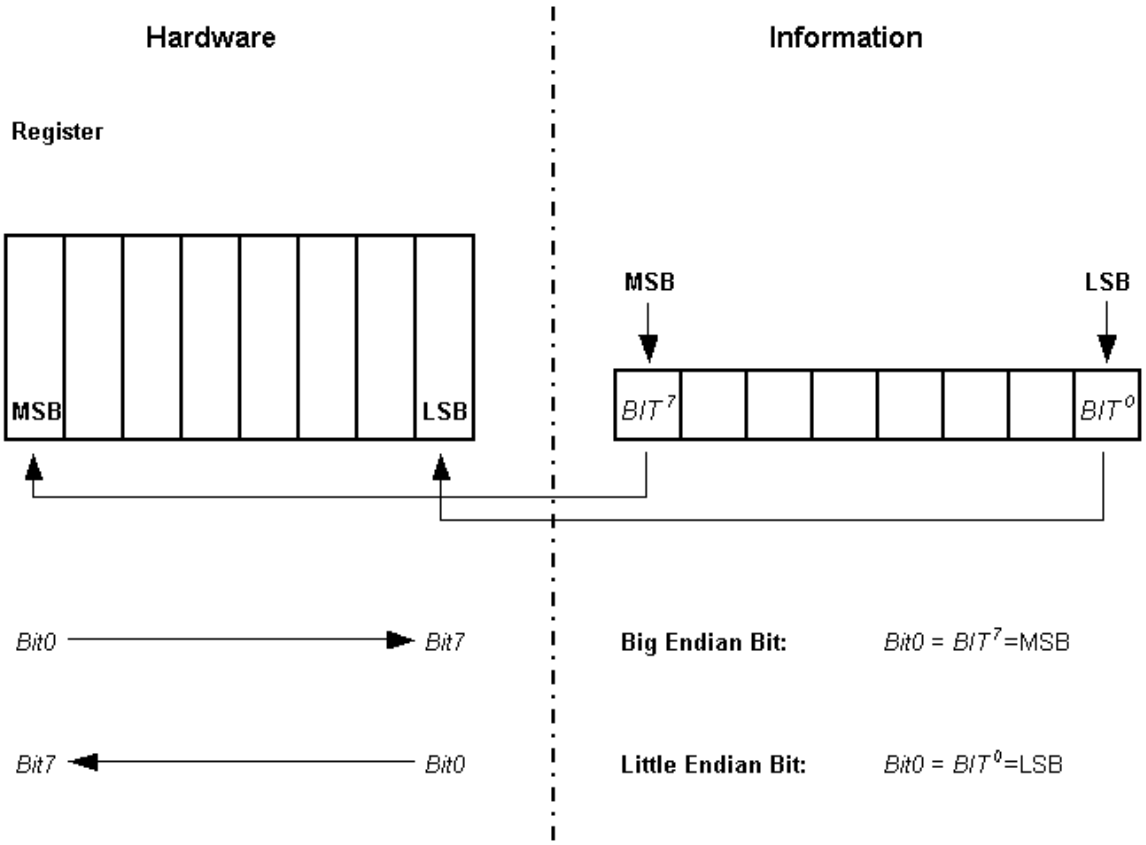
[PLATFORM046] ⌈For each platform the appropriate byte order on memory level shall be indicated in the platform types header file using the symbol `CPU_BYTE_ORDER`.⌋ ( )

7.3.1 Bit Ordering (Register)

[PLATFORM048] ⌈In case of big endian bit ordering `CPU_BIT_ORDER` shall be assigned to `MSB_FIRST` in the platform types header file.⌋ ( )

[PLATFORM049] ⌈In case of little endian bit ordering `CPU_BIT_ORDER` shall be assigned to `LSB_FIRST` in the platform types header file.⌋ ( )

Illustrations:



Important Note:
The *naming* convention Bit0, Bit1, etc. and the bit's *significance* within a byte, word, etc. are different topics and shall not be mixed. The counting scheme of bits in Motorola µC-architecture's (Big Endian Bit Order) starts with Bit0 indicating the Most Significant Bit, whereas all other µC using Little Endian Bit Order assign Bit0 to be the Least Significant Bit!

The MSB in an accumulator is always stored as the left-most bit regardless of the CPU type. Hence, big and little endianess bit orders imply different bit-naming conventions.

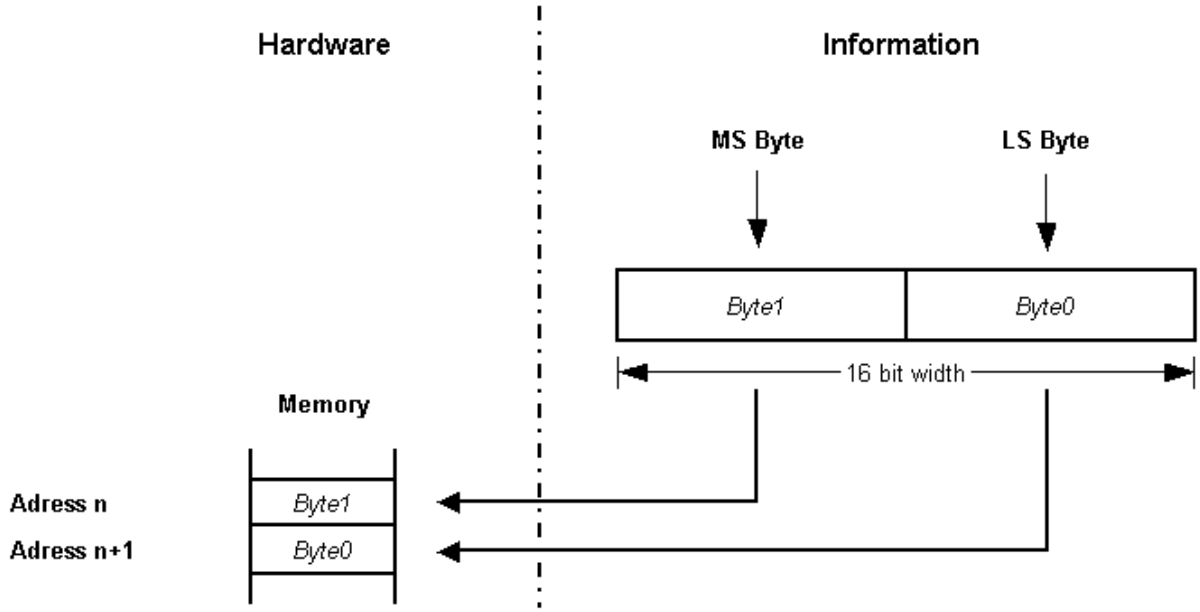
7.3.2 Byte Ordering (Memory)

[PLATFORM050] ⌈In case of big endian byte ordering `CPU_BYTE_ORDER` shall be assigned to `HIGH_BYTE_FIRST` in the platform types header file.⌋ ( )

[PLATFORM051] ⌈In case of little endian byte ordering `CPU_BYTE_ORDER` shall be assigned to `LOW_BYTE_FIRST` in the platform types header file.⌋ ( )

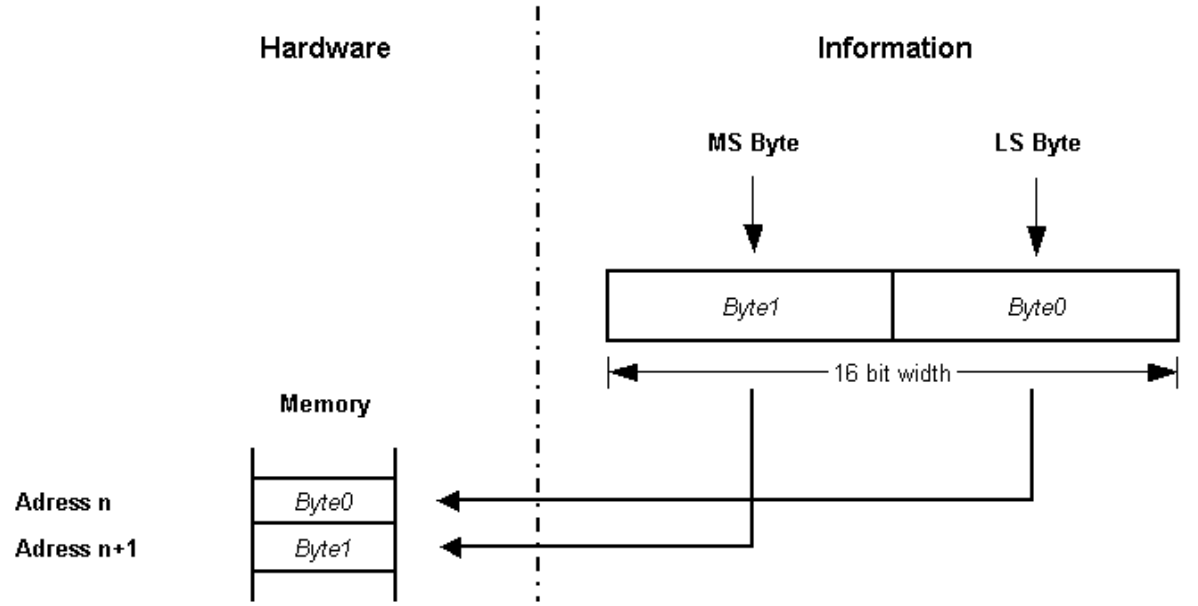
Naming convention for illustration:
The Most Significant Byte within a 16 bit wide data is named Byte1.
The Least Significant Byte within a 16 bit wide data is named Byte0.

Big Endian(HIGH_BYTE_FIRST)



Address	Data	Order
n	Byte1	Most Significant Byte (HIGH_BYTE_FIRST)
n+1	Byte0	Least Significant Byte

Little Endian(LOW_BYTE_FIRST)



Address	Data	Order
n	Byte0	Least Significant Byte (LOW_BYTE_FIRST)
n+1	Byte1	Most Significant Byte

Important Note:
The naming convention Byte0 and Byte1 is not unique and may be different in the manufacturer's reference documentation for a particular µC.

7.4 Optimized integer data types

[PLATFORM005] ⌈The optimized AUTOSAR integer data types (`<typename>_least`) shall have at least the size given by the type name, but the types shall be implemented in a way that the best performance on the specific platform is achieved. 'Best performance' is defined in this context as 'least processor cycles for variable access as possible'. Example: on a TC1796, `uint8_least` is mapped to `unsigned int` (32 bit) because access to this type requires less processor cycles than e.g. `unsigned char` (8 bit). ⌋ (BSW00304)

[PLATFORM032] ⌈The optimized AUTOSAR integer data types (`<typename>_least`) shall only be used with a local scope inside a module. They are not allowed to be used within the API of a module. ⌋ ( )

[PLATFORM033] ⌈Operations on the optimized AUTOSAR integer data types (`*_least`) shall not expect a specific size of this type. The size specified by the name is guaranteed, but can be larger. It is not allowed to use rollover mechanisms during counting and shifting. ⌋ ( )

Examples of usage:

- Loop counters (e.g. maximum loop count = 124 → use `uint8_least`)
- Switch case arguments (e.g. maximum number of states = 17 → use `uint8_least`)

7.5 boolean data type

[PLATFORM027] ⌈The standard AUTOSAR type `boolean` shall be implemented as an unsigned integer with a bit length that is the shortest one natively supported by the platform (in general 8 bits). ⌋ (BSW00378)

[PLATFORM034] ⌈The standard AUTOSAR type `boolean` shall only be used in conjunction with the standard symbols TRUE and FALSE. For value assignments of variables of type boolean no arithmetic or logical operators (+, ++, -, --, *, /, %, <<, >>, ~, &) must be used. The only allowed form of assignment is

```
boolean var = TRUE;
…
var = TRUE;
var = FALSE;
var = (a < b)   /* same for ">", "<=", ">=" */
var = (c && d)  /* same for "!", "||" */
var = (e != f)  /* same for "==" */
```

The only allowed forms of comparison are

```
boolean var = FALSE;
…
if (var == TRUE) …
if (var == FALSE) …
if (var != TRUE) …
if (var != FALSE) …
if (var) …
if (!var) …
```

⌋ (BSW00378)

8 API specification

8.1 Imported types

Not applicable.

8.2 Type definitions

Type definitions.**[PLATFORM061]**「Concerning the signed integer types, AUTOSAR supports for compiler and target implementation only 2 complement arithmetic. This directly impacts the chosen ranges for these types.」( )

8.2.1 boolean

Type:	Unsigned integer
Range:	0 FALSE
	1 TRUE
Description:	**[PLATFORM026]**「This standard AUTOSAR type shall only be used together with the definitions TRUE and FALSE. See PLATFORM027 for implementation and usage.」(BSW00378) **[PLATFORM060]**「The boolean type shall always be mapped to a platform specific type where pointers can be applied to in order to enable a passing of parameters via API. There are specific BIT types of some HW platforms which are very efficient but where no pointers can point to.」( )

8.2.2 uint8

Type:	Unsigned integer
Range:	0..255 8 bit 0x00..0xFF
Description:	**[PLATFORM013]**「This standard AUTOSAR type shall be of 8 bit unsigned.」(BSW00304)

8.2.3 uint16

Type:	Unsigned integer
Range:	0..65535 16 bit 0x0000..0xFFFF
Description:	**[PLATFORM014]**「This standard AUTOSAR type shall be of 16 bit unsigned.」(BSW00304)

8.2.4 uint32

Type:	Unsigned integer
Range:	0..4294967295 32 bit 0x00000000..0xFFFFFFFF
Description:	**[PLATFORM015]** ⌈This standard AUTOSAR type shall be 32 bit unsigned. ⌋ (BSW00304)

8.2.5 sint8

Type:	Signed integer
Range:	-128..+127 7 bit + 1 bit sign 0x80..0x7F
Description:	**[PLATFORM016]** ⌈This standard AUTOSAR type shall be 8 bit signed. ⌋ (BSW00304)

8.2.6 sint16

Type:	Signed integer
Range:	-32768 ..+32767 15 bit + 1 bit sign 0x8000..0x7FFF
Description:	**[PLATFORM017]** ⌈This standard AUTOSAR type shall be 16 bit signed. ⌋ (BSW00304)

8.2.7 sint32

Type:	Signed integer
Range:	-2147483648.. +2147483647 31 bit + 1 bit sign 0x80000000..0x7FFFFFFF
Description:	**[PLATFORM018]** ⌈ ⌋ (BSW00304)

8.2.8 uint8_least

Type:	Unsigned integer
Range:	At least 0..255 At least 8 bit
Description:	**[PLATFORM020]** ⌈This optimized AUTOSAR type shall be at least of 8 bit unsigned. See PLATFORM005 for implementation and usage. ⌋ (BSW00304)

8.2.9 uint16_least

Type:	Unsigned integer
Range:	At least 0..65535 At least 16 bit
Description:	**[PLATFORM021]** ⌈This standard AUTOSAR type shall be at least 16 bit unsigned. See PLATFORM005 for implementation and usage. ⌋ (BSW00304)

8.2.10 uint32_least

Type:	Unsigned integer
Range:	At least 0..4294967295 At least 32 bit
Description:	**[PLATFORM022]**. See PLATFORM005 for implementation and usage. ⌋ (BSW00304)

8.2.11 sint8_least

Type:	Signed integer
Range:	At least -128..+127 At least 7 bit + 1 bit sign
Description:	**[PLATFORM023]**. See PLATFORM005 for implementation and usage. ⌋ (BSW00304)

8.2.12 sint16_least

Type:	Signed integer
Range:	At least -32768 ..+32767 At least 15 bit + 1 bit sign
Description:	**[PLATFORM024]**. See PLATFORM005 for implementation and usage. ⌋ (BSW00304)

8.2.13 sint32_least

Type:	Signed integer
Range:	At least -2147483648 .. +2147483647 At least 31 bit + 1 bit sign
Description:	**[PLATFORM025]** See PLATFORM005 for implementation and usage. ⌋ (BSW00304)

8.2.14 float32

Type:	Float
Range:	- 32 bit
Description:	**[PLATFORM041]** ⌋ ( )

8.2.15 float64

Type:	Double
Range:	- 64 bit
Description:	**[PLATFORM042]**「」( )

8.3 Symbol definitions

8.3.1 CPU_TYPE

Symbol	CPU_TYPE	
Range	CPU_TYPE_8	Indicating a 8 bit processor
	CPU_TYPE_16	Indicating a 16 bit processor
	CPU_TYPE_32	Indicating a 32 bit processor
Description:	This symbol shall be defined as #define having one of the values CPU_TYPE_8, CPU_TYPE_16 or CPU_TYPE_32 according to the platform.	

8.3.2 CPU_BIT_ORDER

Symbol	CPU_BIT_ORDER	
Range	MSB_FIRST	The most significant bit is the first bit of the bit sequence.
	LSB_FIRST	The least significant bit is the first bit of the bit sequence.
Description:	**[PLATFORM038]** ⌈MSB_FIRST LSB_FIRST⌋ ( )	

8.3.3 CPU_BYTE_ORDER

Symbol	CPU_BYTE_ORDER	
Range	HIGH_BYTE_FIRST	Within a uint16, the high byte is located before the low byte.
	LOW_BYTE_FIRST	Within uint16, the low byte is located before the high byte.
Description:	**[PLATFORM039]** ⌈This symbol shall be defined as #define having one of the values HIGH_BYTE_FIRST or LOW_BYTE_FIRST according to the platform.⌋ ( )	

8.3.4 TRUE, FALSE

Symbol/Value:	TRUE	1
Symbol/Value:	FALSE	0

Description:	**[PLATFORM054]** ⌈In case of in-built compiler support of the symbols, redefinitions shall be avoided using a conditional check.⌋ ( )
	[PLATFORM056] ⌈The symbols TRUE and FALSE shall be defined as follows:
	`#ifndef TRUE`
	`#define TRUE 1`
	`#endif`
	`#ifndef FALSE`
	`#define FALSE 0`
	`#endif`⌋ ( )	

	[PLATFORM055] ⌈These symbols shall only be used in conjunction with the `boolean` type defined in Platform_Types.h. ⌋ ( )

8.4 Function definitions

Not applicable.

8.5 Call-back notifications

Not applicable.

8.6 Scheduled functions

Not applicable.

8.7 Expected Interfaces

Not applicable.

9 Sequence diagrams

Not applicable.

10 Configuration specification

10.1 Published parameters

[PLATFORM062] ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [1] 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 [3].⌋(BSW00402, BSW004, BSW003, BSW00318)

The standard common published information like

vendorId (PLATFORM_VENDOR_ID),
moduleId (PLATFORM_MODULE_ID),
arMajorVersion (PLATFORM_AR_MAJOR_VERSION),
arMinorVersion (PLATFORM_AR_MINOR_VERSION),
arPatchVersion (PLATFORM_AR_PATCH_VERSION),
swMajorVersion (PLATFORM_SW_MAJOR_VERSION),
swMinorVersion (PLATFORM_SW_MINOR_VERSION),
swPatchVersion (PLATFORM_SW_PATCH_VERSION),
vendorApiInfix (PLATFORM_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [2] Figure 4.1 and Figure 7.1).

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

11 Annex

11.1 Type definitions – general

[PLATFORM057] ⌈The platform type files for all platforms shall contain the following symbols:

```
#define CPU_TYPE_8        8
#define CPU_TYPE_16       16
#define CPU_TYPE_32       32

#define MSB_FIRST         0
#define LSB_FIRST         1

#define HIGH_BYTE_FIRST   0
#define LOW_BYTE_FIRST    1
```

⌋ ( )

11.2 Type definitions – S12X

[PLATFORM006] ⌈The platform types for Freescale S12X shall have the following mapping to the ANSI C types:

Symbols:

```
#define CPU_TYPE            CPU_TYPE_16
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      HIGH_BYTE_FIRST
```

Types:

```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef signed char         sint8_least;
typedef unsigned char       uint8_least;
typedef signed short        sint16_least;
typedef unsigned short      uint16_least;
typedef signed long         sint32_least;
typedef unsigned long       uint32_least;

typedef float               float32;
typedef double              float64;
```

⌋ ( )

11.3 Type definitions – ST10

[PLATFORM007] ⌈The platform types for ST Microelectronics ST10 shall have the following mapping to the ANSI C types:

Symbols:
```
#define CPU_TYPE            CPU_TYPE_16
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      LOW_BYTE_FIRST
```

Types:
```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef unsigned short      uint8_least;
typedef unsigned short      uint16_least;
typedef unsigned long       uint32_least;
typedef signed short        sint8_least;
typedef signed short        sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```
⌋ ( )

11.4 Type definitions – ST30

[PLATFORM008] ⌈The platform types for STMicroelectronics ST30 shall have the following mapping to the ANSI C types:

Symbols:
```
#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      LOW_BYTE_FIRST
```

Types:
```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
```

```
typedef unsigned long       uint32;

typedef unsigned long       uint8_least;
typedef unsigned long       uint16_least;
typedef unsigned long       uint32_least;
typedef signed long         sint8_least;
typedef signed long         sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```

⌋( )

11.5 Type definitions – V850

[PLATFORM009] ⌈The platform types for NEC V850 shall have the following mapping to the ANSI C types:

Symbols:

```
#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      LOW_BYTE_FIRST
```

Types:

```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef unsigned long       uint8_least;
typedef unsigned long       uint16_least;
typedef unsigned long       uint32_least;
typedef signed long         sint8_least;
typedef signed long         sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```

⌋( )

11.6 Type definitions – MPC5554

[PLATFORM010] ⌈The platform types for Freescale MPC5554 shall have the following mapping to the ANSI C types:

Symbols:
```
#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       MSB_FIRST
#define CPU_BYTE_ORDER      HIGH_BYTE_FIRST
```

Types:
```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef unsigned long       uint8_least;
typedef unsigned long       uint16_least;
typedef unsigned long       uint32_least;
typedef signed long         sint8_least;
typedef signed long         sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```
⌋( )

11.7 Type definitions – TC1796/TC1766

[PLATFORM011] ⌈The platform types for Infineon TC1796/TC1766 shall have the following mapping to the ANSI C types:

Symbols:
```
#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      LOW_BYTE_FIRST
```

Types:
```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef unsigned long       uint8_least;
typedef unsigned long       uint16_least;
typedef unsigned long       uint32_least;
typedef signed long         sint8_least;
```

```
typedef signed long         sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```

⌋( )

11.8 Type definitions – MB91F

[PLATFORM019] ⌈The platform types for Fujitsu MB91F shall have the following mapping to the ANSI C types:

Symbols:

```
#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      HIGH_BYTE_FIRST
```

Types:

```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef unsigned long       uint8_least;
typedef unsigned long       uint16_least;
typedef unsigned long       uint32_least;
typedef signed long         sint8_least;
typedef signed long         sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```

⌋( )

11.9 Type definitions – M16C/M32C

[PLATFORM058] ⌈The platform types for Renesas M16C and M32C shall have the following mapping to the ANSI C types:

Symbols:

```
#define CPU_TYPE            CPU_TYPE_16
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      LOW_BYTE_FIRST
```

Types:

```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed long         sint32;
typedef unsigned long       uint32;

typedef unsigned short      uint8_least;
typedef unsigned short      uint16_least;
typedef unsigned long       uint32_least;
typedef signed short        sint8_least;
typedef signed short        sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```

⌋( )

11.10Type definitions – SHx

[PLATFORM059] ⌈The platform types for Renesas SHx shall have the following mapping to the ANSI C types:

Symbols:

```
#define CPU_TYPE            CPU_TYPE_32
#define CPU_BIT_ORDER       LSB_FIRST
#define CPU_BYTE_ORDER      HIGH_BYTE_FIRST
```

Types:

```
typedef unsigned char       boolean;

typedef signed char         sint8;
typedef unsigned char       uint8;
typedef signed short        sint16;
typedef unsigned short      uint16;
typedef signed int          sint32;
typedef unsigned int        uint32;

typedef unsigned long       uint8_least;
typedef unsigned long       uint16_least;
typedef unsigned long       uint32_least;
typedef signed long         sint8_least;
typedef signed long         sint16_least;
typedef signed long         sint32_least;

typedef float               float32;
typedef double              float64;
```

⌋( )

12 Not applicable requirements

[PLATFORM063] ⌈These requirements are not applicable to this specification.⌋ (BSW00344, BSW00404, BSW00405, BSW00345, BSW159, BSW167, BSW171, BSW170, BSW00380, BSW00419, BSW00381, BSW00412, BSW00383, BSW00384, BSW00387, BSW00388, BSW00389, BSW00390, BSW00391, BSW00392, BSW00393, BSW00394, BSW00395, BSW00396, BSW00397, BSW00398, BSW00399, BSW00400, BSW00375, BSW101, BSW00416, BSW00406, BSW168, BSW00407, BSW00423, BSW00429, BSW00432, BSW00336, BSW00337, BSW00338, BSW00369, BSW00339, BSW00422, BSW00420, BSW00417, BSW00323, BSW00409, BSW00385, BSW00386, BSW161, BSW162, BSW005, BSW00415, BSW164, BSW00325, BSW00326, BSW00342, BSW00343, BSW160, BSW007, BSW00300, BSW00413, BSW00347, BSW00305, BSW00307, BSW00310, BSW00373, BSW00327, BSW00335, BSW00350, BSW00408, BSW00410, BSW00411, BSW00346, BSW158, BSW00314, BSW00370, BSW00348, BSW00361, BSW00301, BSW00302, BSW00328, BSW00312, BSW00357, BSW00377, BSW00355, BSW00306, BSW00308, BSW00309, BSW00371, BSW00358, BSW00414, BSW00376, BSW00359, BSW00360, BSW00329, BSW00330, BSW00331, BSW009, BSW00401, BSW172, BSW010, BSW00333, BSW00374, BSW00379, BSW00321, BSW00341, BSW00334)