

Document Title	Specification of Platform Types
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	048
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Document Change History			
Date	Release	Changed by	Change Description
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> Editorial changes.
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> Support for 64 bit MCU's added. Editorial changes.
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> Float types shall follow the appropriate binary interchange format of IEEE 754-2008. Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> removed SWS_Platform_00063 as the influence of Post-build time configuration parameters on header files is already specified in SWS_BswGeneral
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> Types uint64 and sint64 added Editorial changes Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> Editorial changes
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> Clarified use of operators for boolean variables Implemented new traceability mechanism

Document Change History			
Date	Release	Changed by	Change Description
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"> 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"
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"> Restored PLATFORM012 Clarified endian support Clarified support for variable register width architectures Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> 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
2007-01-24	2.1.15	AUTOSAR Administration	<ul style="list-style-type: none"> 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
2006-05-16	2.0	AUTOSAR Administration	<ul style="list-style-type: none"> Second release

Document Change History

Date	Release	Changed by	Change Description
2005-05-31	1.0	AUTOSAR Administration	<ul style="list-style-type: none">• Initial Release

Disclaimer

This work (specification and/or software implementation) 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 work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work 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 work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	7
2	Acronyms and abbreviations	8
3	Related documentation.....	9
3.1	Input documents.....	9
3.2	Related standards and norms	9
3.3	Related specification	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	Functional specification	15
7.1	General issues	15
7.2	CPU Type.....	15
7.3	Endianness	15
7.3.1	Bit Ordering (Register)	15
7.3.2	Byte Ordering (Memory).....	16
7.4	Optimized integer data types.....	18
7.5	Boolean data type	18
8	API specification.....	19
8.1	Imported types.....	19
8.2	Type definitions	19
8.2.1	boolean	19
8.2.2	uint8	20
8.2.3	uint16	20
8.2.4	uint32	20
8.2.5	uint64	21
8.2.6	sint8	21
8.2.7	sint16	21
8.2.8	sint32	22
8.2.9	sint64	22
8.2.10	uint8_least.....	22
8.2.11	uint16_least.....	23
8.2.12	uint32_least.....	23
8.2.13	sint8_least.....	23
8.2.14	sint16_least.....	23
8.2.15	sint32_least.....	24
8.2.16	float32	24
8.2.17	float64	24

8.3	Symbol definitions	25
8.3.1	CPU_TYPE	25
8.3.2	CPU_BIT_ORDER	25
8.3.3	CPU_BYTE_ORDER	25
8.3.4	TRUE, FALSE	26
8.4	Function definitions	27
8.5	Call-back notifications	27
8.6	Scheduled functions	27
8.7	Expected Interfaces.....	27
9	Sequence diagrams	28
10	Configuration specification	29
10.1	Published parameters	29
11	Annex.....	30
11.1	Type definitions – general	30
11.2	Type definitions – S12X	30
11.3	Type definitions – ST10.....	31
11.4	Type definitions – ST30.....	31
11.5	Type definitions – V850.....	32
11.6	Type definitions – MPC5554	33
11.7	Type definitions – TC1796/TC1766.....	33
11.8	Type definitions – MB91F.....	34
11.9	Type definitions – M16C/M32C	35
11.10	Type definitions – SHx.....	35
11.11	Type definitions - ARM Cortex A53.....	36
12	Not applicable requirements	37

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': <ul style="list-style-type: none">• 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
- [11] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

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

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [11] (SWS BSW General), which is also valid for Platform Types.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Platform Types.

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 [SWS Platform_00027](#)) 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 chapter 7.4) 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

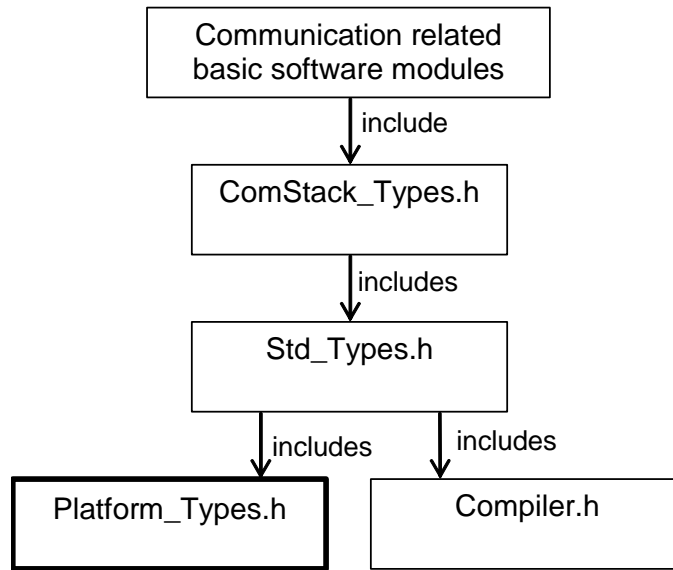


Figure 1: Include File Structure for communication related basic software modules

- 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...).

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

5.1.2.2 Non-communication related basic software modules

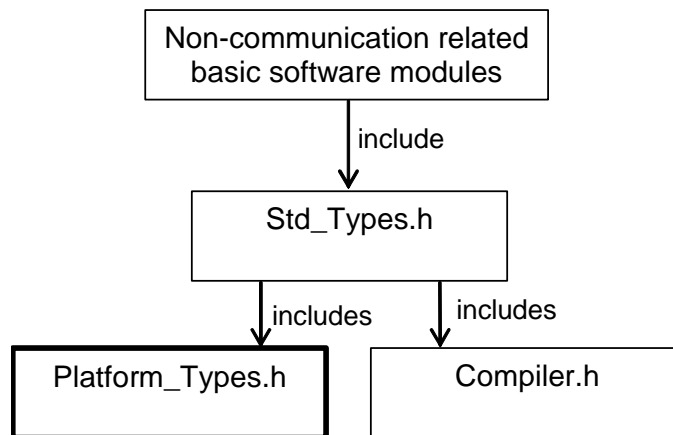


Figure 2: Include File Structure for non-communication related basic software modules

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

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00304	All AUTOSAR Basic Software Modules shall use the following data types instead of native C data types	SWS_Platform_00013, SWS_Platform_00014, SWS_Platform_00015, SWS_Platform_00016, SWS_Platform_00017, SWS_Platform_00018, SWS_Platform_00020, SWS_Platform_00021, SWS_Platform_00022, SWS_Platform_00023, SWS_Platform_00024, SWS_Platform_00025
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_Platform_00026, SWS_Platform_00027, SWS_Platform_00034

7 Functional specification

7.1 General issues

[SWS_Platform_00002] [It is not allowed to add any extension to this file. Any extension invalidates the AUTOSAR conformity.] ()

7.2 CPU Type

[SWS_Platform_00044] [For each platform the register width of the CPU used shall be indicated by defining `CPU_TYPE`.] ()

[SWS_Platform_00045] [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`, `CPU_TYPE_32` or `CPU_TYPE_64`.] ()

7.3 Endianness

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

[SWS_Platform_00043] [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`.] ()

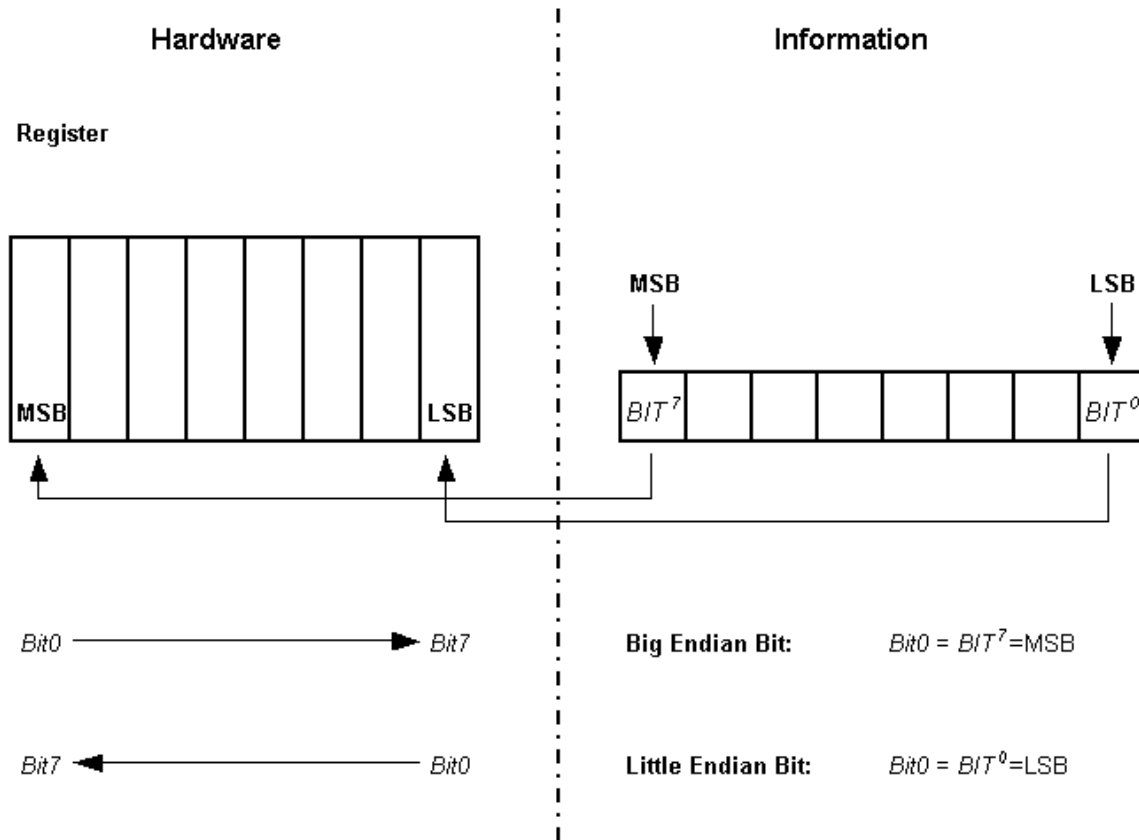
[SWS_Platform_00046] [For each platform the appropriate byte order on memory level shall be indicated in the platform types header file using the symbol `CPU_BYTE_ORDER`.] ()

7.3.1 Bit Ordering (Register)

[SWS_Platform_00048] [In case of big endian bit ordering `CPU_BIT_ORDER` shall be assigned to `MSB_FIRST` in the platform types header file.] ()

[SWS_Platform_00049] [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 μ C-architecture's (Big Endian Bit Order) starts with Bit0 indicating the Most Significant Bit, whereas all other μ C 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 endianness bit orders imply different bit-naming conventions.

7.3.2 Byte Ordering (Memory)

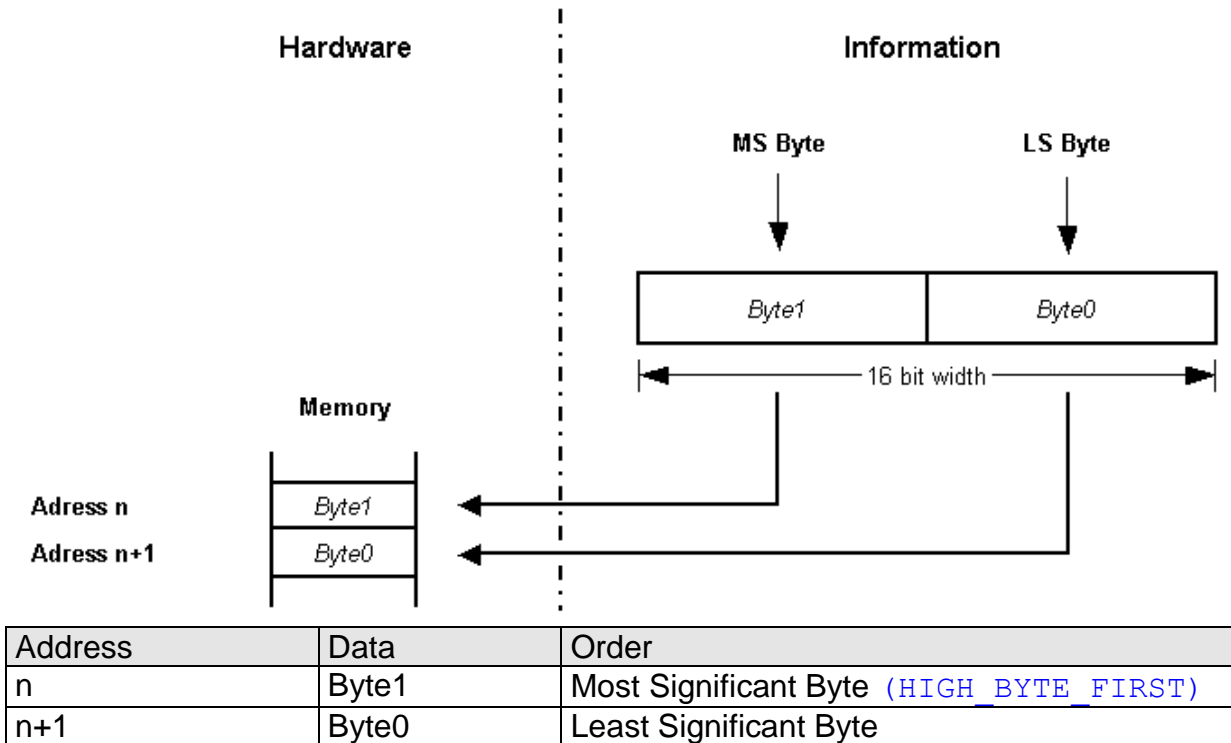
[SWS_Platform_00050] [In case of big endian byte ordering `CPU_BYTE_ORDER` shall be assigned to `HIGH_BYTE_FIRST` in the platform types header file.] ()

[SWS_Platform_00051] [In case of little endian byte ordering `CPU_BYTE_ORDER` shall be assigned to `LOW_BYTE_FIRST` in the platform types header file.] ()

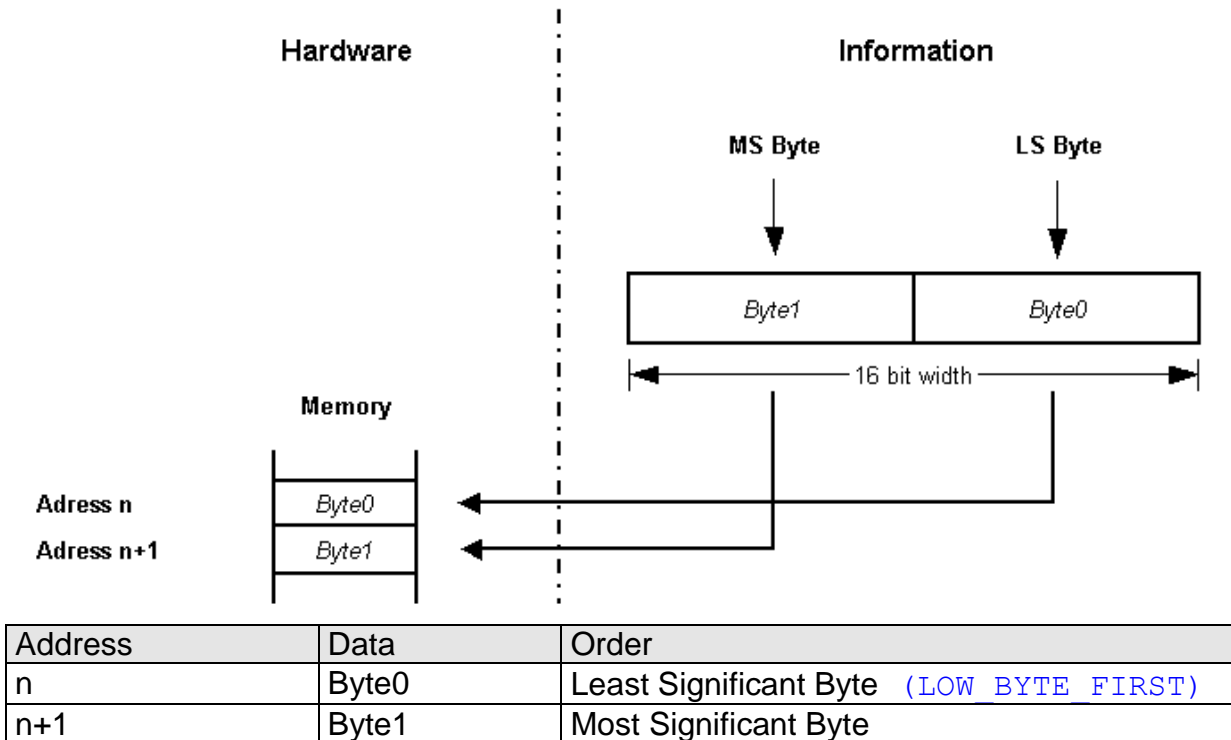
Naming convention for illustration:

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)



Little Endian (LOW_BYTE_FIRST)



Important Note:

The naming convention Byte0 and Byte1 is not unique and may be different in the manufacturer's reference documentation for a particular μ C.

7.4 Optimized integer data types

For details refer to the Chapter 7.1.19.2.1 “AUTOSAR Integer Data Types” in SWS_BSWGeneral

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

[SWS_Platform_00027] [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).] (SRS_BSW_00378)

[SWS_Platform_00034] [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) ...
```

] (SRS_BSW_00378)

8 API specification

8.1 Imported types

Not applicable.

8.2 Type definitions

[SWS_Platform_00061] [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

[SWS_Platform_00026] [

Name	boolean		
Kind	Type		
Derived from	uint		
Description	This standard AUTOSAR type shall only be used together with the definitions TRUE and FALSE.		
Range	FALSE	0	--
	TRUE	1	--
Variation	--		

] (SRS_BSW_00378)

See [SWS_Platform_00027](#) for implementation and usage.

[SWS_Platform_00060] [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

[SWS_Platform_00013] [

Name	uint8		
Kind	Type		
Derived from	uint		
Description	This standard AUTOSAR type shall be of 8 bit unsigned.		
Range	8 bit		0..255 0x00..0xFF
Variation	--		

] (SRS_BSW_00304)

8.2.3 uint16

[SWS_Platform_00014] [

Name	uint16		
Kind	Type		
Derived from	uint		
Description	This standard AUTOSAR type shall be of 16 bit unsigned.		
Range	16 bit		0..65535 0x0000..0xFFFF
Variation	--		

] (SRS_BSW_00304)

8.2.4 uint32

[SWS_Platform_00015] [

Name	uint32		
Kind	Type		
Derived from	uint		
Description	This standard AUTOSAR type shall be 32 bit unsigned.		
Range	32 bit		0..4294967295 0x00000000..0xFFFFFFFF
Variation	--		

] (SRS_BSW_00304)

8.2.5 uint64

[SWS_Platform_00066] [

Name	uint64	
Kind	Type	
Derived from	uint	
Description	This standard AUTOSAR type shall be 64 bit unsigned.	
Range	64 bit	0..18446744073709551615 0x0000000000000000..0xFFFFFFFFFFFFFFFF
Variation	--	

] ()

8.2.6 sint8

[SWS_Platform_00016] [

Name	sint8	
Kind	Type	
Derived from	sint	
Description	This standard AUTOSAR type shall be of 8 bit signed.	
Range	7 bit + 1 bit sign	-128..+127 0x80..0x7F
Variation	--	

] (SRS_BSW_00304)

8.2.7 sint16

[SWS_Platform_00017] [

Name	sint16	
Kind	Type	
Derived from	sint	
Description	This standard AUTOSAR type shall be of 16 bit signed.	
Range	15 bit + 1 bit sign	-32768..+32767 0x8000..0x7FFF
Variation	--	

] (SRS_BSW_00304)

8.2.8 sint32

[SWS_Platform_00018] [

Name	sint32		
Kind	Type		
Derived from	sint		
Description	This standard AUTOSAR type shall be 32 bit signed.		
Range	31 bit + 1 bit sign	--	-2147483648..+2147483647 0x80000000..0x7FFFFFFF
Variation	--		

] (SRS_BSW_00304)

8.2.9 sint64

[SWS_Platform_00067] [

Name:	sint64		
Type:	sint		
Range:	63 bit + 1 bit sign	--	-9223372036854775808..9223372036854775807 0x8000000000000000..0x7FFFFFFFFFFFFFFF
Description:	This standard AUTOSAR type shall be 64 bit signed.		

] ()

8.2.10 uint8_least

[SWS_Platform_00020] [

Name:	uint8_least		
Type:	uint		
Range:	At least 8 bit	--	At least 0..255
Description:	This optimized AUTOSAR type shall be at least 8 bit unsigned.		

] (SRS_BSW_00304)

See chapter 7.4 for implementation and usage.

8.2.11 uint16_least

[SWS_Platform_00021] [

Name:	uint16_least		
Type:	uint		
Range:	At least 16 bit	--	At least 0..65535 0x0000..0xFFFF
Description:	This optimized AUTOSAR type shall be at least 16 bit unsigned.		

] (SRS_BSW_00304)

See chapter 7.4 for implementation and usage.

8.2.12 uint32_least

[SWS_Platform_00022] [

Name:	uint32_least		
Type:	uint		
Range:	At least 32 bit	--	At least 0..4294967295 0x00000000..0xFFFFFFFF
Description:	This optimized AUTOSAR type shall be at least 32 bit unsigned.		

] (SRS_BSW_00304)

See chapter 7.4 for implementation and usage.

8.2.13 sint8_least

[SWS_Platform_00023] [

Name:	sint8_least		
Type:	sint		
Range:	At least 7 bit + 1 bit sign	--	At least -128..+127 0x80..0x7F
Description:	This optimized AUTOSAR type shall be at least 8 bit signed.		

] (SRS_BSW_00304)

See chapter 7.4 for implementation and usage.

8.2.14 sint16_least

[SWS_Platform_00024] [

Name:	sint16_least		
Type:	sint		
Range:	At least 15 bit + 1 bit sign	--	At least -32768..+32767 0x8000..0x7FFF
Description:	This optimized AUTOSAR type shall be at least 16 bit signed.		

] (SRS_BSW_00304)

8.2.15 sint32_least

[SWS_Platform_00025] [

Name:	sint32_least		
Type:	sint		
Range:	At least 31 bit + 1 bit sign	--	At least -2147483648.. +2147483647 0x80000000..0x7FFFFFFF
Description:	This optimized AUTOSAR type shall be at least 32 bit signed.		

] (SRS_BSW_00304)

See chapter 7.4 for implementation and usage.

8.2.16 float32

[SWS_Platform_00041] [

Name	float32		
Kind	Type		
Derived from	float		
Description	This standard AUTOSAR type shall follow the 32-bit binary interchange format according to IEEE 754-2008 with encoding parameters specified in chapter 3.6, table 3.5, column "binary32".		
Range	32 bit		-3.4028235e+38.. +3.4028235e+38
Variation	--		

] ()

8.2.17 float64

[SWS_Platform_00042] [

Name:	float64		
Type:	double		
Range:	64 bit	--	1.7976931348623157e+308.. +1.7976931348623157e+308
Description:	This standard AUTOSAR type shall follow the 64-bit binary interchange format according to IEEE 754-2008 with encoding parameters specified in chapter 3.6, table 3.5, column "binary64".		

] ()

8.3 Symbol definitions

8.3.1 CPU_TYPE

[SWS_Platform_00064] [

Name:	CPU_TYPE		
Type:	Enumeration		
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
	CPU_TYPE_64	--	Indicating a 64 bit processor
Description:	This symbol shall be defined as #define having one of the values CPU_TYPE_8, CPU_TYPE_16, CPU_TYPE_32 or CPU_TYPE_64 according to the platform.		

] ()

8.3.2 CPU_BIT_ORDER

[SWS_Platform_00038] [

Name:	CPU_BIT_ORDER		
Type:	Enumeration		
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:	This symbol shall be defined as #define having one of the values MSB_FIRST or LSB_FIRST according to the platform.		

] ()

8.3.3 CPU_BYTE_ORDER

[SWS_Platform_00039] [

Name:	CPU_BYTE_ORDER		
Type:	Enumeration		
Range:	HIGH_BYTE_FIRST	--	Within 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:	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

[SWS_Platform_00056] [

Name:	TRUE FALSE		
Type:	Enumeration		
Range:	FALSE	0x00	--
	TRUE	0x01	--
Description:	<p>The symbols TRUE and FALSE shall be defined as follows:</p> <pre> #ifndef TRUE #define TRUE 1 #endif #ifndef FALSE #define FALSE 0 #endif </pre>		

] ()

[SWS_Platform_00054] [In case of in-built compiler support of the symbols, redefinitions shall be avoided using a conditional check.] ()

[SWS_Platform_00055] [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

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral*

11 Annex

11.1 Type definitions – general

The platform type files for all platforms could contain the following symbols:

```
#define CPU_TYPE_8      8
#define CPU_TYPE_16    16
#define CPU_TYPE_32    32
#define CPU_TYPE_64    64

#define MSB_FIRST      0
#define LSB_FIRST      1

#define HIGH_BYTE_FIRST 0
#define LOW_BYTE_FIRST 1
```

11.2 Type definitions – S12X

The platform types for Freescale S12X could 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 signed long long sint64;
typedef unsigned long    uint32;
typedef unsigned long long uint64;

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

The platform types for ST Microelectronics ST10 could 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 signed long long   sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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

The platform types for STMicroelectronics ST30 could 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 signed long long    sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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

The platform types for NEC V850 could 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 signed long long   sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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

The platform types for Freescale MPC5554 could 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 signed long long   sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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

The platform types for Infineon TC1796/TC1766 could 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 signed long long    sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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

The platform types for Fujitsu MB91F could 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 signed long long   sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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

The platform types for Renesas M16C and M32C could 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 signed long long   sint64;
typedef unsigned long      uint32;
typedef unsigned long long uint64;

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;
```

11.10 Type definitions – SHx

The platform types for Renesas SHx could 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 signed long long    sint64;
typedef unsigned int       uint32;
typedef unsigned long long uint64;

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;
```

11.11 Type definitions - ARM Cortex A53

The platform types for ARM Cortex A53 in Little Endian could have the following mapping to the ANSI C types:

Symbols:

```
#define CPU_TYPE           CPU_TYPE_64
#define CPU_BIT_ORDER     LSB_FIRST
#define CPU_BYTE_ORDER    LOW_BYTE_FIRST
```

Types:

```
typedef unsigned char      boolean;

typedef unsigned char      uint8;
typedef unsigned short     uint16;
typedef unsigned int       uint32;
typedef unsigned long long uint64;

typedef signed char        sint8;
typedef signed short       sint16;
typedef signed int         sint32;
typedef signed long long   sint64;

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

typedef float              float32;
typedef double             float64;
```

12 Not applicable requirements

[SWS_Platform_00063] [These requirements are not applicable to this specification.

] (SRS_BSW_00344, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00345, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00171, SRS_BSW_00170, SRS_BSW_00419, SRS_BSW_00381, SRS_BSW_00412, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00387, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00391, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00397, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00375, SRS_BSW_00101, SRS_BSW_00416, SRS_BSW_00406, SRS_BSW_00168, SRS_BSW_00407, SRS_BSW_00423, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00336, SRS_BSW_00337, SRS_BSW_00338, SRS_BSW_00369, SRS_BSW_00339, SRS_BSW_00422, SRS_BSW_00420, SRS_BSW_00417, SRS_BSW_00323, SRS_BSW_00409, SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00342, SRS_BSW_00343, SRS_BSW_00160, SRS_BSW_00007, SRS_BSW_00300, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00305, SRS_BSW_00307, SRS_BSW_00310, SRS_BSW_00373, SRS_BSW_00327, SRS_BSW_00335, SRS_BSW_00350, SRS_BSW_00408, SRS_BSW_00410, SRS_BSW_00411, SRS_BSW_00346, SRS_BSW_00158, SRS_BSW_00314, SRS_BSW_00370, SRS_BSW_00348, SRS_BSW_00361, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00328, SRS_BSW_00312, SRS_BSW_00357, SRS_BSW_00377, SRS_BSW_00355, SRS_BSW_00306, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00371, SRS_BSW_00358, SRS_BSW_00414, SRS_BSW_00376, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00329, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00009, SRS_BSW_00401, SRS_BSW_00172, SRS_BSW_00010, SRS_BSW_00333, SRS_BSW_00374, SRS_BSW_00379, SRS_BSW_00321, SRS_BSW_00341, SRS_BSW_00334] ()