| Document Title | Specification of Platform Types |
|---|---|
| Document Owner | AUTOSAR GbR |
| Document Responsibility | AUTOSAR GbR |
| Document Version | 2.0.0 |
| Document Status | Final |

| Document Change History | | | |
|---|---|---|---|
| Date | Version | Changed by | Change Description |
| 12.07.2006 | 2.0.0 | AUTOSAR Administration | Second release |
| 30.06.2005 | 1.0.0 | AUTOSAR Administration | Initial Release |

**Disclaimer**

**Advice to users of AUTOSAR Specification Documents:**

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 AUTOSAR compliance certification 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

- AUTOSAR confidential -

# 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': <br> • An unsigned char has the value of 255 <br> • It is incremented by 1 <br> • The result is 0 |
| SDU | Service Data Unit (payload) |

| Abbreviation: | Description: |
|---|---|
| int | Integer |

- AUTOSAR confidential -

# 3 Related documentation

## 3.1 Input documents

General Requirements on Basic Software Modules,
AUTOSAR_SRS_General.pdf

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

[2] ARM ADS compiler manual

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

[4] 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

[5] 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

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

[7] 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

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

[9] 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

```
┌─────────────────────────┐
│ Communication related   │
│ basic software modules  │
└─────────────────────────┘
            │ include
            ▼
┌─────────────────────────┐
│ ComStack_Types.h        │
└─────────────────────────┘
            │ includes
            ▼
┌─────────────────────────┐
│ Std_Types.h             │
└─────────────────────────┘
      │ includes        │ includes
      ▼                 ▼
┌──────────────┐  ┌──────────────┐
│Platform_Types.h│ │ Compiler.h   │
└──────────────┘  └──────────────┘
```

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

- `<user>_Types.h` shall include `ComStack_Types.h` and `<user>` 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`

### 5.1.3 Non-communication related basic software modules

```
┌─────────────────────────┐
│ Non-communication related│
│ basic software modules  │
└─────────────────────────┘
            │ include
            ▼
┌─────────────────────────┐
│ Std_Types.h             │
└─────────────────────────┘
      │ includes        │ includes
      ▼                 ▼
┌──────────────┐  ┌──────────────┐
│Platform_Types.h│ │ Compiler.h   │
└──────────────┘  └──────────────┘
```
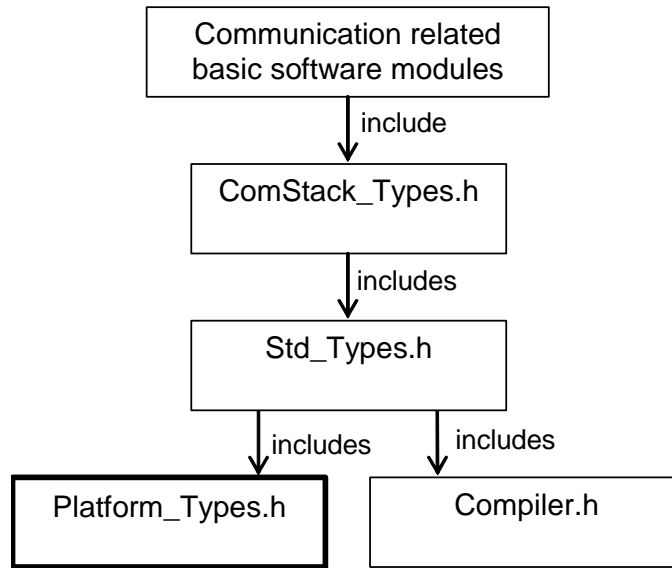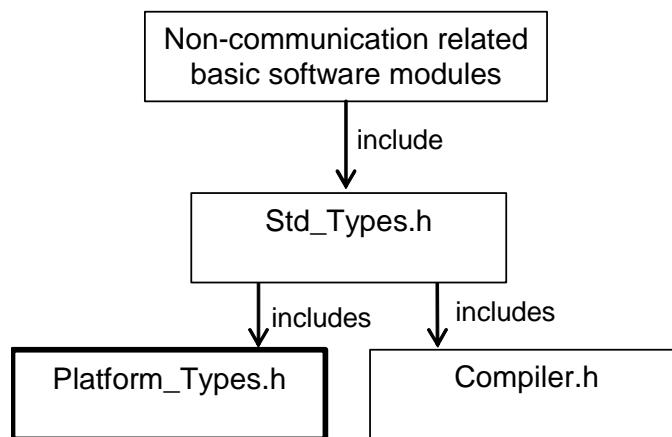
**Figure 2: Include File Structure for non-communication related basic software modules**

- `<user>_Types.h` shall include `ComStack_Types.h` and user 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

Document: General Requirements on Basic Software Modules,

| Requirement | Satisfied by |
| --- | --- |
| [BSW00344] Reference to link-time configuration | Not applicable<br>(i.e. PlatformTypes SWS specifies a header file) |
| [BSW00404] Reference to post build time configuration | Not applicable<br>(i.e. PlatformTypes SWS specifies a header file) |
| [BSW00405] Reference to multiple configuration sets | Not applicable<br>(i.e. PlatformTypes SWS specifies a header file) |
| [BSW00345] Pre-compile-time configuration | Not applicable<br>(i.e. PlatformTypes SWS is not a module specific configuration file) |
| [BSW159] Tool-based configuration | Not applicable<br>(i.e. PlatformTypes SWS is not a BSW module) |
| [BSW167] Static configuration checking | Not applicable<br>(i.e. PlatformTypes SWS is not a BSW module) |
| [BSW171] Configurability of optional functionality | Not applicable<br>(i.e. PlatformTypes SWS specifies a header file) |
| [BSW170] Data for reconfiguration of AUTOSAR SW-Components | Not applicable<br>(i.e. no reconfiguration available for platform types) |
| [BSW00380] Separate C-Files for configuration parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00419] Separate C-Files for pre-compile time configuration parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00381] Separate configuration header file for pre-compile time parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00412] Separate H-File for configuration parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00383] List dependencies of configuration files | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00384] List dependencies to other modules | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00387] Specify the configuration class of callback function | Not applicable<br>(i.e no callback function available for platform types) |
| [BSW00388] Introduce containers | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00389] Containers shall have names | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00390] Parameter content shall be unique within the module | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00391] Parameter shall have unique names | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00392] Parameters shall have a type | Not applicable<br>(i.e no configuration parameters available for platform types) |

| [BSW00393] Parameters shall have a range | Not applicable<br>(i.e no configuration parameters available for platform types) |
|---|---|
| [BSW00394] Specify the scope of the parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00395] List the required parameters (per parameter) | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00396] Configuration classes | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00397] Pre-compile-time parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00398] Link-time parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00399] Loadable Post-build time parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00400] Selectable Post-build time parameters | Not applicable<br>(i.e no configuration parameters available for platform types) |
| [BSW00402] Published information | [PLATFORM012] |
| [BSW00375] Notification of wake-up reason | Not applicable<br>(i.e. no functionality defined with platform types) |
| [BSW101] Initialization interface | Not applicable<br>(i.e. no functionality defined with platform types) |
| [BSW00416] Sequence of Initialization | Not applicable<br>(i.e. no functionality defined with platform types) |
| [BSW00406] Check module initialization | Not applicable<br>(i.e. no functionality defined with platform types) |
| [BSW168] Diagnostic Interface of SW components | Not applicable<br>(i.e. no testing of platform types defined) |
| [BSW00407] Function to read out published parameters | Not applicable<br>(i.e. no functionality defined with platform types) |
| [BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces | Not applicable<br>(i.e platform types is not a module) |
| [BSW00429] Restricted BSW OS functionality access | Not applicable<br>(i.e. no functionality defined with platform types. It's a header file) |
| [BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path | Not applicable<br>(i.e. no functionality defined with platform types. It's a header file) |
| [BSW00336] Shutdown interface | Not applicable<br>(i.e. no functionality defined in platform types. It's a header file) |
| [BSW00337] Classification of errors | Not applicable<br>(i.e. no error classification defined with platform types) |
| [BSW00338] Detection and Reporting of development errors | Not applicable<br>(i.e. no error classification defined with platform types) |
| [BSW00369] Do not return development error codes via API | Not applicable<br>(i.e. no functionality defined in platform types. It's a header file) |
| [BSW00339] Reporting of production relevant error status | Not applicable<br>(i.e. no functionality defined in platform types. It's a header file) |

| | |
|---|---|
| [BSW00422] Debouncing of production relevant error status | Not applicable (i.e. no functionality defined in platform types. It's a header file) |
| [BSW00420] Production relevant error event rate detection | Not applicable (i.e. no functionality defined in platform types. It's a header file) |
| [BSW00417] Reporting of Error Events by Non-Basic Software | Not applicable (i.e. no functionality defined in platform types. It's a header file) |
| [BSW00323] API parameter checking | Not applicable (i.e. no functionality defined in platform types. It's a header file) |
| [BSW004] Version check | [PLATFORM012] |
| [BSW00409] Header files for production code error IDs | Not applicable (i.e. no error defined with platform types) |
| [BSW00385] List possible error notifications | Not applicable (i.e. no error defined with platform types) |
| [BSW00386] Configuration for detecting an error | Not applicable (i.e. no error defined with platform types) |
| [BSW161] Microcontroller abstraction | Not applicable (i.e. no interface provided) |
| [BSW162] ECU layout abstraction | Not applicable (i.e. no interface provided) |
| [BSW005] No hard coded horizontal interfaces within MCAL | Not applicable (i.e. no interface provided) |
| [BSW00415] User dependent include files | Not applicable (i.e. no interface provided) |
| [BSW164] Implementation of interrupt service routines | Not applicable (i.e. only types are defined here) |
| [BSW00325] Runtime of interrupt service routines | Not applicable (i.e. only types are defined here) |
| [BSW00326] Transition from ISRs to OS tasks | Not applicable (i.e. only types are defined here) |
| [BSW00342] Usage of source code and object code | Not applicable (i.e. only types are defined here) |
| [BSW00343] Specification and configuration of time | Not applicable (i.e. no time configuration provided) |
| [BSW160] Human-readable configuration data | Not applicable (i.e. only types are defined here) |
| [BSW007] HIS MISRA C | Not applicable (i.e. only types are defined here) |
| [BSW00300] Module naming convention | Not applicable (i.e. only types are defined here) |
| [BSW00413] Accessing instances of BSW modules | Not applicable (i.e. only types are defined here) |
| [BSW00347] Naming separation of different instances of BSW drivers | Not applicable (i.e. instantiation of platform types required) |
| [BSW00305] Self-defined data types naming convention | Not applicable (i.e. platform types apply to all BSW modules) |
| [BSW00307] Global variables naming convention | Not applicable (i.e. only types are defined here) |
| [BSW00310] API naming convention | Not applicable (i.e. only types are defined here) |
| [BSW00373] Main processing function naming convention | Not applicable (i.e. only types are defined here) |
| [BSW00327] Error values naming convention | Not applicable (i.e. only types are defined here) |
| [BSW00335] Status values naming convention | Not applicable (i.e. only types are defined here) |

| [BSW00350] Development error detection keyword | Not applicable (i.e. only types are defined here) |
|---|---|
| [BSW00408] Configuration parameter naming convention | Not applicable (i.e. only types are defined here) |
| [BSW00410] Compiler switches shall have defined values | Not applicable (i.e. compiler switches not provided) |
| [BSW00411] Get version info keyword | Not applicable (i.e. only types are defined here) |
| [BSW00346] Basic set of module files | Not applicable (i.e. only types are defined here) |
| [BSW158] Separation of configuration from implementation | Not applicable (i.e. no configuration provided with platform types) |
| [BSW00314] Separation of interrupt frames and service routines | Not applicable (i.e. only types are defined here) |
| [BSW00370] Separation of callback interface from API | Not applicable (i.e. only types are defined here) |
| [BSW00348] Standard type header | Not applicable (i.e. platform types are defined here) |
| [BSW00353] Platform specific type header | PLATFORM001, PLATFORM003 Chapter 8.2 Type definitions |
| [BSW00361] Compiler specific language extension header | Not applicable (i.e. only types are defined here) |
| [BSW00301] Limit imported information | Not applicable (i.e. only types are defined here) |
| [BSW00302] Limit exported information | Not applicable (i.e. only types are defined here) |
| [BSW00328] Avoid duplication of code | Not applicable (i.e. only types are defined here) |
| [BSW00312] Shared code shall be reentrant | Not applicable (i.e. only types are defined here) |
| [BSW006] Platform independency | All SWS items present in this document |
| [BSW00357] Standard API return type | Not applicable (i.e. only types are defined here) |
| [BSW00377] Module specific API return types | Not applicable ( i.e. only types are defined here) |
| [BSW00304] AUTOSAR integer data types | PLATFORM001, PLATFORM003, PLATFORM005, PLATFORM013, PLATFORM014, PLATFORM015, PLATFORM016, PLATFORM017, PLATFORM018, PLATFORM020, PLATFORM021, PLATFORM022, PLATFORM023, PLATFORM024, PLATFORM025 |
| [BSW00355] Do not redefine AUTOSAR integer data types | Not applicable (i.e. only types are defined here) |
| [BSW00378] AUTOSAR boolean type | PLATFORM026, PLATFORM027, PLATFORM034 |
| [BSW00306] Avoid direct use of compiler and platform specific keywords | Not applicable (i.e. this SWS does not specify a module) |
| [BSW00308] Definition of global data | Not applicable (i.e. only types are defined here) |
| [BSW00309] Global data with read-only constraint | Not applicable (i.e. only types are defined here) |
| [BSW00371] Do not pass function pointers via API | Not applicable (i.e. only types are defined here) |
| [BSW00358] Return type of init() functions | Not applicable (i.e. only types are defined here) |
| [BSW00414] Parameter of init function | Not applicable (i.e. only types are defined here) |
| [BSW00376] Return type and parameters of main processing functions | Not applicable (i.e. only types are defined here) |
| [BSW00359] Return type of callback functions | Not applicable (i.e. only types are defined here) |

| [BSW00360] Parameters of callback functions | Not applicable (i.e. only types are defined here) |
|---|---|
| [BSW00329] Avoidance of generic interfaces | Not applicable (i.e. only types are defined here) |
| [BSW00330] Usage of macros / inline functions instead of functions | Not applicable (i.e. only types are defined here) |
| [BSW00331] Separation of error and status values | Not applicable (i.e. only types are defined here) |
| [BSW009] Module User Documentation | Not applicable (i.e. only types are defined here) |
| [BSW00401] Documentation of multiple instances of configuration parameters | Not applicable (i.e. only types are defined here) |
| [BSW172] Compatibility and documentation of scheduling strategy | Not applicable (i.e. only types are defined here) |
| [BSW010] Memory resource documentation | Not applicable (i.e. only types are defined here) |
| [BSW00333] Documentation of callback function context | Not applicable (i.e. only types are defined here) |
| [BSW00374] Module vendor identification | Not applicable (i.e. only types are defined here) |
| [BSW00379] Module identification | Not applicable (i.e. only types are defined here) |
| [BSW003] Version identification | PLATFORM012 |
| [BSW00318] Format of module version numbers | PLATFORM012 |
| [BSW00321] Enumeration of module version numbers | Not applicable (i.e. this SWS does not specify a module) |
| [BSW00341] Microcontroller compatibility documentation | Not applicable (i.e. this SWS is not a module documentation) |
| [BSW00334] Provision of XML file | Not applicable (i.e. only types are defined here) |

## 6.1 Linkage items for requirements management

This chapter contents several items which are only used in the requirement management tool. The items are necessary to build up the linkage between requirements, specification, etc. (e.g. for impact and coverage analyses).

Not applicable

For release versions, this chapter has to be removed.

# 7 Functional specification

## 7.1 General issues

**PLATFORM001**: For each platform an own platform types header file has to be provided.

**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'.

**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.

- AUTOSAR confidential -

Illustrations:
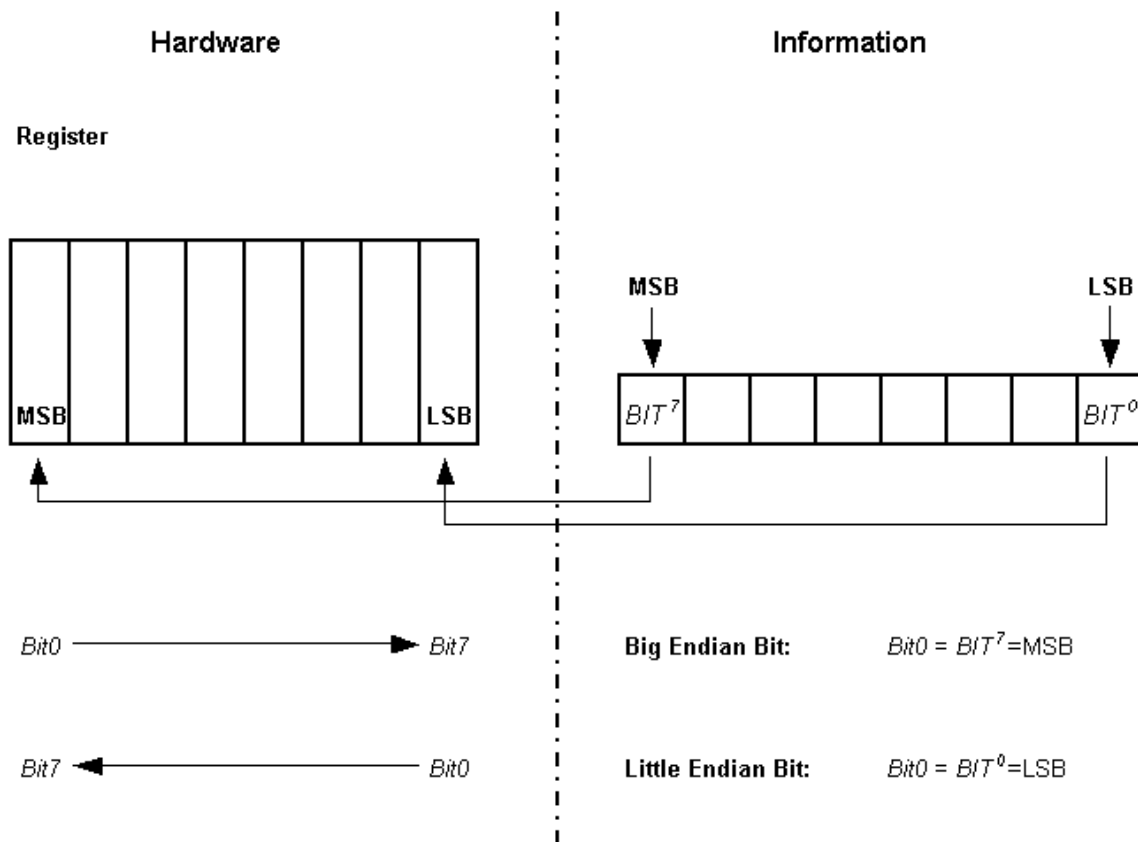


**Important Note:**

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 |
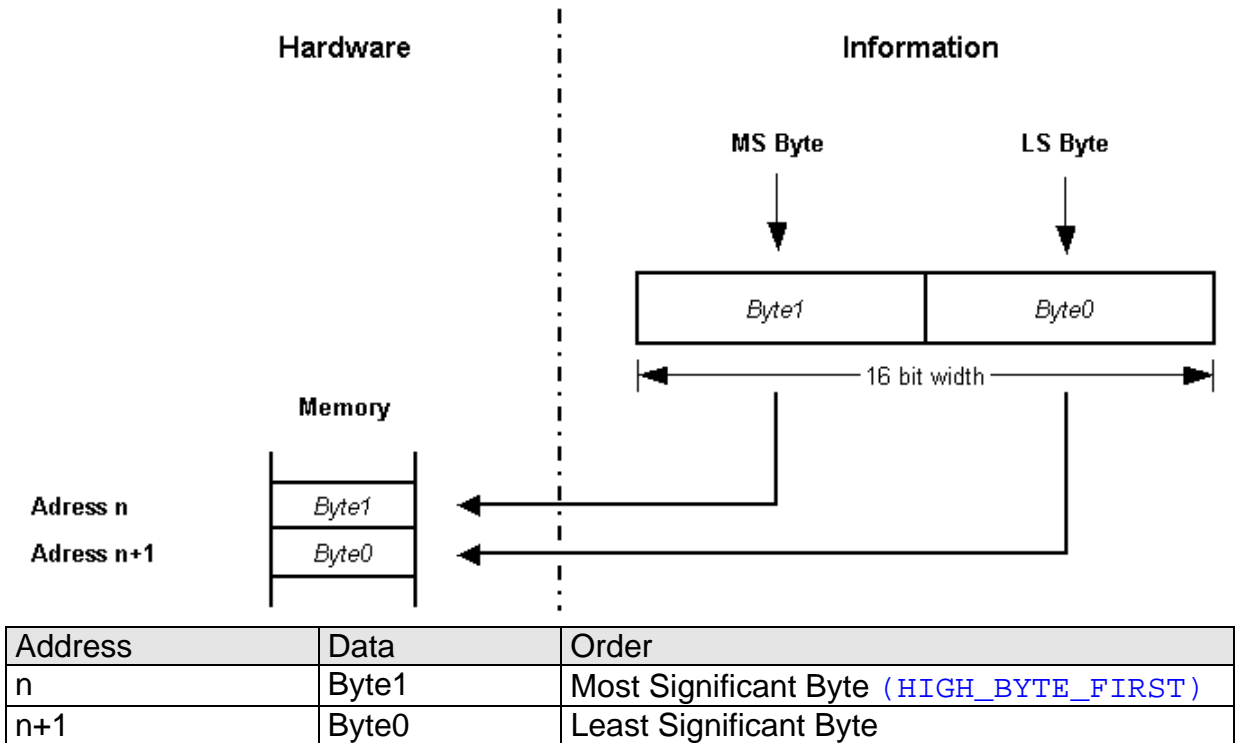
## Little Endian `(LOW_BYTE_FIRST)`



| Address | Data | Order |
|---------|-------|-------|
| n | Byte0 | Least Significant Byte `(LOW_BYTE_FIRST)` |
| n+1 | Byte1 | Most Significant Byte |
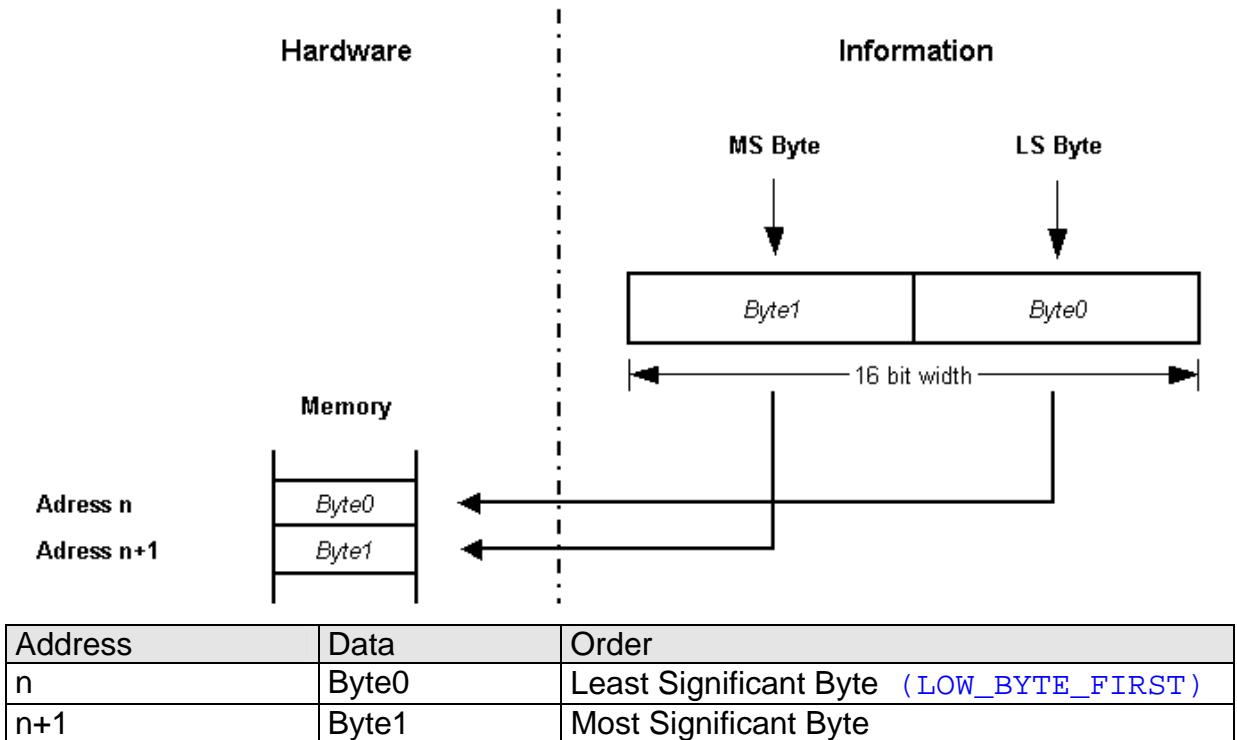
**Important Note:**

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 (`*_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).

**PLATFORM032**: The optimized AUTOSAR integer data types (`*_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 on basis of an unsigned integer. The type 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'.

**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;
…
var = TRUE;
var = FALSE;
```

The only allowed forms of comparison are
```
boolean var;
…
if (var == TRUE)
if (var == FALSE)
if (var != TRUE)
if (var != FALSE)
```

# 8 API specification

## 8.1 Imported types

Not applicable.

## 8.2 Type definitions

Type definitions

### 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. |

### 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. |

### 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. |

### 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. |

### 8.2.5 sint8

| Type: | Signed integer |
|---|---|
| Range: | -128..+127                            7 bit + 1 bit sign<br>0x80..0x7F |
| Description: | PLATFORM016: This standard AUTOSAR type shall be 8 bit signed. |

### 8.2.6 sint16

| Type: | Signed integer |
|---|---|
| Range: | -32768 ..+32767                  15 bit + 1 bit sign<br>0x8000..0x7FFF |
| Description: | PLATFORM017: This standard AUTOSAR type shall be 16 bit signed. |

### 8.2.7 sint32

| Type: | Signed integer |
|---|---|
| Range: | -2147483648 .. +2147483647       31 bit + 1 bit sign<br>0x80000000..0x7FFFFFFF |
| Description: | PLATFORM018: |

### 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. |

### 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. |

### 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. |

### 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. |

### 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. |

### 8.2.13 sint32_least

| Type: | Signed integer |
|---|---|
| Range: | At least -2147483648 ..   At least 31 bit + 1 bit sign +2147483647 |
| Description: | PLATFORM025 See PLATFORM005 for implementation and usage. |

### 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 field |
| | LSB_FIRST | The least significant bit is the first bit of the bit field |
| 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.<br>PLATFORM056: The symbols TRUE and FALSE shall be defined as follows:<br><br>```#ifndef TRUE```<br>``` #define TRUE   ((boolean) 1)```<br>```#endif```<br><br>```#ifndef FALSE```<br>``` #define FALSE  ((boolean) 0)```<br>```#endif```<br>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

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

| SWS Item | PLATFORM012 | |
|---|---|---|
| **Information elements** | | |
| **Information element name** | **Type / Range** | **Information element description** |
| PLATFORM_MODULE_ID | #define /uint8 | Module ID of this module from Module List |
| PLATFORM_AR_MAJOR_VERSION | #define /uint8 | Major version number of AUTOSAR specification on which the appropriate implementation is based on. |
| PLATFORM_AR_MINOR_VERSION | #define /uint8 | Minor version number of AUTOSAR specification on which the appropriate implementation is based on. |
| PLATFORM_AR_PATCH_VERSION | #define /uint8 | Patch level version number of AUTOSAR specification on which the appropriate implementation is based on. |

# 11 Changes to Release 1

## 11.1 Deleted SWS Items

PLATFORM052
PLATFORM053
PLATFORM040
PLATFORM047

## 11.2 Replaced SWS Items

Not applicable

## 11.3 Changed SWS Items

PLATFORM012

## 11.4 Added SWS Items

Not applicable

# 12 Annex

## 12.1 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
```

## 12.2 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;
```

## 12.3 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 short      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;
```

## 12.4 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 long       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;
```

## 12.5 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 long       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;
```

## 12.6 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 long       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;
```

## 12.7 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 long       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;
```

## 12.8 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 long      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;
```

## 12.9 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;
```

## 12.10 Type 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 int       bool;

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 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;
```