Document Title	Requirements on Libraries
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	314

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R24-11

Document Change History

Date	Release	Changed by	Description
2024-11-27	R24-11	AUTOSAR Release Management	• Addition of MSF library
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• No content changes • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Removal of the requirement SRS_LIBS_00006 • Addition of Requirements Tracing section • Addition of details about 64-bit CRC
2014-10-31	4.2.1	AUTOSAR Release Management	• Removed the section 5.1.7 • Added polynomial to CRC Library
2013-10-31	4.1.2	AUTOSAR Release Management	• Editorial changes • Reduced the scope of SRS_LIBS_08535: only provide the current element
2013-03-15	4.1.1	AUTOSAR Administration	• Formal rework of Requirements tracing • Fixed inconsistent Requirements table • Formal updates according to TPS Standardization Template • Link SRS requirements to new feature documents
2011-12-22	4.0.3	AUTOSAR Administration	• Typos correction: E2E instead of E2e (Chapter1, Page 6)
2010-09-30	3.1.5	AUTOSAR Administration	• Enlargement of the scope of the initial document to general library requirement • Introduction of Error handling • Introduction of E2E profiles • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised • Rename from "Requirements on CRC Routines" to "Requirements on Libraries"
2007-12-21	3.0.1	AUTOSAR Administration	• Initial Release: Split from Requirements on Memory Services

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.

Contents

1 Scope of Document

This document specifies requirements on the **AUTOSAR Libraries**. It applies to all the libraries specified by AUTOSAR:

Library short name:	***Description:***
Mfx	Library of **M**athematical **F**i**X**ed point calculations
Mfl	Library of **M**athematical **FL**oating point point calculations
Ifx	Library of **I**nterpolation functions of **F**i**X**ed point
Ifl	Library of **I**nterpolation functions of **FL**oating point
Bfx	Library of **B**it handling
Msf	Library of **M**emory **S**tandard **F**unctions
Efx	Library of **E**xtended functions on **F**ixed point
Crc	Library of CRC routines
E2E	SW-C End-to-End Communication Protection Library

Each of these libraries has its own independent SWS but this SRS document applies to all libraries.

Conformance to all requirements is mandatory for all implementations. "Configurable" also means, the requirement must be met, but such functionality can be disabled, if not needed in an ECU (BSW or SW-C).

This document was initially dedicated to CRC routines. In order to keep traceability, the CRC requirements are still present but in a dedicated chapter.

2 Conventions to be used

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see Standardization Template, chapter Support for Traceability ([1]).

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see Standardization Template, chapter Support for Traceability ([1]).

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective "LEGALLY REQUIRED", means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase "MUST NOT", means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective "REQUIRED", means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase "NOT RECOMMENDED", means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective "OPTIONAL", means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to Libraries that are not included in the AUTOSAR Glossary [2].

Abbreviation / Acronym:	Description:
API	Application Programming Interface. Preferred term is "function".
AR AR_	Abbreviation used in place of AUTOSAR
BFX	Library of Bit handling
CRC	Cyclic Redundancy check
DET	Default Error Tracer
E2E	End to End
EcuM	ECU Manager
EFX	Extended function on Fixed point
IFL	Interpolation function s of FLoating point
IFX	Interpolation function s of FiXed point
Library	Set of APIs (i.e. functions) that may be called from any modules (BSW modules or SW-C)
MFL	Mathematical FLoating point calculation
MFX	Mathematical FiXed point calculation
MSF	Memory Standard Functions
OS	Operating System

Table 3.1: Acronyms and abbreviations used in the scope of this Document

4 Requirements Specification

This chapter describes all requirements driving the work to define the Libraries.

4.1 Functional Overview

The AUTOSAR libraries provide other BSW modules and application SW-Cs with mathematical services.

The libraries offer C functions that can be called from source code, i.e. from BSW modules, from SW-C, from RTE or from Complex Drivers.

4.1.1 CRC library

The CRC Library provides functions for 8 bit, 16 bit, 32 bit and 64 bit CRC (cyclic redundancy check) calculations. The CRC library can be scaled in terms of

- Table based calculation (fast, but higher code size)
- Runtime calculation (slow, but smaller code size)
- Different standard CRC generator polynomials

Hardware supported CRC calculation is already supported by automotive microcontrollers.

4.1.2 SW-C End-to-End Communication Protection Library

The SW-C End-to-End Communication Protection Library (in short: E2E library) provides functions for detecting errors in (safety-related) communication between safety-related SW-Cs. The protection is done by means of protecting of safety-related data elements exchanged between SW-Cs, and the responsibility to protect/check the signal is given to the SW-Cs (application), which call directly the E2E library. The library is supposed to work over a priori any communication stack used for inter-SW-C communication, which are currently FlexRay, CAN and LIN. In future, when further communication stacks are added, there might be a need to add further E2E profiles.

4.2 Functional Requirements

4.2.1 Configuration

[SRS_LIBS_00001] The functional behavior of each library functions shall not be configurable

Upstream requirements: RS_BRF_02080

⌈

Description:	The functional behavior of library functions shall not be configurable. For some given inputs (c function parameters), a function shall return always the same outputs, as defined with the function specification. However, the internal behavior can be configurable. For examples: choose resource consumption strategy e.g. priority to CPU/RAM/ROM. But this is implementation specific and not standardized by AUTOSAR
Rationale:	A SW-C that uses a library function expects a deterministic and standardized behavior. It the SW integrator has the possibility to configure and change the behavior, the SW-C may react unexpectedly.
Use Case:	Division function. In case of division by zero, the function shall always return the same value. If the SW integrator has the possibility to configure this return value, it may be right for some SW-Cs but it may also be catastrophic for other SW-Cs.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_00015] It shall be possible to configure the microcontroller so that the library code is shared between all callers

Upstream requirements: RS_BRF_02072

⌈

Description:	In case memory partitioning is enabled on a given microcontroller, then it shall be possible to configure the microcontroller so that the library code is shared between all callers of the shared library.
Rationale:	This enables to reduce the Flash memory consumption.
Use Case:	In a partitioned system, SW-Cs in different partitions access the same library.
Dependencies:	–
Supporting Material:	–

⌋

4.2.2 Initialization

[SRS_LIBS_00002] A library shall be operational before all BSW modules and application SW-Cs

Upstream requirements: RS_BRF_01128

⌈

Rationale:	A Library function may be called at the very first step of ECU initialization, e.g. even by the OS or EcuM, thus the library shall be ready.
Use Case:	AUTOSAR OS initialization may call bit handling function
Dependencies:	–
Supporting Material:	–

⌋

4.2.3 Normal Operation

[SRS_LIBS_00004] Using libraries shall not pass through a port interface

Upstream requirements: RS_BRF_02080

⌈

Description:	SW-Cs shall directly invoke library functions, without passing through port RTE interface. To access the library API, the SW-Cs shall directly include the library header file.
Rationale:	The SW developer should be free to use libraries without having to change the SW-C interface description: reduce effort and inconsistencies The port+RTE mechanism is not required for library calls: no consistency check, no queuing, no communication outside ECU, etc. • Using library functions is a software designer decision. It is related to implementation, not to SW-C interface. • Calling a library function is an elemental operation. It is not like a client/server operation • because, they are often used, library function shall be called in an efficient way Thus, the function can be directly called from the source code, e.g. runnables, without using RTE API.
Use Case:	Application includes floating point arithmetic library header, and calls floating point routines in control loop calculations, without crossing RTE.
Dependencies:	–

▽

△

Supporting Material:	–

⌋

[SRS_LIBS_00005] Each library shall provide one header file with its public interface

Upstream requirements: RS_BRF_02080

⌈

Description:	Each library shall provide one header file with its public interface. This header shall declare all the public function prototypes and types defined by the library specification. The header file shall be named like: <library short name>.h
Rationale:	Access to function prototypes and types Standardization of header file name
Use Case:	#include "AR_MFX.h"
Dependencies:	[SRS_LIBS_00004]
Supporting Material:	–

⌋

[SRS_LIBS_00009] All library functions shall be re-entrant

Upstream requirements: RS_BRF_02088

⌈

Description:	All library functions shall be re-entrant, which means that they shall be able to handle several simultaneous, interleaved or/and concurrent requests.. A function, in order to be re-entrant, it (1) shall not call any non-re-entrant functions, (2) shall not write any global nor static variables. If some kind of data shall be handled, the callers have to create (define) them and pass them as function parameters. A library function only runs in the context of the caller, on the core where it is called, in the same protection environment. A library function can only call library functions. A library function is synchronous, e.g. it does not have wait points.
Rationale:	Avoid consistency mechanisms which bring to weak efficiency
Use Case:	Multitasking environments Every BSW modules and SW-C will use the same functions in different tasks

▽

△

Supporting Material:	–

⌋

[SRS_LIBS_00010] A library shall define its own specific types in the library header file if and only if they are not yet defined by AUTOSAR ⌈

Description:	A library shall define (typedef) its own specific types in the library header file if and only if they are not yet defined by AUTOSAR in std_types.h and platform_types.h. These types shall be identified in the corresponding SWS by the name and the description, but not the actual implementation. No new implementation-specific types are allowed in the implementation of the public library interface, i.e. other types that are not specified in the SWS shall not be present. The implementation (typedef) can be different, e.g. according the platform. The caller shall not rely on any implementation. Using C operators on these specific types is forbidden.
Rationale:	A library may handle some specific types not defined by AUTOSAR Only SWS specified type are allowed in order to insure code portability independent from any specific AUTOSAR library implementation
Use Case:	Types u64, S64 for the 64bits data math library
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_00011] All function names and type names shall start with "Library short name_"

Upstream requirements: RS_BRF_01024

⌈

Description:	All function names and type names shall start with "Library short name_"
Rationale:	Avoid collision with already existing library Quickly identified AUTOSAR library calls in the code
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_00012] Passing parameters with structure shall be allowed

Upstream requirements: RS_BRF_02080

⌈

Rationale:	Function calls become simpler. If a set of fixed parameters is required. They can be defined once in a structure and may be used several times. In some circumstances, it makes sense to group several function parameters into one of few structures: • when there are many parameters, • when some parameters can be grouped by functionality

Use Case:	`sint16 EFX_PGOV_WIN ( sint32 X,`
	`sint32 Kp,`
	`sint32 KpPos,`
	`sint32 KpNeg,`
	`sint32 WinPos,`
	`sint32 WinNeg )`
	
	`sint16 EFX_PGOV_WIN (sint32 X, const PWin_Type * Struct)`

Dependencies:	[SRS_LIBS_00008]: if a parameter is added to the structure, a new structure name and a new function name shall be defined. So there is no risk that a new structure field would be forgotten in case of library evolution.
Supporting Material:	–

⌋

4.2.4 Shutdown Operation

[SRS_LIBS_00003] A library shall be operational until the shutdown

Upstream requirements: RS_BRF_01240, RS_BRF_02088

⌈

Description:	A library shall be operational until the shutdown .A library should not require a shutdown operation phase. If so, it shall occur after all AUTOSAR BSW modules shutdown operations
Rationale:	A Library function may be called at the very latest step of ECU shutdown, e.g. even by the OS, thus the library shall be ready until the end
Use Case:	AUTOSAR OS shutdown operation may call bit handling function
Dependencies:	–
Supporting Material:	–

⌋

4.2.5 Fault Operation

[SRS_LIBS_00013] The error cases, resulting in the check at runtime of the value of input parameters, shall be listed in SWS

Upstream requirements: RS_BRF_01440

⌈

Description:	Function should check at runtime (both in production and development code) the value of input parameters, especially cases where erroneous value can bring to fatal error or unpredictable result, if they have the values allowed by the function specification. All the error cases shall be listed in SWS and the function should return a specified value (in SWS) that is not configurable. This value is dependant of the function and the error case so it is determined case by case.
Rationale:	Avoid fatal error, provide standardized behaviour
Use Case:	Division by zero, negative number of a square root, out of range, overflow, underflow, etc.
Dependencies:	[SRS_LIBS_00001]
Supporting Material:	–

⌋

4.2.6 CRC library

[SRS_LIBS_08525] The CRC library shall support the standard generator polynomials

Upstream requirements: RS_BRF_02096

⌈

Description:	The CRC library shall support the following generator polynomials: CRC 8-bit SAE-J850 (0x1D) CRC 16-bit CRC-CCITT CRC 32-bit Ethernet IEEE-802 (0x04C11DB7) CRC 32-bit 0xF4ACFB13 CRC 64-bit ECMA 0x42F0E1EBA9EA3693
Rationale:	These polynomials are considered to be standard.

▽

△

Use Case:	CRC 8-bit : Detect errors/inconsistent data in CAN/LIN/FlexRay Communication CRC16 and 32-bit : Detect errors/inconsistent data in NVRAM/RAM and FlexRay/Ethernet communication CRC64: Detect errors/inconsistent data in NVRAM/RAM and ethernet/wireless communication
Dependencies:	–
Supporting Material:	–

⌋

4.3 Non-Functional Requirements (Qualities)

4.3.1 General

[SRS_LIBS_08518] The CRC Library shall provide different calculation methods, optimizing either performance or memory usage

Upstream requirements: RS_BRF_02096

⌈

Description:	The CRC Library shall provide different calculation methods (algorithm), optimizing either performance or memory usage (e.g. for runtime calculation).
Rationale:	Allow for optimization of code size or execution time depending on specific ECU requirements.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_08526] The CRC Library shall support current standards of CRC calculation

Upstream requirements: RS_BRF_02096

⌈

Description:	The CRC Library shall support current standards of CRC calculation: • table based • runtime calculated • hardware based (may be supported in the future)
Rationale:	Allow for optimization of code size or execution time depending on specific ECU requirements.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_08521] All CRC routines shall allow step-by-step-wise calculation of a large data block

Upstream requirements: RS_BRF_02096

⌈

Description:	All CRC routines shall allow step-by-step-wise calculation of a large data block that is passed by start address, length and start value.
Rationale:	CRC calculation of large data blocks without blocking the whole system.
Use Case:	Example: The CRC calculation of a 4k ROM data block shall be performed. If the CRC routine would calculate the WHOLE block within one call, the watchdog would not be triggered anymore and cause a reset. Thus, the calculation has to be done in several steps (e.g. 16 byte wise)
Dependencies:	–
Supporting Material:	–

⌋

4.3.2 CRC library

4.3.2.1 Compatibility

[SRS_LIBS_00008] For a given function prototype name, the behavior and the parameters shall not evolve once it is a part of an AUTOSAR final release

Upstream requirements: RS_BRF_02096

⌈

Description:	For a given function prototype name, the behavior and the parameters shall not evolve once it is a part of an AUTOSAR final release. For any reasons, if the specification of a behavior has to be changed in AUTOSAR release N+1, e.g. the WP LIBRARIES decides of a new feature, or a parameter/type has to be added/removed/modified from the specification, and if library implementations might be already in production, then a new function prototype shall be created. The previous one (AUTOSAR release N) shall be kept unchanged. In case of a type implementation is specified in SWS, then if this implementation changes, a new type name and API name shall be created and the previous one shall be retained. In case of a type implementation is not defined in SWS, then the library developer is free to change the implementation without creating new type name.
Rationale:	Avoid SW-C re-developing in case of library evolution. Avoid having to integrate new library releases if SW-C do not require new function. if an integrator has to add different SW-C which rely on different libraries(AUTOSAR specification) versions, he can (has to) purchase the latest library version since the ascending compatibility is insured.
Use Case:	On a project, the integrator uses a library compatible with AR4.0, and a SW-C that use this library. On a next V cycle, a new library release compatible with AR4.1 is integrated. With the ascending compatibility rule, it is ensured that the SW-C will react in the same way even if new functions have been added.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_00016] A SW-C may use a non-AUTOSAR library available on the market

Upstream requirements: RS_BRF_02096

⌈

Description:	A SW-C may use a non-AUTOSAR library available on the market. In this case, the developer may not deliver this library but just mention it in the SW-C template (DependencyOnLibrary). It is the responsibility of the integrator to get this library to be able to integrate the SW-C. This non-AUTOSAR library should respect the requirements of this document. It is recommended that non-AUTOSAR functions start with a specific vendor prefix
Rationale:	Allows freedom of implementation. Avoid name collisions.
Use Case:	Supplier specific libraries.
Dependencies:	[SRS_LIBS_00011]
Supporting Material:	–

⌋

4.3.2.2 Others

[SRS_LIBS_00007] Using a library should be documented

Upstream requirements: RS_BRF_01192

⌈

Description:	If a BSW module or a SW-C uses a Library, the developer should add an Implementation-DependencyOnLibrary in the BSW/SW-C template. minVersion and maxVersion parameters correspond to the supplier version. In case of AUTOSAR library, these parameters may be left empty because a SW-C or BSW module may rely on a library behaviour, not on a supplier implementation. However, the SW-C or BSW modules shall be compatible with the AUTOSAR platform where they are integrated.
Rationale:	SW integrator to checks the AUTOSAR platform compatibilities while integrating a BSW module or a SW-C
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_00017] Usage of macros should be avoided ⌈

Description:	The function should be declared as function or inline function. Macro #define should not be used
Rationale:	Macros do not specify argument type and return type so there is more chance of improper use
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_LIBS_00018] A library function may only call library functions ⌈

Description:	A library function shall not call any BSW modules functions, e.g. the DET. A library function can call other library functions.
Rationale:	A library function shall be re-entrant. Other BSW modules functions may not be re-entrant.
Use Case:	Multi-core architecture Memory protection scheme
Dependencies:	[SRS_LIBS_00009]
Supporting Material:	–

⌋

[RS_LIBS_00019] Memory handling library

Upstream requirements: RS_BRF_02072

⌈

Description:	AUTOSAR shall offer memory handling functions similar to those found in standard C libraries.
Rationale:	Standard C libraries are available as part of C compiler environments. Unfortunetally such libraries may interfere with the AUTOSAR environment and cannot be used in all situations. Therefore, AUTOSAR offers own variants of it. E.g. standard C types like size_t or int shall be avoided in AUTOSAR.
Use Case:	–
Dependencies:	[SRS_LIBS_00009]
Supporting Material:	–

⌋

5 Requirements Tracing

The following table references the features specified in [3] and links to the fulfillments of these.

Requirement	Description	Satisfied by
[RS_BRF_01024]	AUTOSAR shall provide naming rules for public symbols	[SRS_LIBS_00011]
[RS_BRF_01128]	AUTOSAR shall allow software components to be started before all BSW modules are initialized	[SRS_LIBS_00002]
[RS_BRF_01192]	AUTOSAR shall document all architectural constraints which exist to use the RTE and the BSW	[SRS_LIBS_00007]
[RS_BRF_01240]	AUTOSAR OS shall support communication between OSApplications	[SRS_LIBS_00003]
[RS_BRF_01440]	AUTOSAR services shall support system diagnostic functionality	[SRS_LIBS_00013]
[RS_BRF_02072]	AUTOSAR shall provide generic functionality which is in wide use in the automotive domain as libraries	[RS_LIBS_00019] [SRS_LIBS_00015]
[RS_BRF_02080]	AUTOSAR libraries shall use C interfaces	[SRS_LIBS_00001] [SRS_LIBS_00004] [SRS_LIBS_00005] [SRS_LIBS_00012]
[RS_BRF_02088]	AUTOSAR library functionality shall be reentrant	[SRS_LIBS_00003] [SRS_LIBS_00009]
[RS_BRF_02096]	AUTOSAR shall provide checksum computation of cyclic redundancy check sums as a library	[SRS_LIBS_00008] [SRS_LIBS_00016] [SRS_LIBS_08518] [SRS_LIBS_08521] [SRS_LIBS_08525] [SRS_LIBS_08526]

Table 5.1: Requirements Tracing

6 References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Requirements on AUTOSAR Features
AUTOSAR_CP_RS_Features