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

Document Version	1.1.1
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change description
28.02.2014	1.1.1	AUTOSAR Release Management	• Editorial change
27.04.2011	1.1.0	AUTOSAR Administration	• Back-porting of E2E Communication • Legal disclaimer revised
23.06.2007	1.0.1	AUTOSAR Administration	• Legal disclaimer revised • Rename from “Requirements on CRC Routines” to “Requirements on Libraries”
13.12.2007	1.0.0	AUTOSAR Administration	Initial Release: Split from Requirements on Memory Services

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of content

1 Scope of this document

This document specifies requirements on Basic Software Modules of the following software layers:

- Service Layer

Those modules are of the following type:

- NVRAM Management
- Interfaces

The selection of modules is derived from the Basic Software Module List and the AUTOSAR Layered Software Architecture. The following modules are in scope:

- CRC

The requirements are structured in the following way:

- General requirements on Basic Software Modules (other document)
- General requirements which apply to all modules of the CRC Management
- Module specific requirements

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).

Constraints

First scope for specification of requirements on basic software modules is systems which are not safety relevant. For this reason safety requirements are assigned to medium priority.

2 How to read this document

Each requirement has its unique identifier starting with the prefix “BSW” (for “Basic Software”). For any review annotations, remarks or questions, please refer to this unique ID rather than chapter or page numbers!

2.1 Conventions used

In requirements, the following specific semantics are used (taken from Request for Change RFC 2119 from the Internet Engineering Task Force IETF)

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 described in RFC 2119. Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
- MUST NOT: This phrase, or the phrase „SHALL NOT“, means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective "RECOMMENDED", mean 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" mean 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, MUST 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, MUST be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides.)

2.2 Requirements structure

Each module specific chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped under the following headlines (where applicable):

Functional Requirements:
- Configuration (which elements of the module need to be configurable)
- Initialization
- Normal Operation
- Shutdown Operation
- Fault Operation
- ...

Non-Functional Requirements:
- Timing Requirements
- Resource Usage
- Usability
- Output for other WPs (e.g. Description Templates, Tooling, ...)
- ...

3 Acronyms and abbreviations

Acronym:	*Description:*
CRC	Cyclic Redundancy Check
E2E	End to End

4 Requirement Specification

4.1 CRC Library

4.1.1 Functional Overview

The CRC Library provides functions for 8 bit, 16 bit and 32 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 may be supported in the future.

4.1.2 Functional Requirements

4.1.2.1 General

4.1.2.1.1 [BSW08524] Error detection

Initiator:	BMW
Date:	10.03.2005
Short Description:	Error detection
Type:	New
Importance:	High
Description:	The chosen polynomial shall detect the following errors: • single bit errors • two-bit errors • errors with an odd number of bits or at least three-bit error detection
Rationale:	An ordinary checksum would provide single bit error detection with less complexity in calculation. Detection of parity errors is easily implemented by using a simple parity bit. Therefore at least three-bit errors should be detected.
Use Case:	Detect errors/inconsistent data in NVRAM/RAM.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2.1.2 [BSW08525] Support of standard polynomials

Initiator:	BMW
Date:	10.03.2005
Short Description:	Support of standard polynomials
Type:	New
Importance:	High
Description:	The CRC library shall support the following generator polynomials: • CRC 8-bit SAE-J1850 (0x1D) • CRC 16-bit CRC-CCITT (0x1021) • CRC 32-bit Ethernet IEEE-802 (0x04C11DB7)
Rationale:	These polynomials are considered to be standard.

Use Case:	CRC 8-bit : Detect errors/inconsistent data in Communication CRC16 and 32-bit : Detect errors/inconsistent data in NVRAM/RAM.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2 Configuration

4.1.2.2.1 [BSW08518] CRC calculation method complexity

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC calculation method complexity
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

4.1.2.2.2 [BSW08526] CRC standard calculation methods

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC standard calculation methods
Type:	New
Importance:	High
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:	--
Conflicts:	--
Supporting Material:	--

5 Requirements Specification of SW-C End-to-End Communication Protection Library

5.1 Functional Overview

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.

5.2 Functional Requirements

5.2.1 General

5.2.1.1 [BSW08527] Protection mechanisms for inter SW-C communication up to ASIL D

Initiator:	BMW
Date:	03.09.2008
Short Description:	Protection mechanisms for inter SW-C communication up to ASIL D
Type:	New
Importance:	High
Description:	E2E library shall provide a set of safety protocols, in a form of library functions invoked by SW-Cs. The protocols shall provide the error detection rate sufficient for transmitting safety-related data up to ASIL D.
Rationale:	E2E communication protection is state-of-art in automotive safety-related series products.
Use Case:	Communication between main chassis ECU SW-C and power steering ECU SW-C
Dependencies:	None
Conflicts:	None

5.2.1.2 [BSW08528] E2E profiles

Initiator:	BMW
Date:	03.09.2008
Short Description:	E2E profiles
Type:	New
Importance:	High
Description:	E2E library shall provide E2E profiles, where each E2E profile completely

	defines a particular safety protocol (including header structure, behavior as state machines, error handling etc). Each E2E profile shall be an efficient solution for a particular communication stack used underneath (which are either FlexRay, CAN, or LIN), and the ASIL rating of the exchanged signals. Note: Each communication stack (e.g. FlexRay) has different error rates which depend on for example: - Bit error rate on channel - FIT values of HW - number of ECUs - topology (e.g. CAN->Gateway->FR) - open/closed transmission system - frequency of safety related messages The profiles, based on proven-in-use solutions, are supposed to cover typical combinations of above factors.
Rationale:	Too many standardized profiles reduce interoperability between applications. Moreover, it introduces too much specification and development efforts.
Use Case:	Protocol with 8-bit CRC for CAN, and 16-bit for long FlexRay signals.
Dependencies:	None
Conflicts:	None

5.2.1.3 [BSW08529] Available mechanisms in the E2E profiles

Initiator:	BMW
Date:	03.09.2008
Short Description:	Available mechanisms in the E2E profiles
Type:	New
Importance:	High
Description:	Each of the defined E2E profiles shall use an appropriate subset of the following mechanisms: 1. Sequence number (different sizes possible; in the state-of art it is alternatively called alive counter or consecutive number 2. Checksum: CRC8, CRC16, CRC32 3. IDs: Source ID, Destination ID, Data ID 4. Timeouts: reception timeout In other words, mechanisms not listed shall not be used. In each E2E profile, the sequence number and IDs, if used, should be all part of the transmitted data element. However, it is allowed that in a given profile, the sequence number and/or IDs are “hidden” (not transmitted), but included in the checksum.
Rationale:	These are typical measures used by safety protocols, and they can be realized by AUTOSAR.
Use Case:	Mechanisms used in an exemplary profile: 4-bit sequence counter, CRC8, Data ID, timeout
Dependencies:	None
Conflicts:	None

5.2.1.4 [BSW08530] Profile definition and interoperability

Initiator:	BMW
Date:	03.09.2008
Short Description:	Profile definition and interoperability
Type:	New
Importance:	High

Description:	Each E2Eprofile defined within the library shall: 1. Have a unique ID (IDs from E2E_01 to E2E_16 are reserved for standard AUTOSAR profiles). 2. Define precisely a set of mechanisms (e.g. CRC of a particular polynomial) 3. Define its behavior in a semi-formal way (including state machines, error handling etc).
Rationale:	A protocol is not just a list of mechanisms (e.g CRC8 + sequence number) , but the whole logic managing the process. Standardization of header is by far not sufficient. Standardized behaviour is needed to achieve interoperability.
Use Case:	Usually one state machine per profile per communicating partner (sender, receiver, client server) is sufficient. ECU1 and ECU2 communicating. ECU1 has different implementation of E2E library than ECU2.
Dependencies:	None
Conflicts:	None

5.2.1.5 [BSW08531] Reuse of CRC routines

Initiator:	BMW
Date:	03.09.2008
Short Description:	Reuse of CRC routines
Type:	New
Importance:	High
Description:	E2E library shall not provide CRC routine implementations. Instead, it shall call the CRC routines of CRC library (document UID 016).
Rationale:	Reuse of existing AUTOSAR functionality
Use Case:	CRC8 of CRC library to be used in one of profiles for protecting CAN communication.
Dependencies:	None
Conflicts:	None

5.2.1.6 [BSW08533] E2E CRC polynomials to be different to those used by network stack

Initiator:	Fiat / CRF
Date:	03.09.2008
Short Description:	E2E CRC polynomials to be different to those used by network stack
Type:	New
Importance:	High
Description:	CRC used in each E2E profile shall be different than the CRC used by the underlying communication protocols (FlexRay, CAN, LIN), for which the given profile is supposed to be used with.
Rationale:	Using the same polynomials twice (once in com stack and again in E2E) provides significantly lower joint detection rate than using two different polynomials. The polynomials available in AUTOSAR R3.1 are not optimal for E2E anyway.
Use Case:	If profile X is supposed to be used only for FlexRay, then its CRC shall be different than the one of FlexRay.
Dependencies:	None
Conflicts:	None

5.2.1.7 [BSW08534] Separate Error flag and error counters

Initiator:	Fiat/CRF
Date:	09.09.2008
Short Description:	Separate Error flag and error counters
Type:	New
Importance:	High
Description:	E2E library shall provide to the application layer separate errors flag and error counters for each type of detected communication failure. In other words if E2E profile X is supposed to use the sequence counter and CRC, then the following error flag shall be available to the application layer: • Data corruption • Wrong sequence • Repetition • Data loss
Rationale:	Error handling strategies are “application dependent”, and cannot be “a priory defined”
Use Case:	Enable error-dependent reaction of the SW-C using E2E library.
Dependencies:	None
Conflicts:	None

5.2.1.8 [BSW08535] Data element provided to the application layer

Initiator:	Fiat/CRF
Date:	09.09.2008
Short Description:	Data element provided to the application layer
Type:	New
Importance:	medium
Description:	E2E library should provide to the application layer the last received data element and the last correct data element.
Rationale:	Provision of means to design different error reaction strategies based upon the type of detected failure
Use Case:	Use of the last correct data in case of data corruption. Use of interpolation algorithm between last received data and last correct data in case of data loss
Dependencies:	6.2.1.8
Conflicts:	None

5.2.1.9 [BSW08536] CRC computation over E2E protection mechanism

Initiator:	Fiat / CRF
Date:	16.09.2008
Short Description:	CRC computation over E2E protection mechanism
Type:	New
Importance:	High
Description:	Either SW-C or E2E Library shall compute the intermediate CRC over application data element. E2E library shall use as initial CRC value the intermediate CRC and shall compute the CRC over the sequence counter (if it is used) and IDs (if used).
Rationale:	In case of complex data elements, the E2E library cannot compute the CRC over the data element (because the library does not know the layout of the data element – a data type may be e.g. an array of pointers to data structures, which does not occupy a consecutive address space). In such a case, the application needs to compute the CRC over the data element, and

	pass the computed CRC to the library. However, regardless who invokes the CRC computation (SW-C or library), the CRC used is the one of the used E2E profile.
Use Case:	
Dependencies:	None
Conflicts:	None

5.2.1.10 [BSW08537] Tolerance in SW-Cs against undetected single errors in signals

Initiator:	Exida
Date:	10.12.2008
Short Description:	Tolerance in SW-Cs against undetected single errors in signals
Type:	New
Importance:	High
Description:	SW-Cs shall tolerate at least one incorrect signal, in which error has not been detected by SW-Cs.
Rationale:	Requiring that 100% errors are detected by E2E protocol has high impact on implementation of E2E library (e.g. requiring SW or/and HW redundancy). Allowing to have a signal (in a sequence of received signals) with an error that is not detected by E2E
Use Case:	Example 1: multiple bit errors (e.g. 5 corrupted bits) that generate the same CRC as the original signal. Example 2: random HW fault or SW fault in E2E library causing that CRC Sequence Counter computation does not detect an error.
Dependencies:	The tolerance can be reached by means of an appropriate safety concept of an application.
Conflicts:	None

5.2.2 Non-Functional Requirements (Qualities)

5.2.2.1 General

5.2.2.1.1 [BSW08520] CRC routine reentrancy

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC routine reentrancy
Type:	New
Importance:	High
Description:	All CRC calculation routines shall be re-entrant.
Rationale:	Quasi-parallel access to the CRC routines shall be allowed from several clients within a multi tasking environment.
Use Case:	NVRAM-Manager, application and Flash Driver are using CRC routines in parallel.
Dependencies:	--
Conflicts:	--
Supporting Material:	--

5.2.2.1.2 [BSW08521] CRC routine schedulability

Initiator:	BMW
Date:	18.02.2005
Short Description:	CRC routine schedulability
Type:	New
Importance:	High
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:	General requirement: [BSW00313] Deadlock prevention
Conflicts:	--
Supporting Material:	--

6 References

6.1 Deliverables of AUTOSAR

[AUTOSAR_SW_ARCH] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[AUTOSAR_BASIC_SW] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf