

<b>Document Title</b>	Specification of CRC Routines
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	016
<b>Document Classification</b>	Standard

<b>Document Version</b>	3.1.0
<b>Document Status</b>	Final
<b>Part of Release</b>	3.1
<b>Revision</b>	5

<b>Document Change History</b>			
<b>Date</b>	<b>Version</b>	<b>Changed by</b>	<b>Change Description</b>
10.09.2010	3.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Check value for J1850 CRC8 changed from F4h to 4Bh</li> <li>• Legal disclaimer revised</li> </ul>
23.06.2008	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Legal disclaimer revised</li> </ul>
22.01.2008	3.0.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Separated CRC requirements from Memory Services Requirements</li> <li>• CRC8 management added</li> </ul>
31.10.2007	2.1.2	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Document meta information extended</li> <li>• Small layout adaptations made</li> </ul>
05.06.2007	2.1.2	AUTOSAR Technical Office	Tables generated from UML-models, UML-diagrams linked to UML-model, general improvements of requirements in preparation of CT-development. No changes in the technical contents of the specification.
24.01.2007	2.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• “Advice for users” revised</li> <li>• “Revision Information” added</li> </ul>
15.12.2006	2.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Crc_CalculateCRC16 and Crc_CalculateCRC32 APIs, Crc_DataPtr parameter : void pointer changed to uint8 pointer</li> <li>• Legal disclaimer revised</li> </ul>
28.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template. <ul style="list-style-type: none"> <li>• UML model introduction</li> <li>• Requirements traceability update</li> <li>• Reentrancy at calculating CRC with hardware support</li> </ul>
31.05.2005	1.0.0	AUTOSAR Administration	Initial Release

## Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

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

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Advice for users

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

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

## Table of Contents

1	Introduction and functional overview .....	5
2	Acronyms and abbreviations .....	6
3	Related documentation.....	7
3.1	Input documents.....	7
3.2	Related standards and norms .....	7
4	Constraints and assumptions .....	8
4.1	Limitations .....	8
4.2	Applicability to car domains.....	8
5	Dependencies to other modules.....	9
5.1	File structure .....	9
6	Requirements traceability.....	10
7	Functional specification .....	15
7.1	Basic Concepts of CRC Codes .....	15
7.1.1	Mathematical Description .....	15
7.1.2	Euclidian Algorithm for Binary Polynomials and Bit-Sequences .....	17
7.1.3	CRC calculation, Variations and Parameter .....	18
7.2	Standard parameters.....	18
7.2.1	8-bit CRC calculation .....	19
7.2.2	16-bit CRC calculation .....	19
7.2.3	32-bit CRC calculation .....	19
7.3	General behavior.....	21
7.4	Error classification.....	21
7.5	Error detection.....	21
7.6	Error notification .....	21
7.7	Version check.....	21
8	API specification.....	22
8.1	Imported types.....	22
8.2	Type definitions .....	22
8.3	Function definitions .....	22
8.3.1	8-bit CRC Calculation .....	22
8.3.2	16-bit CRC Calculation .....	23
8.3.3	32-bit CRC Calculation .....	24
8.3.4	Crc_GetVersionInfo .....	24
8.4	Call-back notifications .....	25
8.5	Scheduled functions.....	25
8.6	Expected Interfaces.....	25
8.6.1	Mandatory Interfaces .....	25
8.6.2	Optional Interfaces.....	25
8.6.3	Configurable interfaces.....	26
9	Sequence diagrams .....	27
9.1	Crc_CalculateCRC8 ().....	27

9.2	Crc_CalculateCRC16().....	27
9.3	Crc_CalculateCRC32().....	28
10	Configuration specification.....	29
10.1	How to read this chapter.....	29
10.1.1	Configuration and configuration parameters.....	29
10.1.2	Variants.....	29
10.1.3	Containers.....	29
10.2	Containers and configuration parameters.....	30
10.2.1	Variants.....	30
10.2.2	Crc.....	30
10.2.3	CrcGeneral.....	30
10.3	Published Information.....	32
10.3.1	CrcPublishedInformation.....	32
11	Changes to Release 1.....	34
11.1	Deleted SWS Items.....	34
11.2	Replaced SWS Items.....	34
11.3	Changed SWS Items.....	34
11.4	Added SWS Items.....	34
12	Changes during SWS Improvements by Technical Office.....	35
12.1	Deleted SWS Items.....	35
12.2	Replaced SWS Items.....	35
12.3	Changed SWS Items.....	35
12.4	Added SWS Items.....	35
13	Changes to Release 2.....	36
13.1	Deleted SWS Items.....	36
13.2	Replaced SWS Items.....	36
13.3	Changed SWS Items.....	36
13.4	Added SWS Items.....	36

## 1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module CRC.

The CRC library contains the following routines for CRC calculation:

- CRC8
- CRC16
- CRC32

For both routines (CRC8, CRC16 and CRC32), the following calculation methods are possible:

- Table based calculation:  
Fast execution, but larger code size (ROM table)
- Runtime calculation:  
Slower execution, but small code size (no ROM table)
- Hardware supported CRC calculation (device specific):  
Fast execution, less CPU time

All routines are re-entrant and can be used by multiple applications at the same time. Hardware supported CRC calculation may be supported by some devices in the future.

## 2 Acronyms and abbreviations

Acronyms and abbreviations, which have a local scope and therefore are not contained in the AUTOSAR glossary, must appear in a local glossary.

<b><i>Abbreviation / Acronym:</i></b>	<b><i>Description:</i></b>
CRC	Cyclic Redundancy Check
ALU	Arithmetic Logic Unit

### **3 Related documentation**

#### **3.1 Input documents**

- [1] List of Basic Software Modules,  
AUTOSAR\_BasicSoftwareModules.pdf
- [2] Layered Software Architecture,  
AUTOSAR\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules,  
AUTOSAR\_SRS\_General.pdf
- [4] Requirements on CRC Routines,  
AUTOSAR\_SRS\_CRC\_Routines.pdf
- [5] Specification of ECU Configuration,  
AUTOSAR\_ECU\_Configuration.pdf
- [6] A Painless Guide To CRC Error Detection Algorithms, Ross N. Williams
- [7] AUTOSAR Basic Software Module Description Template,  
AUTOSAR\_BSW\_Module\_Description.pdf

#### **3.2 Related standards and norms**

- [7] SAE-J1850 8-bit CRC
- [8] CCITT 16-bit CRC
- [9] IEEE 802.3 Ethernet 32-bit CRC

## **4 Constraints and assumptions**

### **4.1 Limitations**

No limitations.

### **4.2 Applicability to car domains**

No restrictions.

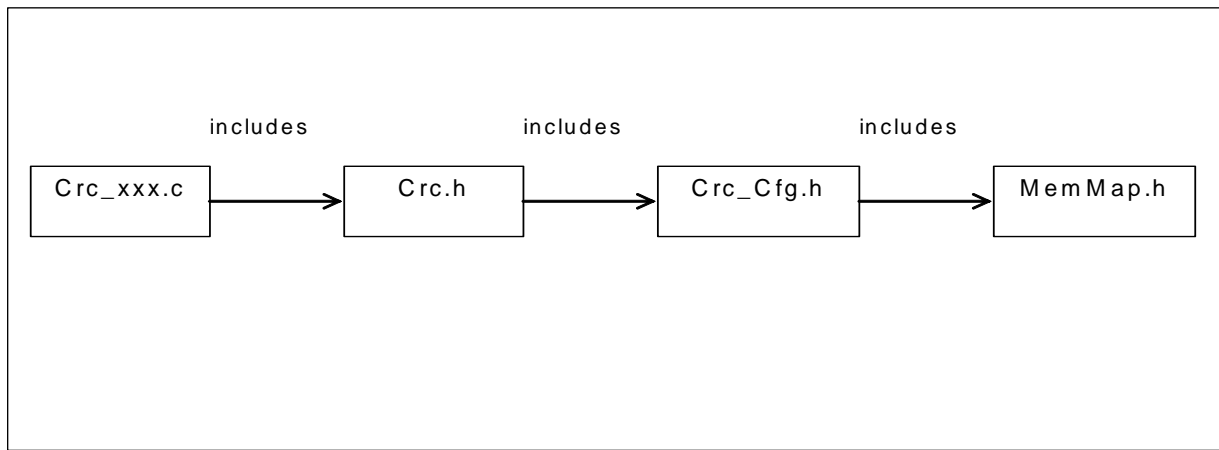


## 5 Dependencies to other modules

### 5.1 File structure

**CRC024:** The Crc module shall provide the following files:

- C file `Crc_xxx.c` containing parts of CRC code
- An API interface `Crc.h` providing the function prototypes to access the library CRC functions
- A header file `Crc_Cfg.h` providing specific parameters for the CRC.



**Figure 1: File structure**

**CRC022:** The Crc module shall comply with the following include file structure:

- `Crc.h` shall include `Crc_Cfg.h` and `MemMap.h`
- `Crc_xxx.c` shall include `Crc.h`

**CRC023:** Users of the Crc module (e.g. NVRAM Manager) shall only include `Crc`.

## 6 Requirements traceability

Document: General Requirements on Basic Software

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00344] Reference to link-time configuration	Not applicable (no use case)
[BSW00404] Reference to post build time configuration	Not applicable (no use case)
[BSW00405] Reference to multiple configuration sets	Not applicable (no use case)
[BSW00345] Storage of Pre-compile-time configuration	<a href="#">CRC006</a>
[BSW159] Tool-based configuration	<a href="#">CRC006</a>
[BSW167] Static configuration checking	Requirement on configuration tool
[BSW171] Configurability of optional functionality	<a href="#">CRC006</a>
[BSW170] Data for reconfiguration of AUTOSAR SW-components	Not applicable (no use case)
[BSW00380] Separate C-Files for configuration parameters	Not applicable (no use case)
[BSW00419] Separate C-Files for pre-compile time configuration parameters	<a href="#">CRC006</a>
[BSW00381] Separate H-Files for configuration parameters	<a href="#">CRC006</a>
[BSW00412] Separate H-File for configuration parameters	Not applicable (no use case)
[BSW00383] List dependencies of configuration files	Not applicable (no use case)
[BSW00384] List dependencies to other modules	Not applicable (no use case)
[BSW00387] Specify the configuration class of callback function	Not applicable (no use case)
[BSW00388] Introduce containers	Not applicable (no use case)
[BSW00389] Containers shall have names	Not applicable (no use case)
[BSW00390] Parameter content shall be unique within the module	<a href="#">CRC006</a>
[BSW00391] Parameter shall have unique names	<a href="#">CRC006</a>
[BSW00392] Parameters shall have a type	<a href="#">CRC006</a>
[BSW00393] Parameters shall have a range	<a href="#">CRC006</a>
[BSW00394] Specify the scope of the parameters	<a href="#">CRC006</a>
[BSW00395] List the required parameters (per parameter)	Not applicable (no use case)
[BSW00396] Configuration classes	<a href="#">CRC006</a>
[BSW00397] Pre-compile-time parameters	<a href="#">CRC006</a>
[BSW00398] Link-time parameters	Not applicable (no use case)
[BSW00399] Loadable Post-build time parameters	Not applicable (no use case)
[BSW00400] Selectable Post-build time parameters	Not applicable (no use case)
[BSW00401] Creating multiplicity	Not applicable (no use case)
[BSW00402] Published information	<a href="#">CRC005</a> , <a href="#">CRC004</a>
[BSW00375] Notification of wake-up reason	Not applicable (no use case)

<b>Requirement</b>	<b>Satisfied by</b>
[BSW101] Initialization interface	Not applicable (no use case)
[BSW00416] Sequence of Initialization	Not applicable (no use case)
[BSW00406] C-Initialization routine	Not applicable (no use case)
[BSW168] Diagnostic Interface of SW components	Not applicable (no use case)
[BSW00407] Function to read out published parameters	<a href="#">CRC011</a> , <a href="#">CRC012</a>
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (CRC is not part of service layer)
[BSW00424] BSW main processing function task allocation	Not applicable (no use case)
[BSW00425] Trigger conditions for schedulable objects	Not applicable (no use case)
[BSW00426] Exclusive areas in BSW modules	<a href="#">CRC008</a>
[BSW00427] ISR description for BSW modules	Not applicable (no use case)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (no use case)
[BSW00429] Restricted BSW OS functionality access	Not applicable (no use case)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (no use case)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (no use case)
[BSW00433] Calling of main processing functions	Not applicable (no use case)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (to be defined in Schedule module SWS)
[BSW00336] Shutdown interface	Not applicable (no use case)
[BSW00337] Classification of errors	Not applicable (no error reported by CRC module)
[BSW00338] Detection and Reporting of development errors	Not applicable (no error reported by CRC module)
[BSW00369] Do not return development error codes via API	Not applicable (no error reported by CRC module)
[BSW00339] Reporting of production relevant errors and exceptions	Not applicable (no error reported by CRC module)
[BSW00421] Reporting of production relevant error events	Not applicable (no error reported by CRC module)
[BSW00422] Debouncing of production relevant error status	Not applicable (no error reported by CRC module)
[BSW00420] Production relevant error event rate detection	Not applicable (no error reported by CRC module)
[BSW00417] Reporting of Error Events by Non-Basic Software	Not applicable (no error reported by CRC module)
[BSW00323] API parameter checking	Not applicable (no error reported by CRC module)
[BSW003] Version identification	<a href="#">CRC004</a>
[BSW004] Version check	<a href="#">CRC005</a> , <a href="#">CRC004</a>
[BSW00409] Header files for production code error IDs	Not applicable (no error reported by CRC module)
[BSW00385] Configuration of error notifications	Not applicable (no error reported by CRC module)

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00386] How errors shall be detected	Not applicable (no error reported by CRC module)
[BSW00318] Format of module version numbers	<a href="#">CRC004</a>
[BSW161] Microcontroller abstraction	Not applicable (Requirement on AUTOSAR architecture, not a single module)
[BSW162] ECU layout abstraction	Not applicable (Requirement on AUTOSAR architecture, not a single module)
[BSW324] Do not use HIS I/O Library	Not applicable (architecture decision)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (Requirement on AUTOSAR architecture, not a single module)
[BSW00415] User dependent include files	Not applicable (CRC doesn't provide restricted access for several modules)
[BSW164] Implementation of interrupt service routines	Not applicable (this module doesn't implement any ISR)
[BSW00325] Runtime of interrupt service routines	Not applicable (this module doesn't implement any ISR)
[BSW00326] Transition from ISRs to OS tasks	Not applicable (this module doesn't implement any ISR)
[BSW00342] Usage of source code and object code	Not applicable (Requirement on AUTOSAR architecture, not a single module)
[BSW00343] Specification and configuration of time	Not applicable (no configurable timings)
[BSW160] Human-readable configuration data	Not applicable (Requirement on documentation, not on specification)
[BSW007] HIS MISRA C	Not applicable (Requirement on implementation, not on specification)
[BSW00300] Module naming convention	Chapter 5.1
[BSW00413] Accessing instances of BSW modules	Conflict: This requirement will have impact on almost all BSW modules, therefore it can not be implemented within the Release 2.0 timeframe.
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (Requirement on implementation, not on specification)
[BSW00305] Self-defined data types naming convention	Not applicable (no data type definition in this specification)
[BSW00307] Global variables naming convention	Not applicable (Requirement on implementation, not on specification)
[BSW00310] API naming convention	Chapter 8.3
[BSW00373] Main processing function naming convention	Not applicable (no main function available)
[BSW00327] Error values naming convention	Not applicable (no error reported by CRC module)
[BSW00335] Status values naming convention	Not applicable (no status values available)
[BSW00350] Development error detection keyword	Not applicable (no error reported by CRC module)
[BSW00408] Configuration parameter naming convention	Chapter 10

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00410] Compiler switches shall have defined values	Not applicable (this module doesn't implement any compiler switch)
[BSW00411] Get version info keyword	<a href="#">CRC011</a> , Chapter 10
[BSW00346] Basic set of module files	Chapter 5.1
[BSW158] Separation of configuration from implementation	Chapter 5.1
[BSW00314] Separation of interrupt frames and service routines	Not applicable (this module doesn't implement any ISR)
[BSW00370] Separation of callback interface from API	Not applicable (this module doesn't implement any callback function)
[BSW00348] Standard type header	Not applicable (this module simply includes the standard type header via the module header file)
[BSW00353] Platform specific type header	Not applicable (this module simply includes the standard type header via the module header file)
[BSW00361] Compiler specific language extension header	Not applicable (this module simply includes the standard type header via the module header file)
[BSW00301] Limit imported information	Chapter 5.1
[BSW00302] Limit exported information	Not applicable (Requirement on the implementation, not on the specification)
[BSW00328] Avoid duplication of code	Not applicable (Requirement on the implementation, not on the specification)
[BSW00312] Shared code shall be reentrant	Not applicable (Requirement on the implementation, not on the specification)
[BSW006] Platform independency	Not applicable (Requirement on the implementation, not on the specification)
[BSW00357] Standard API return type	Chapters 8.2, 8.1
[BSW00377] Module specific API return types	Chapter 8.2
[BSW00304] AUTOSAR integer data types	Not applicable (Requirement on implementation, not for specification)
[BSW00355] Do not redefine AUTOSAR integer data types	Not applicable (Requirement on implementation, not for specification)
[BSW00378] AUTOSAR boolean type	Not applicable (Requirement on implementation, not for specification)
[BSW00306] Avoid direct use of compiler and platform specific keywords	Not applicable (Requirement on implementation, not for specification)
[BSW00308] Definition of global data	Not applicable (Requirement on implementation, not for specification)
[BSW00309] Global data with read-only constraint	Not applicable (Requirement on implementation, not for specification)
[BSW00371] Do not pass function pointers via API	Not applicable (no function pointers in this specification)
[BSW00358] Return type of init() functions	Not applicable (no init function in this specification)

<b>Requirement</b>	<b>Satisfied by</b>
[BSW00414] Parameter of init function	Not applicable (no init function in this specification)
[BSW00376] Return type and parameters of main processing functions <sup>7</sup>	Not applicable (no main processing function in this specification)
[BSW00359] Return type of callback functions	Not applicable (no callback function in this specification)
[BSW00360] Parameters of callback functions	Not applicable (no callback function in this specification)
[BSW00329] Avoidance of generic interfaces	Chapter 8.3
[BSW00330] Usage of macros / inline functions instead of functions	Not applicable Requirement on implementation, not for specification)
[BSW00331] Separation of error and status values	Not applicable (no error and status values)
[BSW009] Module User Documentation	Not applicable (Requirement on documentation, not on specification)
[BSW00401] Documentation of multiple instances of configuration parameters	Chapter 10
[BSW172] Compatibility and documentation of scheduling strategy	Not applicable (no scheduling in this specification)
[BSW010] Memory resource documentation	Not applicable (Requirement on documentation, not on specification)
[BSW00333] Documentation of callback function context	Not applicable (no callback function in this specification)
[BSW00374] Module vendor identification	<a href="#">CRC004</a>
[BSW00379] Module identification	<a href="#">CRC004</a>
[BSW003] Version identification	<a href="#">CRC004</a>
[BSW00318] Format of module version numbers	<a href="#">CRC004</a>
[BSW00321] Enumeration of module version numbers	Not applicable (Requirement on implementation, not for specification)
[BSW00341] Microcontroller compatibility documentation	Not applicable (Requirement on documentation, not on specification)
[BSW00334] Provision of XML file	Not applicable (Requirement on documentation, not on specification)
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	Chapter 5

Document: Requirements on CRC Routines

<b>Requirement</b>	<b>Satisfied by</b>
[BSW08518] CRC calculation method complexity	<a href="#">CRC001</a>
[BSW08520] CRC routine reentrancy	<a href="#">CRC008</a>
[BSW08521] CRC routine schedulability	<a href="#">CRC008</a>
[BSW08524] Error detection	<a href="#">CRC007</a>
[BSW08525] Support of standard polynomials	<a href="#">CRC002</a> , <a href="#">CRC003</a>
[BSW08526] CRC Standard calculation methods	<a href="#">CRC001</a>

## 7 Functional specification

### 7.1 Basic Concepts of CRC Codes

#### 7.1.1 Mathematical Description

Let D be a bitwise representation of data with a total number of n bit, i.e.

$$D = (d_{n-1}, d_{n-2}, d_{n-3}, \dots, d_1, d_0),$$

with  $d_0, d_1, \dots = 0b, 1b$ . The corresponding *Redundant Code* C is represented by n+k bit as

$$C = (D, R) = (d_{n-1}, d_{n-2}, d_{n-3}, \dots, d_2, d_1, d_0, r_{k-1}, \dots, r_2, r_1, r_0)$$

with  $r_0, r_1, \dots = 0b, 1b$  and  $R = (r_{k-1}, \dots, r_2, r_1, r_0)$  The code is simply a concatenation of the data and the redundant part. (For our application, we will chose  $k = 16, 32$  and n as a multiple of 16 resp. 32).

CRC-Algorithms are related to *polynomials* with coefficients in the finite *field of two element*, using arithmetic operations  $\oplus$  and  $*$  according to the following tables.

The  $\oplus$  operation is identified as the binary operation *exclusive-or*, that is usually available in the ALU of any CPU.

$\oplus$	0b	1b
0b	0b	1b
1b	1b	0b

*	0b	1b
0b	0b	0b
1b	0b	1b

For simplicity, we will write ab instead of a\*b

We introduce some examples for *polynomials* with coefficients in the *field of two elements* and give the simplified notation of it.

$$(ex. 1) \quad p_1(X) = 1b X^3 + 0b X^2 + 1b X^1 + 0b X^0 = X^3 + X$$

$$(ex. 2) \quad p_2(X) = 1b X^2 + 1b X^1 + 1b X^0 = X^2 + X^1 + 1b$$

Any code word, represented by n+k bit can be mapped to a polynomial of order n+k-1 with coefficients in the field of two elements. We use the intuitive mapping of the bits i.e.

$$C(X) = d_{n-1}X^{k+n-1} + d_{n-2}X^{k+n-2} + \dots + d_2X^{k+2} + d_1X^{k+1} + d_0 X^k + r_{k-1}X^{k-1} + r_{k-2}X^{k-2} + \dots r_1 X + r_0$$

$$C(X) = X^k(d_{n-1}X^{n-1} + d_{n-2}X^{n-2} + \dots + d_2X^2 + d_1X^1 + d_0) + r_{k-1}X^{k-1} + r_{k-2}X^{k-2} + \dots r_1 X + r_0$$

$$C(X) = X^k D(X) \oplus R(X)$$

This mapping is one-to-one.

A certain space  $CRC_G$  of *Cyclic Redundant Code Polynomials* is defined to be a multiple of a given *Generator Polynomial*  $G(X) = X^k + g_{k-1} X^{k-1} + g_{k-2} X^{k-2} + \dots + g_2 X^2 + g_1 X + g_0$ . By definition, for any code polynomial  $C(X)$  in  $CRC_G$  there is a polynomial  $M(X)$  with

$$C(X) = G(X) M(X).$$

For a fixed irreducible (i.e. prime-) polynomial  $G(X)$ , the mapping  $M(X) \rightarrow C(X)$  is one-to-one. Now, how are data of a given codeword verified? This is basically a division of polynomials, using the *Euclidian Algorithm*. In practice, we are not interested in  $M(X)$ , but in the *remainder* of the division,  $C(X) \bmod G(X)$ . For a correct code word  $C$ , this remainder has to be zero,  $C(X) \bmod G(X) = 0$ . If this is not the case – there is an error in the codeword. Given  $G(X)$  has some additional algebraic properties, one can determine the error-location and correct the codeword.

Calculating the code word from the data can also be done with the Euclidian Algorithm. For a given data polynomial  $D(x) = d_{n-1}X^{n-1} + d_{n-2}X^{n-2} + \dots + d_1X^1 + d_0$  and the corresponding code polynomial  $C(X)$  we have

$$C(X) = X^k D(X) \oplus R(X) = M(X) G(X)$$

Performing the operation “mod  $G(X)$ ” on both sides, one obtains

$$0 = C(X) \bmod G(X) = [X^k D(X)] \bmod G(X) \oplus R(X) \bmod G(X) \quad (*)$$

We denote that the order of the Polynomial  $R(X)$  is less than the order of  $G(X)$ , so the modulo division gives zero with remainder  $R(X)$ :

$$R(X) \bmod G(X) = R(X).$$

For polynomial  $R(X)$  with coefficients in the finite field with two elements we have the remarkable property  $R(X) + R(X) = 0$ . If we add  $R(X)$  on both sides of equation (\*) we obtain

$$R(X) = X^k D(X) \bmod G(X).$$

The important implication is that the redundant part of the requested code can be determined by using the Euclidian Algorithm for polynomials. At present, any CRC calculation method is a more or less sophisticated variation of this basic algorithm.

Up to this point, the propositions on CRC Codes are summarized as follows:

1. The construction principle of CRC Codes is based on polynomials with coefficients in the finite field of two elements. The  $\oplus$  operation of this field is identical to the binary operation “xor” (exclusive or)
2. There is a natural mapping of bit-sequences into this space of polynomials.



3. Both calculation and verification of the CRC code polynomial is based on division modulo a given generator polynomial.
4. This generator polynomial has to have certain algebraic properties in order to achieve error-detection and eventually error-correction.

### 7.1.2 Euclidian Algorithm for Binary Polynomials and Bit-Sequences

Given a Polynomial  $P_n(X) = p_n X^n + p_{n-1} X^{n-1} + \dots + p_2 X^2 + p_1 X + p_0$  with coefficients in the finite field of two elements. Let  $Q(X) = X^k + q_{k-1} X^{k-1} + q_{k-2} X^{k-2} + \dots + q_2 X^2 + q_1 X + q_0$  be another polynomial of exact order  $k > 0$ . Let  $R_n(X)$  be the remainder of the polynomial division of maximum order  $k-1$  and  $M_n(X)$  corresponding so that

$$R_n(X) \oplus M_n(X) Q(X) = P_n(X).$$

#### Euclidian Algorithm - Recursive

(Termination of recursion)

If  $n < k$ , then choose  $R_n(X) = P_n(X)$  and  $M_n = 0$ .

(Recursion  $n+1 \rightarrow n$ )

Let  $P_{n+1}(X)$  be of maximum order  $n+1$ .

If  $n+1 \geq k$  calculate  $P_n(X) = P_{n+1}(X) - p_{n+1} Q(X) X^{n-k+1}$ . This polynomial is of maximum order  $n$ . Then

$$P_{n+1}(X) \bmod Q(X) = P_n(X) \bmod Q(X).$$

#### Proof of recursion

Choose  $R_{n+1}(X) = P_{n+1}(X) \bmod Q(X)$  and  $M_{n+1}(X)$  so that

$$R_{n+1}(X) \oplus M_{n+1}(X) Q(X) = P_{n+1}(X).$$

Then  $R_{n+1}(X) - R_n(X) = P_{n+1}(X) - M_{n+1}(X) Q(X) - P_n(X) \oplus M_n(X) Q(X)$ .

With  $P_{n+1}(X) - P_n(X) = p_{n+1} Q(X) X^{n-k+1}$  we obtain

$$R_{n+1}(X) - R_n(X) = p_{n+1} Q(X) X^{n-k+1} + M_n(X) Q(X) - M_{n+1}(X) Q(X)$$

$$R_{n+1}(X) - R_n(X) = Q(X) [p_{n+1} X^{n-k+1} + M_n(X) - M_{n+1}(X)]$$

On the left side, there is a polynomial of maximum order  $k-1$ . On the right side  $Q(X)$  is of exact order  $k$ . This implies that both sides are trivial and equal to zero. One obtains

$$R_{n+1}(X) = R_n(X) \tag{1}$$

$$M_{n+1}(X) = M_n(X) + p_{n+1} X^{n-k+1} \tag{2}$$

(end of proof)

## Example

$$P(X) = P_4(X) = X^4 + X^2 + X + 1b; \quad Q(X) = X^2 + X + 1b; \quad n = 4; \quad k = 2$$

$$P_3(X) = X^4 + X^2 + X + 1b - 1b(X^2 + X + 1b) X^2 = X^3 + X + 1b.$$

$$P_2(X) = X^3 + X + 1b - 1b X (X^2 + X + 1b) = X^2 + 1b.$$

$$P_1(X) = X^2 + 1 - 1b (X^2 + X + 1) = X$$

$$R(X) = P(X) \bmod Q(X) = R_1(X) = P_1(X) = X.$$

### 7.1.3 CRC calculation, Variations and Parameter

Based on the Euclidian Algorithm, some variations have been developed in order to improve the calculation performance. All these variations do not improve the capability to detect or correct errors – the so-called Hamming Distance of the resulting code is determined only by the generator polynomial. Variations simply optimize for different implementing ALUs.

CRC-Calculation methods are characterized as follows:

1. Rule for Mapping of Data to a bit sequence ( $d_{n-1}, d_{n-2}, d_{n-3}, \dots, d_1, d_0$ ) and the corresponding data polynomial  $D(X)$  (standard or reflected data).
2. Generator polynomial  $G(X)$
3. Start value and corresponding Polynomial  $S(X)$
4. Appendix  $A(X)$ , also called XOR-value for modifying the final result.
5. Rule for mapping the resulting CRC-remainder  $R(X)$  to codeword. (Standard or reflected data)

The calculation itself is organized in the following steps

- Map Data to  $D(X)$
- Perform Euclidian Algorithm on  $X^k D(X) + X^{n-k-1} S(X) + A(X)$  and determine  $R(X) = [X^k D(X) + X^{n-k-1} S(X) + A(X)] \bmod G(X)$
- Map  $D(X), R(X)$  to codeword

## 7.2 Standard parameters

This section gives a rough overview on the standard parameters which are commonly used for 8-bit, 16-bit and 32-bit CRC calculation.

- CRC result width: Defines the result data width of the CRC calculation.
- Polynomial: Defines the generator polynomial which is used for the CRC algorithm.
- Initial value: Defines the start condition for the CRC algorithm.
- Input data reflected: Defines whether the bits of each input byte are reflected before being processed.

- Result data reflected: Similar to “Input data reflected” this parameter defines whether the bits of the CRC result are reflected.
- XOR value: This Value is XORed to the final register value before the value is returned as the official checksum.
- Check: This field is a check value that can be used as a weak validator of implementations of the algorithm. The field contains the checksum obtained when the ASCII string "123456789" is fed through the specified algorithm.

### 7.2.1 8-bit CRC calculation

**CRC025:** The CRC module shall implement the CRC8 routine based on the SAE-J1850 CRC8 Standard:

CRC result width:	8 bits
Polynomial:	1Dh
Initial value:	0h
Input data reflected:	No
Result data reflected:	No
XOR value:	00h
Check:	4Bh

### 7.2.2 16-bit CRC calculation

**CRC002:** The CRC module shall implement the CRC16 routine based on the CCITT CRC16 Standard:

CRC result width:	16 bits
Polynomial:	1021h
Initial value:	FFFFh
Input data reflected:	No
Result data reflected:	No
XOR value:	0000h
Check:	29B1h

### 7.2.3 32-bit CRC calculation

**CRC003:** The CRC module shall implement the CRC32 routine based on the IEEE-802.3 CRC32 Ethernet Standard:

CRC result width:	32 bits
Polynomial:	04C11DB7h
Initial value:	FFFFFFFFh
Input data reflected:	Yes
Result data reflected:	Yes
XOR value:	FFFFFFFFh
Check:	CBF43926h

### 7.3 General behavior

Data blocks are passed to the CRC routines using the parameters “start address”, “size” and “start value”. The return value is the CRC result.

### 7.4 Error classification

The CRC library functions do not provide any error classification. CRC recalculation and comparison must be done by each module in the upper layer (e.g. NVRAM Manager).

### 7.5 Error detection

The CRC library functions do not provide any error detection mechanism.

### 7.6 Error notification

The CRC library functions do not provide any error notification.

### 7.7 Version check

**CRC005:** `Crc.c` shall check if the correct version of `Crc.h` is included. This shall be done by a preprocessor check of the version number `CRC_MAJOR_VERSION` and `CRC_MINOR_VERSION`.

## 8 API specification

### 8.1 Imported types

In this chapter all types included from the following files are listed:

#### CRC018:

Header file	Imported Type
Std_Types.h	Std_VersionInfoType

### 8.2 Type definitions

None.

### 8.3 Function definitions

**CRC013:** If CRC routines are to be used as a library, the CRC modules' implementer shall develop the CRC module in a way that only those parts of the CRC code that are used by other modules are linked into the final binary.

**CRC014:** When calculating CRC over multiple blocks, the first call should use CRC\_INITIAL\_VALUEXX as start value and then for subsequent calls, the start value shall be the result of the previous call.

#### 8.3.1 8-bit CRC Calculation

#### CRC026:

<b>Service name:</b>	Crc_CalculateCRC8	
<b>Syntax:</b>	<pre>uint8 Crc_CalculateCRC8(     const uint8* Crc_DataPtr,     uint32 Crc_Length,     uint8 Crc_StartValue8 )</pre>	
<b>Service ID[hex]:</b>	0x01	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	Crc_DataPtr	Pointer to start address of data block to be calculated.
	Crc_Length	Length of data block to be calculated in bytes.
	Crc_StartValue8	Initial value when the algorithm starts.
<b>Parameters (in-out):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	uint8	8 bit result of CRC calculation.
<b>Description:</b>	This service makes a CRC8 calculation on Crc_Length data bytes.	

**CRC027:** The function `Crc_CalculateCRC8` shall perform a CRC8 calculation on `Crc_Length` data bytes, pointed to by `Crc_DataPtr`, with the starting value of `Crc_StartValue8`.

**CRC028:** If the CRC calculation within the function `Crc_CalculateCRC8` is performed by hardware, then the CRC module's implementer shall ensure reentrancy of this function by implementing a (software based) locking mechanism.

Note: If large data blocks have to be calculated (>32 bytes, depending on performance of processor platform), the table based calculation method should be configured for the function `Crc_CalculateCRC8` in order to decrease the calculation time.

The function `Crc_CalculateCRC8` requires specification of configuration parameters defined in [CRC006](#).

### 8.3.2 16-bit CRC Calculation

#### CRC019:

<b>Service name:</b>	Crc_CalculateCRC16	
<b>Syntax:</b>	<pre>uint16 Crc_CalculateCRC16(     const uint8* Crc_DataPtr,     uint32 Crc_Length,     uint16 Crc_StartValue16 )</pre>	
<b>Service ID[hex]:</b>	0x02	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	<code>Crc_DataPtr</code>	Pointer to start address of data block to be calculated.
	<code>Crc_Length</code>	Length of data block to be calculated in bytes.
	<code>Crc_StartValue16</code>	Initial value when the algorithm starts.
<b>Parameters (in-out):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	uint16	16 bit result of CRC calculation.
<b>Description:</b>	This service makes a CRC16 calculation on <code>Crc_Length</code> data bytes.	

**CRC015:** The function `Crc_CalculateCRC16` shall perform a CRC16 calculation on `Crc_Length` data bytes, pointed to by `Crc_DataPtr`, with the starting value of `Crc_StartValue16`.

**CRC009:** If the CRC calculation within the function `Crc_CalculateCRC16` is performed by hardware, then the CRC module's implementer shall ensure reentrancy of this function by implementing a (software based) locking mechanism.

Note: If large data blocks have to be calculated (>32 bytes, depending on performance of processor platform), the table based calculation method should be configured for the function `Crc_CalculateCRC16` in order to decrease the calculation time.

The function `Crc_CalculateCRC16` requires specification of configuration parameters defined in [CRC006](#).

### 8.3.3 32-bit CRC Calculation

#### CRC020:

<b>Service name:</b>	Crc_CalculateCRC32	
<b>Syntax:</b>	<pre>uint32 Crc_CalculateCRC32(     const uint8* Crc_DataPtr,     uint32 Crc_Length,     uint32 Crc_StartValue32 )</pre>	
<b>Service ID[hex]:</b>	0x03	
<b>Sync/Async:</b>	Synchronous	
<b>Reentrancy:</b>	Reentrant	
<b>Parameters (in):</b>	Crc_DataPtr	Pointer to start address of data block to be calculated.
	Crc_Length	Length of data block to be calculated in bytes.
	Crc_StartValue32	Initial value when the algorithm starts.
<b>Parameters (in-out):</b>	None	
<b>Parameters (out):</b>	None	
<b>Return value:</b>	uint32	32 bit result of CRC calculation.
<b>Description:</b>	This service makes a CRC32 calculation on Crc_Length data bytes.	

**CRC016:** The function `Crc_CalculateCRC32` shall perform a CRC32 calculation on `Crc_Length` data bytes, pointed to by `Crc_DataPtr`, with the starting value of `Crc_StartValue32`.

**CRC010:** If the CRC calculation within the function `Crc_CalculateCRC32` is performed by hardware, then the CRC module's implementer shall ensure reentrancy of this function by implementing a (software based) locking mechanism.

Note: If large data blocks have to be calculated (>32 bytes, depending on performance of processor platform), the table based calculation method should be configured for the function `Crc_CalculateCRC32` in order to decrease the calculation time.

The function `Crc_CalculateCRC32` requires specification of configuration parameters defined in [CRC006](#).

### 8.3.4 Crc\_GetVersionInfo

#### CRC021:

<b>Service name:</b>	Crc_GetVersionInfo	
<b>Syntax:</b>	<pre>void Crc_GetVersionInfo(     Std_VersionInfoType VersionInfo )</pre>	
<b>Service ID[hex]:</b>	0x04	
<b>Sync/Async:</b>	Synchronous	



<b>Reentrancy:</b>	Non Reentrant
<b>Parameters (in):</b>	None
<b>Parameters (in-out):</b>	None
<b>Parameters (out):</b>	Versioninfo   Pointer to where to store the version information of this module.
<b>Return value:</b>	None
<b>Description:</b>	This service returns the version information of this module.

**CRC011:** The function `Crc_GetVersionInfo` shall return the version information of the CRC module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

**CRC012:** The function `Crc_GetVersionInfo` shall be pre-compile time configurable On/Off by the configuration parameter `CRC_VERSION_INFO_API`.

**CRC017:** If source code for caller and callee of the function `Crc_GetVersionInfo` is available, the CRC module should realize this function as a macro, defined in the modules header file.

## 8.4 Call-back notifications

None.

## 8.5 Scheduled functions

This chapter lists all functions called directly by the Basic Software Module Scheduler.

The `Crc` module does not have scheduled functions.

## 8.6 Expected Interfaces

In this chapter, all interfaces required from other modules are listed.

### 8.6.1 Mandatory Interfaces

None

### 8.6.2 Optional Interfaces

None.

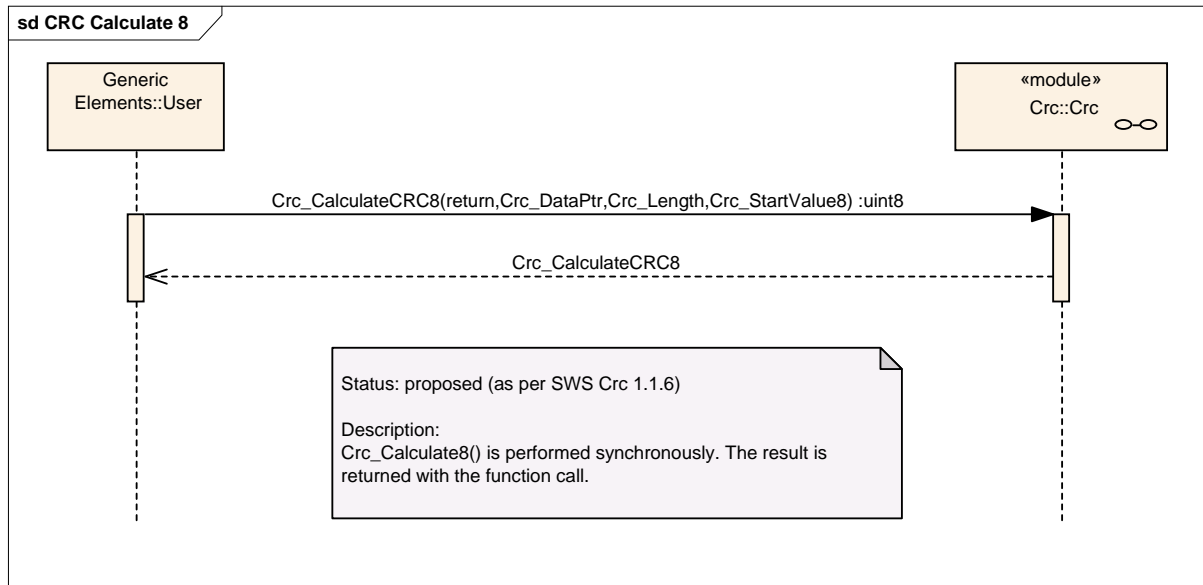
### 8.6.3 Configurable interfaces

None.

## 9 Sequence diagrams

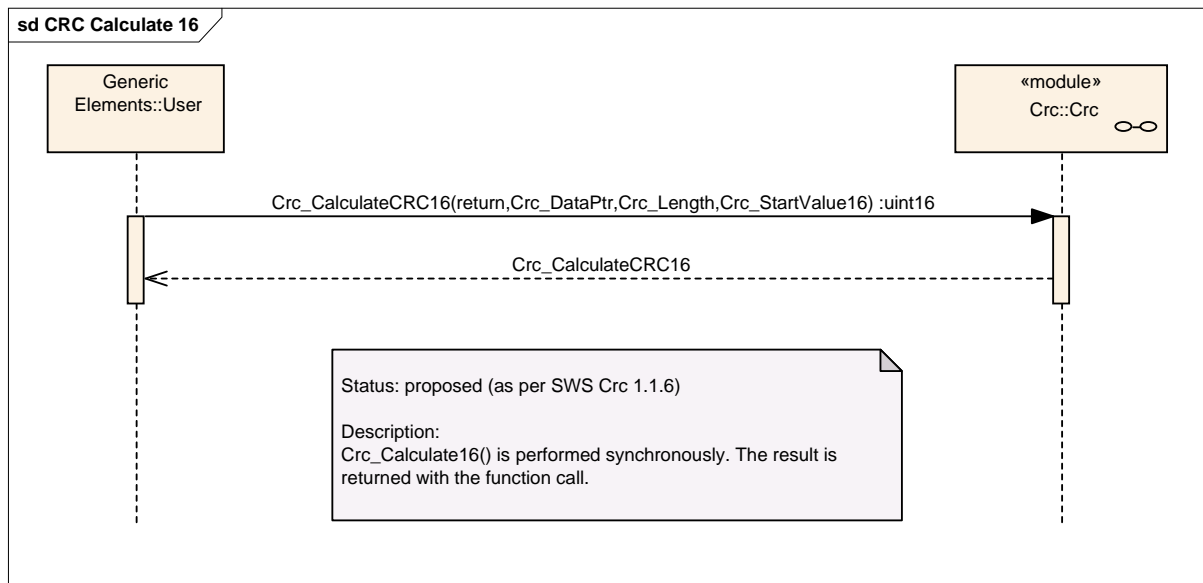
### 9.1 Crc\_CalculateCRC8 ()

The following diagram shows the synchronous function call `Crc_CalculateCRC8`.



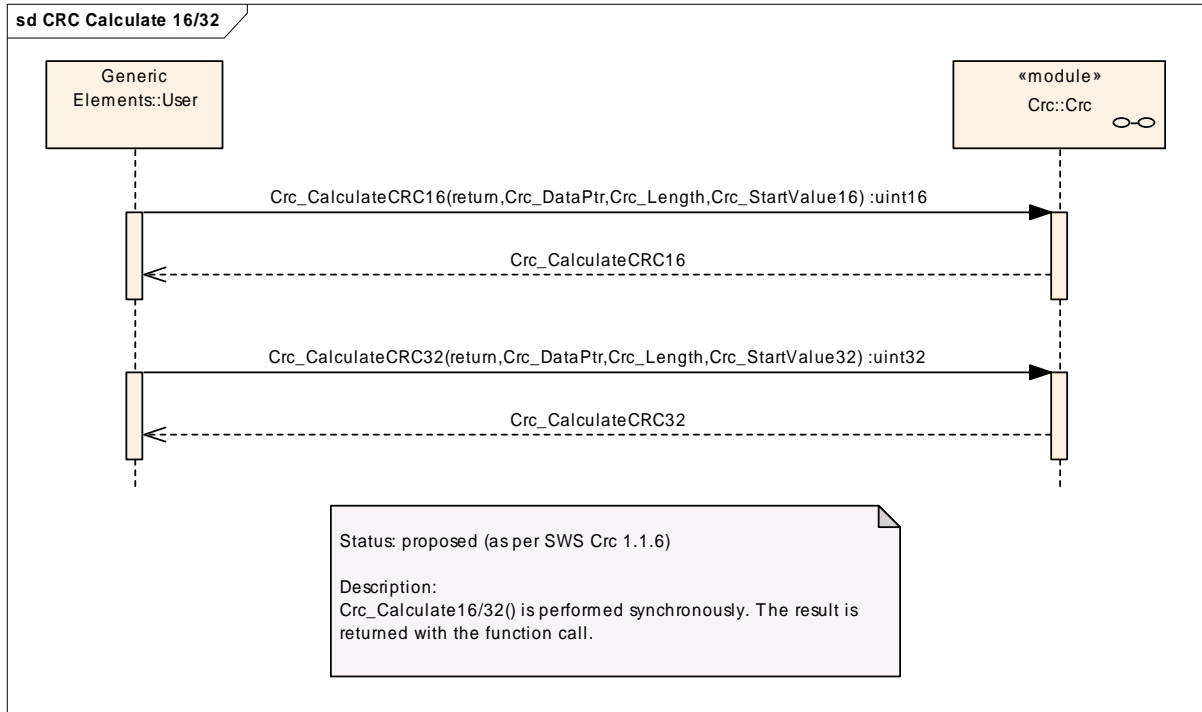
### 9.2 Crc\_CalculateCRC16()

The following diagram shows the synchronous function call `Crc_CalculateCRC16`.



### 9.3 Crc\_CalculateCRC32()

The following diagram shows the synchronous function call `Crc_CalculateCRC32`.



## 10 Configuration specification

### 10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Architecture [2]
- AUTOSAR ECU Configuration Specification [5]:  
This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

#### 10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

#### 10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

#### 10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

## 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in Chapters 7 and Chapter 8.

### 10.2.1 Variants

No variants specified.

### 10.2.2 Crc

<b>Module Name</b>	<b>Crc</b>
<b>Module Description</b>	Configuration of the Crc (Crc routines) module.

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CrcGeneral	1	General configuration of CRC module
CrcPublishedInformation	1	Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, since they are published information.

### 10.2.3 CrcGeneral

<b>SWS Item</b>	<b>CRC006 :</b>
<b>Container Name</b>	CrcGeneral{CRC_COMMON}
<b>Description</b>	General configuration of CRC module
<b>Configuration Parameters</b>	

<b>SWS Item</b>	--									
<b>Name</b> <small>last modified in: V 1.0.0</small>	Crc8Mode {CRC_8_MODE}									
<b>Description</b>	Switch to select one of the available CRC8 calculation methods.									
<b>Multiplicity</b>	1									
<b>Type</b>	EnumerationParamDef									
<b>Range</b>	<table border="1"> <tr> <td>Crc8Runtime</td> <td>runtime based CRC8 calculation</td> </tr> <tr> <td>Crc8Hardware</td> <td>hardware based CRC8 calculation</td> </tr> <tr> <td>Crc8Table</td> <td>table based CRC8 calculation (default selection)</td> </tr> </table>	Crc8Runtime	runtime based CRC8 calculation	Crc8Hardware	hardware based CRC8 calculation	Crc8Table	table based CRC8 calculation (default selection)			
Crc8Runtime	runtime based CRC8 calculation									
Crc8Hardware	hardware based CRC8 calculation									
Crc8Table	table based CRC8 calculation (default selection)									
<b>ConfigurationClass</b>	<table border="1"> <tr> <td><b>Pre-compile time</b></td> <td>X</td> <td>All Variants</td> </tr> <tr> <td><b>Link time</b></td> <td>--</td> <td></td> </tr> <tr> <td><b>Post-build time</b></td> <td>--</td> <td></td> </tr> </table>	<b>Pre-compile time</b>	X	All Variants	<b>Link time</b>	--		<b>Post-build time</b>	--	
<b>Pre-compile time</b>	X	All Variants								
<b>Link time</b>	--									
<b>Post-build time</b>	--									
<b>Scope / Dependency</b>										

<b>SWS Item</b>	<b>CRC025 :</b>						
<b>Name</b>	Crc16Mode {CRC_16_MODE}						
<b>Description</b>	Switch to select one of the available CRC16 calculation methods.						
<b>Multiplicity</b>	1						
<b>Type</b>	EnumerationParamDef						
<b>Range</b>	<table border="1"> <tr> <td>CRC_16_HARDWARE</td> <td>hardware based CRC16 calculation</td> </tr> <tr> <td>CRC_16_TABLE</td> <td>table based CRC16 calculation (default selection)</td> </tr> <tr> <td>CRC_16_RUNTIME</td> <td>runtime based CRC16 calculation</td> </tr> </table>	CRC_16_HARDWARE	hardware based CRC16 calculation	CRC_16_TABLE	table based CRC16 calculation (default selection)	CRC_16_RUNTIME	runtime based CRC16 calculation
CRC_16_HARDWARE	hardware based CRC16 calculation						
CRC_16_TABLE	table based CRC16 calculation (default selection)						
CRC_16_RUNTIME	runtime based CRC16 calculation						
<b>ConfigurationClass</b>	<table border="1"> <tr> <td><b>Pre-compile time</b></td> <td>X</td> <td>All Variants</td> </tr> </table>	<b>Pre-compile time</b>	X	All Variants			
<b>Pre-compile time</b>	X	All Variants					

	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	<b>CRC026 :</b>		
<b>Name</b>	Crc32Mode {CRC_32_MODE}		
<b>Description</b>	Switch to select one of the available CRC32 calculation methods.		
<b>Multiplicity</b>	1		
<b>Type</b>	EnumerationParamDef		
<b>Range</b>	CRC_32_RUNTIME	runtime based CRC32 calculation	
	CRC_32_TABLE	table based CRC32 calculation (default selection)	
	CRC_32_HARDWARE	hardware based CRC32 calculation	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

<b>SWS Item</b>	<b>CRC027 :</b>		
<b>Name</b>	CrcVersionInfoApi {CRC_VERSION_INFO_API}		
<b>Description</b>	Pre-processor switch to enable / disable the API to read out the modules version information. true: Version info API enabled. false: Version info API disabled.		
<b>Multiplicity</b>	1		
<b>Type</b>	BooleanParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

**No Included Containers**

## 10.3 Published Information

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.

The standard common published information like

- vendorId (CRC\_VENDOR\_ID),
- moduleId (CRC\_MODULE\_ID),
- arMajorVersion (CRC\_AR\_MAJOR\_VERSION),
- arMinorVersion (CRC\_AR\_MINOR\_VERSION),
- arPatchVersion (CRC\_AR\_PATCH\_VERSION),
- swMajorVersion (CRC\_SW\_MAJOR\_VERSION),
- swMinorVersion (CRC\_SW\_MINOR\_VERSION),
- swPatchVersion (CRC\_SW\_PATCH\_VERSION),
- vendorApiInfix (CRC\_VENDOR\_API\_INFIX)
- 

is provided in the BSW Module Description Template (see [7], Figure 4.1 and Figure 7.1). Additional published parameters are listed below if applicable for this module.

### 10.3.1 CrcPublishedInformation

<b>SWS Item</b>	<b>CRC004 :</b>
<b>Container Name</b>	CrcPublishedInformation
<b>Description</b>	Additional published parameters not covered by CommonPublishedInformation container. Note that these parameters do not have any configuration class setting, since they are published information.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	--	
<b>Name</b> <small>last modified in: V 2.1.0</small>	CrclnitialValue8 {CRC_INITIAL_VALUE8}	
<b>Description</b>	Initial value for the 8-bit CRC calculation.	
<b>Multiplicity</b>	1	
<b>Type</b>	IntegerParamDef	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--
	<b>Link time</b>	--
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>		

<b>SWS Item</b>	<b>CRC028 :</b>	
<b>Name</b>	CrclnitialValue16 {CRC_INITIAL_VALUE16}	
<b>Description</b>	Initial value for the 16-bit CRC calculation.	
<b>Multiplicity</b>	1	
<b>Type</b>	IntegerParamDef	
<b>Default value</b>	--	
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--
	<b>Link time</b>	--
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>		



<b>SWS Item</b>	<b>CRC029 :</b>		
<b>Name</b>	CrclInitialValue32 {CRC_INITIAL_VALUE32}		
<b>Description</b>	Initial value for the 32-bit CRC calculation.		
<b>Multiplicity</b>	1		
<b>Type</b>	IntegerParamDef		
<b>Default value</b>	--		
<b>ConfigurationClass</b>	<b>Pre-compile time</b>	--	
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>			

**No Included Containers**

## 11 Changes to Release 1

### 11.1 Deleted SWS Items

No deleted SWS items to Release 1.

### 11.2 Replaced SWS Items

No replaced SWS items to Release 1.

### 11.3 Changed SWS Items

No changed SWS items to Release 1.

### 11.4 Added SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">CRC009</a>	Bugfix #5825
<a href="#">CRC010</a>	Bugfix #5825
<a href="#">CRC011</a>	Added due to adaptation to new SWS template.
<a href="#">CRC012</a>	Added due to adaptation to new SWS template.
<a href="#">CRC013</a>	Bugfix #8839
<a href="#">CRC014</a>	RfC #13416
<a href="#">CRC015</a>	RfC #13416
<a href="#">CRC016</a>	RfC #13416
<a href="#">CRC016</a>	RfC #13416
<a href="#">CRC016</a>	RfC #13416
<a href="#">CRC016</a>	RfC #13416
<a href="#">CRC016</a>	RfC #13416

## 12 Changes during SWS Improvements by Technical Office

### 12.1 Deleted SWS Items

<i>SWS Item</i>	<i>Rationale</i>
CRC001	SWS Improvement: No requirement on CRC module but generic information.
CRC007	SWS Improvement: No requirement on CRC module but generic information.
CRC008	SWS Improvement: No requirement.

### 12.2 Replaced SWS Items

No replaced SWS items.

### 12.3 Changed SWS Items

No changed SWS items.

### 12.4 Added SWS Items

<i>SWS Item</i>	<i>Rationale</i>
<a href="#">CRC017</a>	Hint for Crc_GetVersionInfo
<a href="#">CRC018</a>	UML Model linking of imported types
<a href="#">CRC019</a>	UML Model linking of Crc_CalculateCRC16
<a href="#">CRC020</a>	UML Model linking of Crc_CalculateCRC32
<a href="#">CRC021</a>	UML Model linking of Crc_GetVersionInfo
<a href="#">CRC022</a>	Gave ID to existing text
<a href="#">CRC023</a>	Gave ID to existing text
<a href="#">CRC024</a>	Gave ID to existing text
<a href="#">CRC025</a>	Add support for CRC8
<a href="#">CRC026</a>	Add support for CRC8
<a href="#">CRC027</a>	Add support for CRC8
<a href="#">CRC028</a>	Add support for CRC8

## 13 Changes to Release 2

### 13.1 Deleted SWS Items

No deleted SWS items.

### 13.2 Replaced SWS Items

No replaced SWS items.

### 13.3 Changed SWS Items

No changed SWS items.

### 13.4 Added SWS Items

<b>SWS Item</b>	<b>Rationale</b>
<a href="#">CRC025</a>	Add support for CRC8
<a href="#">CRC026</a>	Add support for CRC8
<a href="#">CRC027</a>	Add support for CRC8
<a href="#">CRC028</a>	Add support for CRC8