MICROSAR CRC

Technical Reference

Version 4.00.00

Authors	Claudia Mausz
Status	Released

1 Document Information

1.1 History

Author	Date	Version	Remarks
Tobias Schmid	13-12-2006	1.0	Initial Version
Tobias Schmid	21-01-2008	3.00.00	Update to ASR 2.1 Changed versioning to new notation
Claudia Mausz	19-05-2008	4.00.00	Update to ASR3.0.0 Add Crc8 calculation

Table 1-1 History of the document

1.2 Reference Documents

No.	Title	Version
[1]	AUTOSAR_SWS_CRC_Routines.pdf	V3.0 Rev 0002
[2]	AUTOSAR_BasicSoftwareModules.pdf	V1.2.0

Table 1-2 Reference documents

1.1 Scope of the Document

This technical reference describes the general use of the CRC driver basis software. There are no aspects which are controller specific.

Please note
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector's release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

1 Document Information 2
1.1 History 2
1.2 Reference Documents 2
1.1 Scope of the Document 2

2 Component History 6

3 Introduction 7
3.1 Architecture Overview 8

4 Functional Description 9
4.1 Features 9
4.2 Initialization 9
4.3 States 9
4.4 Main Functions 9
4.5 Error Handling 10
4.5.1 Development Error Reporting 10
4.5.1.1 Parameter Checking 10
4.5.2 Production Code Error Reporting 10
4.5.2.1 Parameter Checking 10

5 Integration 11
5.1 Scope of Delivery 11
5.1.1 Static Files 11
5.1.2 Dynamic Files 11
5.2 Include Structure 11
5.3 Compiler Abstraction and Memory Mapping 12

6 API Description 13
6.1 Type Definitions 13
6.2 Interrupt Service Routines provided by CRC 13
6.3 Services provided by CRC 13
6.3.1 Crc_CalculateCRC8 14
6.3.2 Crc_CalculateCRC16 15
6.3.3 Crc_CalculateCRC32 16
6.3.4 Crc_GetVersionInfo 17
6.4 Services used by CRC 17
6.5 Callback Functions 17

6.6 Configurable Interfaces ... 17
6.6.1 Notifications ... 17
6.6.2 Callout Functions ... 17
6.7 Service Ports ... 17

7 Configuration ... 19
7.1 Configuration with EAD ... 19
7.1.1 Start configuration of the CRC ... 19
7.1.2 CRC configuration tab ... 19
7.1.2.1 General Settings ... 19
7.1.2.2 Module API ... 20

8 AUTOSAR Standard Compliance ... 22
8.1 Deviations ... 22
8.2 Additions/ Extensions ... 22
8.3 Limitations ... 22

9 Glossary and Abbreviations ... 23
9.1 Glossary ... 23
9.2 Abbreviations ... 23

10 Contact ... 24

Illustrations

Figure 3-1 AUTOSAR architecture 8
Figure 5-1 Include structure 11
Figure 7-1 EAD Configuration – General Settings 19
Figure 7-2 EAD Configuration – Module API 20

Tables

Table 1-1 History of the document 2
Table 1-2 Reference documents 2
Table 2-1 Component history 6
Table 4-1 Supported SWS features 9
Table 4-2 Not supported SWS features 9
Table 5-1 Static files 11
Table 5-2 Generated files 11
Table 5-3 Compiler abstraction and memory mapping 12
Table 6-1 Type definitions 13
Table 6-2 Std_VersionInfoType 13
Table 6-3 Crc_CalculateCRC8 14
Table 6-4 Crc_CalculateCRC16 15
Table 6-5 Crc_CalculateCRC32 16
Table 6-6 Crc_GetVersionInfo 17
Table 7-1 EAD Configuration – General Settings 20
Table 7-2 EAD Configuration – Module API 21
Table 9-1 Glossary 23
Table 9-2 Abbreviations 23

2 Component History

The component history gives an overview over the important milestones, that are supported in the different versions of the component.

Component Version	New Features
04.00.00	Update to AUTOSAR Version 3.0 Add Crc8 calculation

Table 2-1 Component history

3 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module CRC as specified in [1].

Supported AUTOSAR Release:	3.0	
Supported Configuration Variants:	pre-compile	
Vendor ID:	CRC_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	CRC_MODULE_ID	201 decimal (according to ref. [2])

This component implements service functions in ANSI C for calculating CRC checksums. The module allows pre-compile configuration of the calculation method, which is used to compute the CRC values. The three possible methods are table based or runtime calculation of CRC values.

There are three different CRC calculation services available:

- Checksum calculation of 8bit CRC value from a buffer,
- Checksum calculation of 16bit CRC value from a buffer,
- Checksum calculation of 32bit CRC value from a buffer.

3.1 Architecture Overview

The following figure shows where the CRC is located in the AUTOSAR architecture.

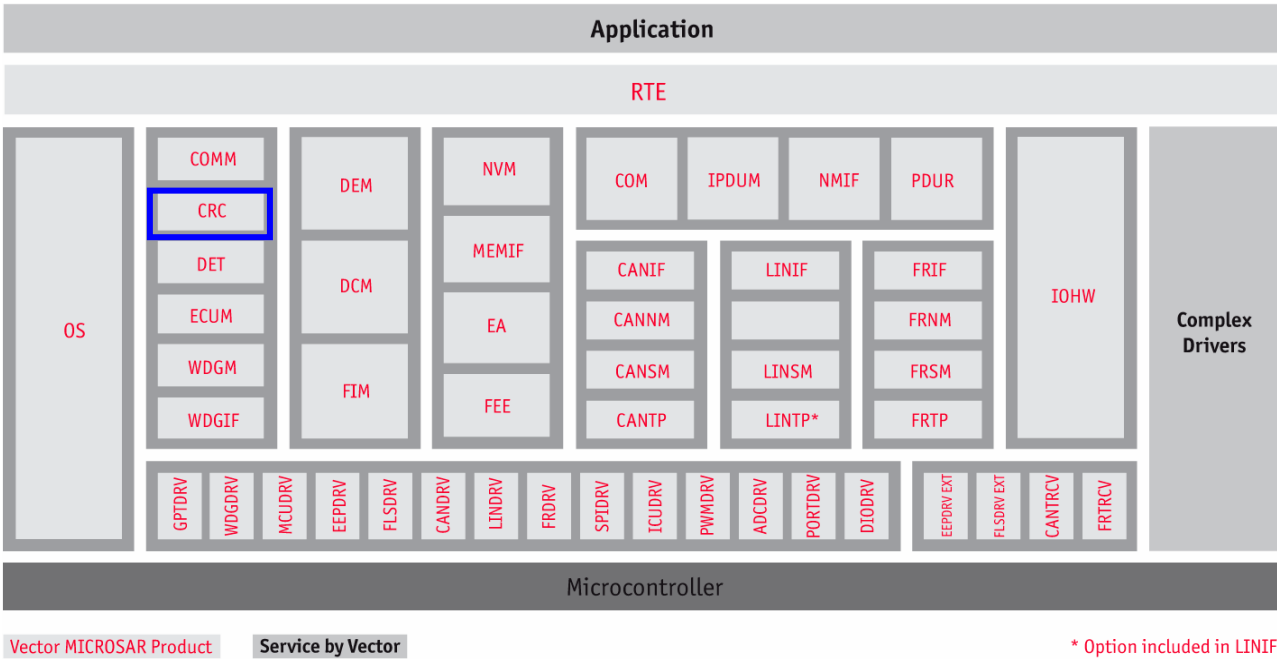

Figure 3-1 AUTOSAR architecture

No interface from other module is required. The CRC component is a service module. It can be called from any module or application (e.g. NVRAM Manager).

4 Functional Description

4.1 Features

The features listed in this chapter cover the complete functionality specified in [1].

The "supported" and "not supported" features are presented in the following two tables. For further information of not supported features also see chapter 8.

The following features described in [1] are supported:

Supported Feature
Crc8 calculation based on the SAE-J1850 CRC8 Standard
Crc16 calculation based on the CCITT Standard
Crc32 calculation based on the IEEE-802.3 Ethernet Standard
For all CRC calculations following methods can be used: ■ Table based calculation: results in fast execution, but increases code size (ROM table) ■ Runtime calculation: results in smaller code size, but slower execution time
All routines are re-entrant and can be used by multiple applications at the same time.

Table 4-1 Supported SWS features

The following features described in [1] are not supported:

Not Supported Feature
None

Table 4-2 Not supported SWS features

4.2 Initialization

No initialization is necessary.

4.3 States

This component contains no state machine.

4.4 Main Functions

This component has no main function. It contains only the routines to calculate the checksums. These API functions are called from the NVRAM Manager.

4.5 Error Handling

4.5.1 Development Error Reporting

Module CRC does not provide any development error detection mechanism.

Info
CRC recalculation and comparison with original CRC values must be done by the user itself (e.g. NVRAM Manager).

4.5.1.1 Parameter Checking

Module CRC's functions do not perform any checks.

4.5.2 Production Code Error Reporting

Module CRC does not provide any production error detection mechanism.

4.5.2.1 Parameter Checking

Module CRC's functions do not perform any checks.

5 Integration

This chapter gives necessary information for the integration of the MICROSAR CRC into an application environment of an ECU.

5.1 Scope of Delivery

The delivery of the CRC contains the files which are described in the chapters 5.1.1 and 5.1.2:

5.1.1 Static Files

File Name	Description
Crc.c	Implementation of the CRC routines.
Crc.h	Header files of the CRC routines.

Table 5-1 Static files

5.1.2 Dynamic Files

The dynamic files are generated by the configuration tool EAD.

File Name	Description
Crc_Cfg.h	Configuration of the calculation method.

Table 5-2 Generated files

5.2 Include Structure

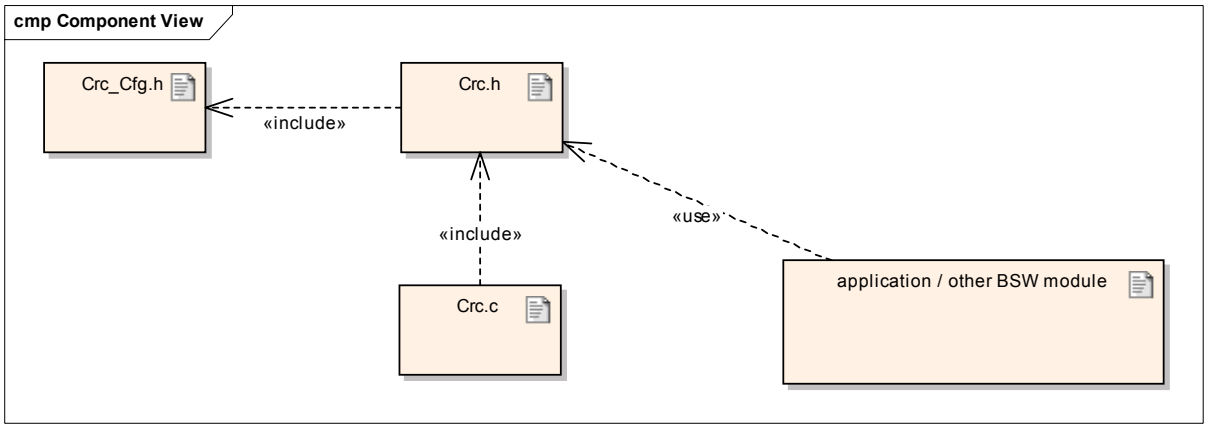

Figure 5-1 Include structure

5.3 Compiler Abstraction and Memory Mapping

The objects (e.g. variables, functions, constants) are declared by compiler independent definitions – the compiler abstraction definitions. Each compiler abstraction definition is assigned to a memory section.

The following table contains the memory section names and the compiler abstraction definitions, defined for the CRC and illustrates their assignment among each other.

Compiler Abstraction Definitions **Memory Mapping Sections**	CRC_CODE	CRC_PUBLIC_CONST	CRC_CONST
CRC_START_SEC_CONST_UNSPECIFIED		■	
CRC_START_SEC_CONST_8BIT			■
CRC_START_SEC_CONST_16BIT			■
CRC_START_SEC_CONST_32BIT			■
CRC_START_SEC_CODE	■		

Table 5-3 Compiler abstraction and memory mapping

6 API Description

6.1 Type Definitions

The types defined by the CRC are described in this chapter.

Type Name	C-Type	Description	Value Range
Crc_DataRefType	uint8	Pointer type for pointers to data buffers, from which the CRC value is to be calculated.	`0-254 (uint8)`

Table 6-1 Type definitions

Std_VersionInfoType

Struct Element Name	C-Type	Description	Value Range
Std_VersionInfoType	struct	Returns the version information.	`uint16 vendorID`
			`uint8 moduleID`
			`uint8 instanceID`
			`uint8 sw_major_version`
			`uint8 sw_minor_version`
			`uint8 sw_patch_version`

Table 6-2 Std_VersionInfoType

6.2 Interrupt Service Routines provided by CRC

The CRC component has no interrupt service routine.

6.3 Services provided by CRC

The CRC API consists of services, which are realized by function calls.

6.3.1 Crc_CalculateCRC8

Prototype	
`uint8 Crc_CalculateCRC8( const uint8* Crc_DataPtr, uint32 Crc_Length, uint8 Crc_StartValue8 )`	
Parameter	
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.
Return code	
uint8	8 bit result of CRC calculation.

Functional Description

The function Crc_CalculateCRC8 (service id: 1) performs a CRC8 calculation on Crc_Length data bytes, pointed to by Crc_DataPtr, with the starting value of Crc_StartValue8. The CRC8 routine is based on the SAE-J1850 CRC8 Standard:

SAE-J1850 CRC8 Standard	value
CRC result width:	8bits
Polynomial:	1Dh
Initial value:	0h
Input data reflected:	No
Result data reflected:	No
XOR value:	00h
Check:	37h

Particularities and Limitations

- If large data blocks have to be calculated (>32 bytes), the table based calculation method should be configured in order to decrease the calculation time.

Expected Caller Context

- Application, any module.

Table 6-3 Crc_CalculateCRC8

6.3.2 Crc_CalculateCRC16

Prototype	
`uint16 Crc_CalculateCRC16( const uint8* Crc_DataPtr, uint32 Crc_Length, uint16 Crc_StartValue16 )`	
Parameter	
Crc_DataPtr	Pointer to start address of data block to be calculated.
Crc_Length	Length of data block to be calculated in bytes.
Crc_StartValue16	Initial value when the algorithm starts.
Return code	
uint16	16 bit result of CRC calculation.

Functional Description

The function Crc_CalculateCRC16 (service id: 2) performs a CRC16 calculation on Crc_Length data bytes, pointed to by Crc_DataPtr, with the starting value of Crc_StartValue16. The CRC16 routine is based on the CCITT CRC16 Standard:

SAE-J1850 CRC8 Standard	value
CRC result width:	16bits
Polynomial:	1021h
Initial value:	FFFFh
Input data reflected:	No
Result data reflected:	No
XOR value:	0000h
Check:	29B1h

Particularities and Limitations

- If large data blocks have to be calculated (>32 bytes), the table based calculation method should be configured in order to decrease the calculation time.

Expected Caller Context

- Application, any module.

Table 6-4 Crc_CalculateCRC16

6.3.3 Crc_CalculateCRC32

Prototype	
`uint32 Crc_CalculateCRC32( const uint8* Crc_DataPtr, uint32 Crc_Length, uint32 Crc_StartValue32 )`	
Parameter	
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.
Return code	
uint32	32 bit result of CRC calculation.
Functional Description	

The function Crc_CalculateCRC32 (service id: 3) performs a CRC32 calculation on Crc_Length data bytes, pointed to by Crc_DataPtr, with the starting value of Crc_StartValue32. The CRC32 routine is based on the IEEE-802.3 CRC32 Ethernet Standard:

SAE-J1850 CRC8 Standard	value
CRC result width:	32bits
Polynomial:	04C11DB7h
Initial value:	FFFFFFFFh
Input data reflected:	Yes
Result data reflected:	Yes
XOR value:	FFFFFFFFh
Check:	CBF43926h

Particularities and Limitations

- If large data blocks have to be calculated (>32 bytes), the table based calculation method should be configured in order to decrease the calculation time.

Expected Caller Context

- Application, any module.

Table 6-5 Crc_CalculateCRC32

6.3.4 Crc_GetVersionInfo

Prototype	
`void Crc_GetVersionInfo( Std_VersionInfoType *versioninfo )`	
Parameter	
versioninfo	Pointer to where to store the version information of this module.
Return code	
none	--
Functional Description	
This service returns the version information of the module. The version information includes: ■ Module Id ■ Vendor Id ■ Vendor specific version numbers. ■ InstanceId This function is pre compile time configurable On/Off by the configuration parameter: `CRC_VERSION_INFO_API`.	
Particularities and Limitations	
■ None	
Expected Caller Context	
■ Application.	

Table 6-6 Crc_GetVersionInfo

6.4 Services used by CRC

The CRC component does not use services from other modules. Module CRC does not provide any error detection mechanism

6.5 Callback Functions

Module CRC's functions do not provide any call-back functions.

6.6 Configurable Interfaces

6.6.1 Notifications

The CRC module does not provide notifications for upper layers. The CRC routines return the checksum; this value will be evaluated by application or other modules.

6.6.2 Callout Functions

Further on the CRC module supports no callout function.

6.7 Service Ports

Service Ports are not available and necessary by this module.

7 Configuration

In the CRC the attributes can be configured with the following methods:

- Configuration in EAD for a detailed description see 7.1

7.1 Configuration with EAD

The CRC is configured with the help of the configuration tool EAD.

Info
In case of object delivery modifications are without effect, if a parameter is specified as a pre-compile value. Different object code is needed for different settings. Other object files can be obtained at Vector Informatik.

7.1.1 Start configuration of the CRC

The component name of the Cyclic Redundancy Check in EAD is “Crc”. In the “Architecture view” (initial page) of the EAD, the CRC can be opened by its context menu to start its configuration. Optionally, the CRC can be opened for configuration with the component list under the “System” tab located at the left side of the EAD.

7.1.2 CRC configuration tab

7.1.2.1 General Settings

This tab describes the three different CRC routines and the available methods, which can be used for the checksum.

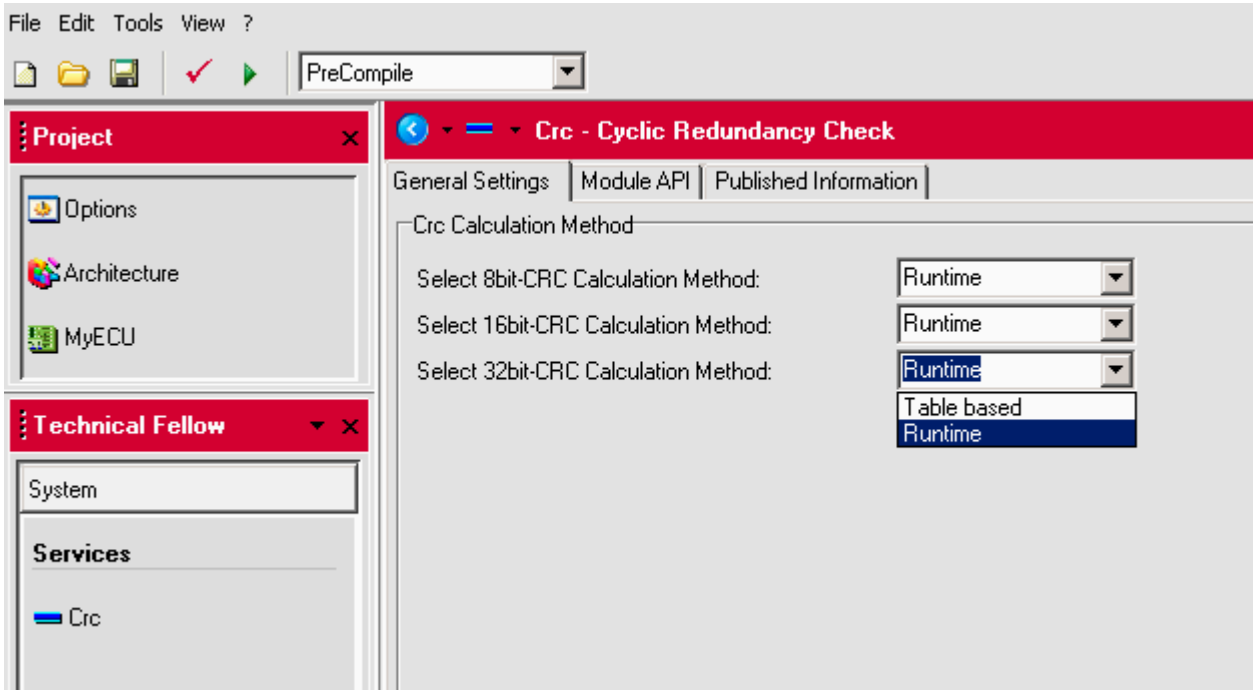

Figure 7-1 EAD Configuration – General Settings

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Select 8-Bit CRC Calculation Method	pre-compile	uint8	Table based **Runtime**	Selection between “Table based” calculation method and “Runtime” calculation method to calculate the CRC8.
Select 16-Bit CRC Calculation Method	pre-compile	uint8	Table based **Runtime**	Selection between “Table based” calculation method and “Runtime” calculation method to calculate the CRC16.
Select 32-Bit CRC Calculation Method	pre-compile	uint8	Table based **Runtime**	Selection between “Table based” calculation method and “Runtime” calculation method to calculate the CRC32.

Table 7-1 EAD Configuration – General Settings

7.1.2.2 Module API

In this Tab the provided APIs of the module are shown.

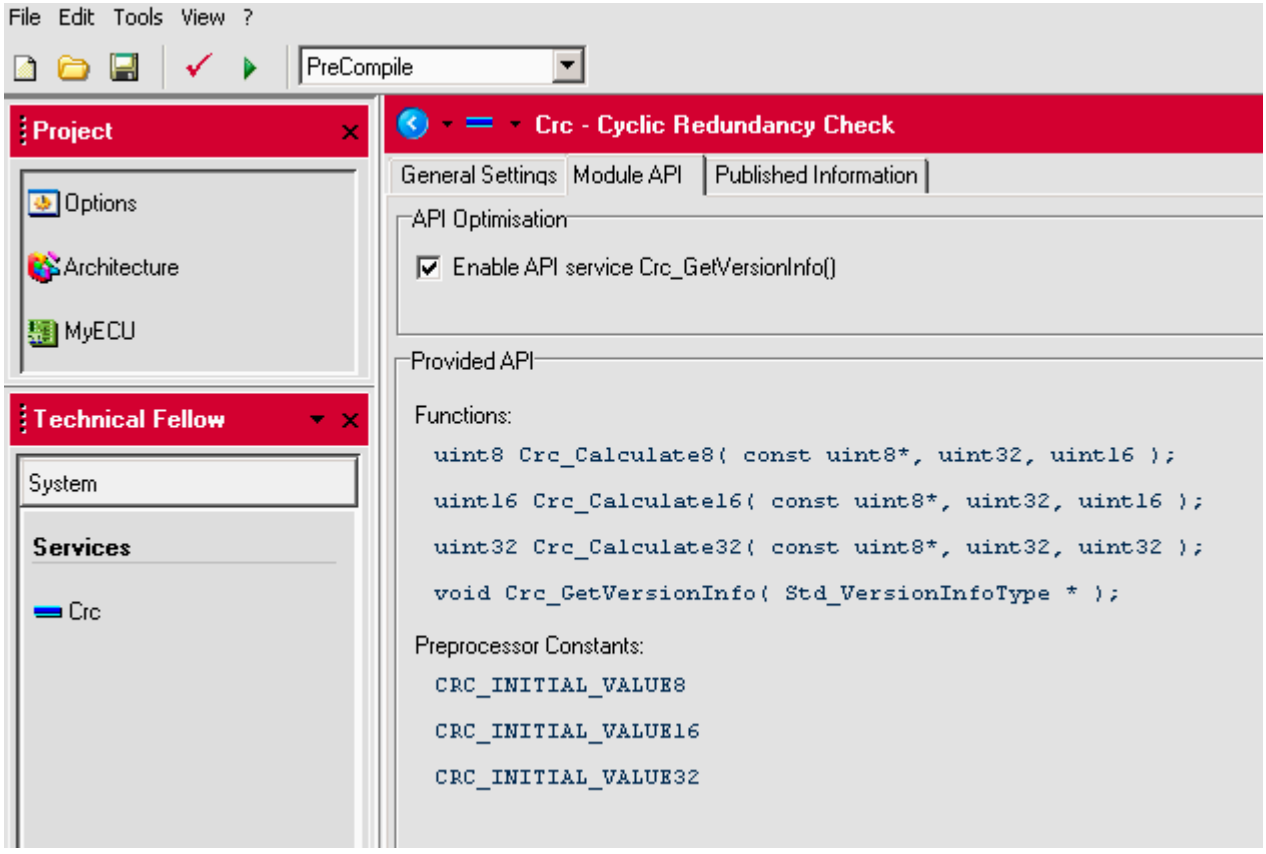

Figure 7-2 EAD Configuration – Module API

Additionally parts of the API can be removed to optimize the module towards ROM consumption:

Attribute Name	Configuration Variant	Value Type	Values The default value is written in bold	Description
Enable API service Crc_GetVersionInfo()	pre-compile	uint8	OFF **ON**	This switch can be used to reduce ROM consumption of module CRC. If this switch is deactivated, the code for the version information API is omitted by the compiler.

Table 7-2 EAD Configuration – Module API

8 AUTOSAR Standard Compliance

8.1 Deviations

No deviations exist.

8.2 Additions/ Extensions

No extensions available.

8.3 Limitations

No limitations exist.

9 Glossary and Abbreviations

9.1 Glossary

Term	Description
EAD	Embedded Architecture Designer; generation tool for MICROSAR components

Table 9-1 Glossary

9.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
EAD	Embedded Architecture Designer
ECU	Electronic Control Unit
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
PPort	Provide Port
RPort	Require Port
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification
CRC	Cyclic Redundancy Check

Table 9-2 Abbreviations

10 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector-informatik.com