MICROSAR Diagnostic Transformer

Technical Reference

Version 1.7.0

Authors	Christian Fischer, Sascha Sommer
Status	Released

Document Information

History

Author	Date	Version	Remarks
Christian Fischer	2016-06-23	1.0.0	Initial version
Christian Fischer	2016-11-18	1.1.0	Version update only
Bernd Sigle	2017-03-20	1.2.0	Version update only
Christian Fischer	2017-06-06	1.3.0	Version update only
Christian Fischer	2017-08-17	1.4.0	Support of AUTOSAR 4.3.0
Bernd Sigle	2017-03-20	1.5.0	Version update only
Sascha Sommer Bernd Sigle	2018-03-26	1.6.0	Adapted transformer length parameters according to AUTOSAR 4.3.x Updated referenced documents Added generator to static files
Sascha Sommer	2018-05-07	1.7.0	Reworked chapter 4.1

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V4.3.1
[2]	Vector	AN-ISC-8-1218_Atomic_Dcm_S-R_Interfaces_with_Diagnostic_Transformer.pdf	

Scope of the Document

This technical reference describes the general use of the MICROSAR Diagnostic Transformer.

Caution
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

Illustrations

Tables

1 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
1.0.0	Initial Creation
1.1.0	Version update for DiagXf 1.1.0 only
1.2.0	Version update only
1.3.0	Support UINT16_N, UINT32_N, SINT8_N, SINT16_N, SINT32_N arrays
1.4.0	Support of AUTOSAR 4.3.0
1.5.0	Version update only
1.6.0	Adapted transformer length parameters according to AUTOSAR 4.3.x
1.7.0	Version update only

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the MICROSAR BSW module DiagXf.

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	DIAGXF_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	DIAGXF_MODULE_ID	FF decimal (according to ref. [1])

* For the detailed functional specification please also refer to the corresponding AUTOSAR SWS.

The DiagXf module provides the functionality to serialize complex data.

2.1 Architecture Overview

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

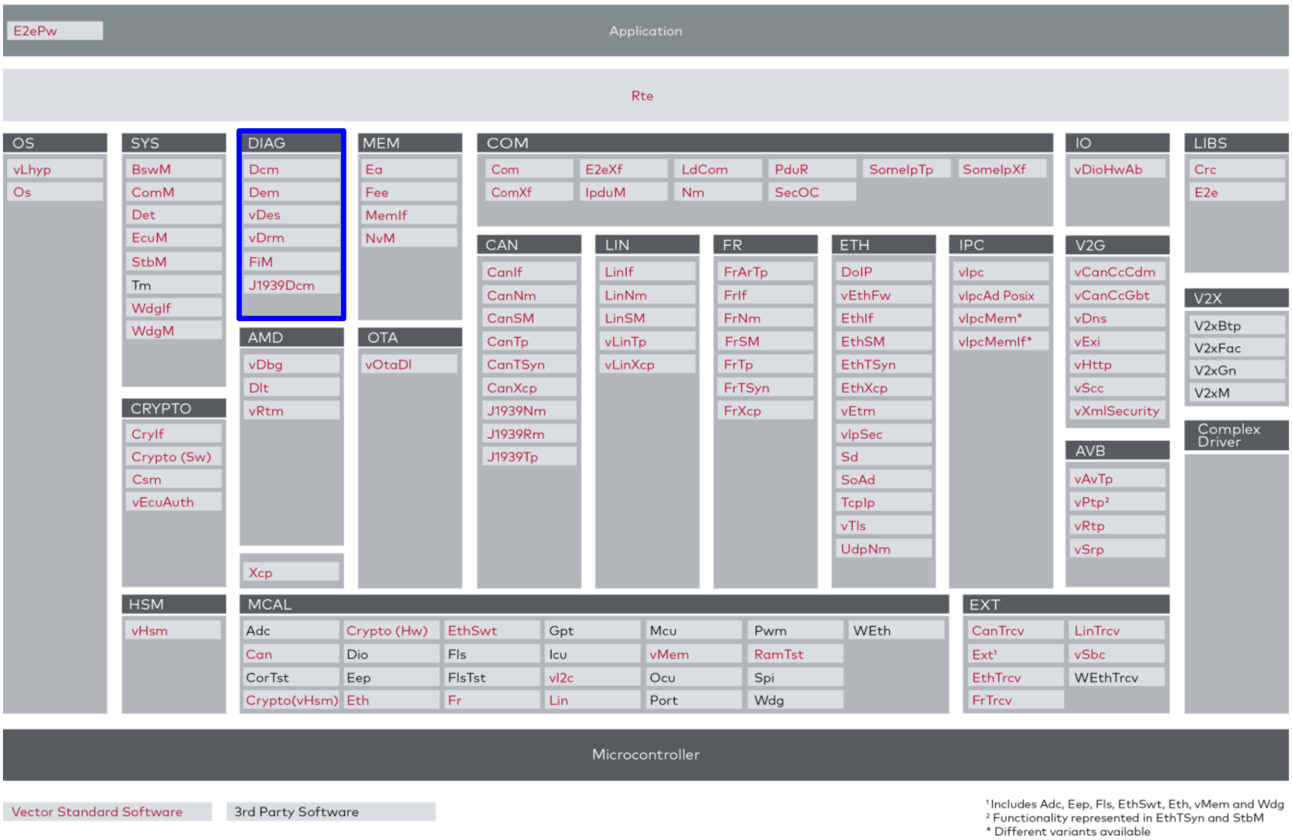

Figure 2-1 AUTOSAR Architecture Overview

3 Functional Description

3.1 Features

The features listed in the following tables cover the complete functionality specified for the DiagXf.

Supported Features
Serialization of diagnostic data elements
Deserialization of diagnostic data elements
Support for uint8, uint16, uint32, sint8, sint16, sint32, boolean record elements
Support for UINT16_N, UINT32_N, SINT8_N, SINT16_N, SINT32_N record elements

Table 3-1 Supported features

3.2 Initialization

The DiagXf does not have to be initialized or deinitialized. Calls to `DiagXf_Init()` and `DiagXf_DeInit()` can be omitted.

3.3 States

No internal states exist.

3.4 Main Functions

No main function exists because all functionality is performed within the called API.

3.5 Error Handling

3.5.1 Development Error Reporting

No development error reporting is currently supported by DiagXf.

3.5.2 Production Code Error Reporting

No production errors are specified for DiagXf.

4 Integration

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

4.1 Embedded Implementation

The delivery of the DiagXf consists out of these files:

File Name	Description	Integration Tasks
DiagXf.c	Generated source file of the DiagXf module.	-
DiagXf.h	Generated main header file which shall be included by modules using the DiagXf module.	-
DiagXf_MemMap.h	Generated file with template areas that can be adapted by the user. It contains the DiagXf specific part of the memory mapping.	Adapt the dedicated code areas within that file. See hints within that file.
DiagXf_Compiler_Cfg.h	Generated file with template areas that can be adapted by the user. It contains the DiagXf specific part of the compiler abstraction.	Adapt the dedicated code areas within that file. See hints within that file.
DiagXf_rules.mak, DiagXf_defs.mak, DiagXf_check.mak, DiagXf_cfg.mak	Generated make files according to the AUTOSAR make environment proposal. They are generated into the mak subdirectory.	-

Table 4-1 Implementation files

5 API Description

5.1 Services provided by DiagXf

5.1.1 DiagXf_Init

Prototype	
void **DiagXf_Init** (const DiagXf_ConfigType *config)	
Parameter	
config	Pointer to the transformer's configuration data.
Return code	
void	none
Functional Description	
Initialization function.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-1 DiagXf_Init

5.1.2 DiagXf_DeInit

Prototype	
void **DiagXf_DeInit** (void)	
Parameter	
void	none
Return code	
void	none
Functional Description	
Deinitialization function.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-2 DiagXf_DeInit

5.1.3 DiagXf_GetVersionInfo

Prototype	
`void` **`DiagXf_GetVersionInfo`** `(Std_VersionInfoType *versioninfo)`	
Parameter	
`versioninfo`	Pointer to where to store the version information of this module.
Return code	
`void`	none
Functional Description	
This API returns version information, vendor ID and AUTOSAR module ID of the called transformer module.	
Particularities and Limitations	
This API is only available if enabled by the configuration parameter XfrmVersionInfoApi.	
Expected Caller Context	
This function can be called in any context.	

Table 5-3 DiagXf_GetVersionInfo

5.1.4 DiagXf_<transformerId>

Prototype	
`Std_ReturnType` **`DiagXf_<transformerId>`** `(uint8 *buffer, uint32 *bufferLength, const <type> *dataElement)`	
Parameter	
`buffer`	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer.
`bufferLength`	Used length of the buffer.
`dataElement`	Data element which shall be transformed.
Return code	
`E_OK`	Serialization successful.
Functional Description	
Serialization of the dataElement when communicating from the DataPrototypeMapping.firstDataPrototype to the DataPrototypeMapping.secondDataPrototype.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-4 DiagXf_<transformerId>

5.1.5 DiagXf_Inv_<transformerId>

Prototype	
`Std_ReturnType` **`DiagXf_Inv_<transformerId>`** `(const uint8 *buffer, uint32 bufferLength, <type> *dataElement)`	
Parameter	
`buffer`	Buffer allocated by the RTE, where the serialized data is stored by the Rte.
`bufferLength`	Used length of the buffer.
`dataElement`	Data element which is the result of the transformation and contains the deserialized data element.
Return code	
`E_OK`	Deserialization successful.
Functional Description	
Deserialization of the buffer when communicating from the DataPrototypeMapping.secondDataPrototype to the DataPrototypeMapping.firstDataPrototype.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-5 DiagXf_Inv_<transformerId>

6 Configuration

In the DiagXf the attributes can be configured with the following tools:

- Configuration in DaVinci Configurator

Currently, only the GetVersionInfo API can be enabled / disabled in the DiagXf Ecu configuration.
The serialization / deserialization is based on the `DiagnosticDataElement` described in the DiagnosticExtract. The `BitOffset`, `BaseTypeSize` and `ByteOrder` are considered for each `DiagnosticDataElement`. It is assumed that the `DiagnosticDataElements` are aligned to a byte boundary.
If two incompatible ports are connected using a `DataPrototypeMapping` which references a diagnostic transformer through `firstToSecondDataTransformation`, the DiagXf implementation shall be generated.

6.1 Configuration Variants

The DiagXf supports the configuration variants

- `VARIANT-PRE-COMPILE`

The configuration classes of the DiagXf parameters depend on the supported configuration variants. For their definitions please consider `DiagXf_bswmd.arxml`.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator	Configuration and generation tool for MICROSAR components

Table 7-1 Glossary

7.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DEM	Diagnostic Event Manager
DET	Development Error Tracer
ECU	Electronic Control Unit
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
RTE	Runtime Environment
SRS	Software Requirement Specification
SWC	Software Component
SWS	Software Specification

Table 7-2 Abbreviations

8 Additional Copyrights

The MICROSAR DIAGXF Generator contains *Free and Open Source Software* (FOSS). The following table lists the files which contain this software, the kind and version of the FOSS, the license under which this FOSS is distributed and a reference to a license file which contains the original text of the license terms and conditions. The referenced license files can be found in the directory of the RTE Generator.

File	FOSS	License	License Reference
MicrosarDiagXfGen.exe	Perl 5.20	Artistic License	License_Artistic.txt
MicrosarDiagXfGen64.exe	Perl 5.20	Artistic License	License_Artistic.txt

Table 8-1 Free and Open Source Software Licenses

9 Contact

Visit our website for more information on

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

www.vector.com