VECTOR

MICROSAR SOME/IP Transformer

Technical Reference

Version 1.15.0

Authors	Cornelius Reuss, Sascha Sommer, Patrick Alschbach, Michael Salmen, Siegfried Derksen
Status	Released

Document Information

History

Author	Date	Version	Remarks
Cornelius Reuss	2015-07-07	1.0.0	Initial version
Cornelius Reuss	2016-02-17	1.1.0	Update to AR 4.2.2
Sascha Sommer	2016-05-17	1.2.0	Version update only
Sascha Sommer	2016-05-17	1.3.0	Version update only
Cornelius Reuss	2016-06-23	1.4.0	Version update only
Patrick Alschbach	2016-11-16	1.5.0	Version update only
Bernd Sigle	2017-03-20	1.6.0	Version update only
Patrick Alschbach	2017-06-06	1.7.0	Added information about default behavior
Patrick Alschbach	2017-08-17	1.8.0	Minor improvements
Patrick Alschbach	2017-10-17	1.9.0	Adaptations related to DaVinci Developer 4.X
Sascha Sommer	2018-03-26	1.10.0	Adapted transformer length parameters according to AUTOSAR 4.3.0 Added generator to static files
Sascha Sommer	2018-05-07	1.11.0	Updated referenced AUTOSAR documents to AUTOSAR 4.3.1 Reworked chapter 4.1 Added deviation
Sascha Sommer Siegfried Derksen	2018-09-28	1.12.0	Version update only Added deviation
Sascha Sommer Siegfried Derksen Michael Salmen	2019-04-04	1.13.0	Added deviation Minor improvements
Sascha Sommer Siegfried Derksen	2019-08-27	1.14.0	Support session handling for sender-receiver communication Support fire and forget methods without parameters Support TLV with optional data elements
Sascha Sommer	2020-04-06	1.15.0	Removed 32-bit build Support for UTF-8 Strings without BOM

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_SOMEIPTransformer.pdf	4.3.1
[2]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	4.3.1

Scope of the Document

This technical reference describes the general use of the SOME/IP 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

1 Component History 7

2 Introduction 8
2.1 Architecture Overview 8

3 Functional Description 9
3.1 Features 9
3.1.1 Deviations 9
3.2 Initialization 10
3.3 States 10
3.4 Main Functions 10
3.5 Error Handling 10
3.5.1 Development Error Reporting 10
3.5.2 Production Code Error Reporting 10

4 Integration 11
4.1 Embedded Implementation 11

5 API Description 12
5.1 Services provided by SomeIpXf 12
5.1.1 SomeIpXf_InitMemory 12
5.1.2 SomeIpXf_Init 12
5.1.3 SomeIpXf_DeInit 13
5.1.4 SomeIpXf_GetVersionInfo 14
5.1.5 Sender / Receiver communication 15
5.1.5.1 SomeIpXf_<transformerId> 15
5.1.5.2 SomeIpXf_Inv_<transformerId> 16
5.1.6 Client / Server communication 17
5.1.6.1 SomeIpXf_<transformerId> 17
5.1.6.2 SomeIpXf_Inv_<transformerId> 18

6 Configuration 19
6.1 Configuration Variants 19
6.2 Enabling / Disabling of data transformation 19
6.3 Configuration of Sender / Receiver Communication 19

7 Glossary and Abbreviations 20
7.1 Glossary 20

8 Additional Copyrights 21

8.1 Abbreviations 21

9 Contact 22

Illustrations

Figure 2-1 AUTOSAR 4.2 Architecture Overview 8

Tables

Table 1-1 Component history 7
Table 3-1 Supported AUTOSAR standard conform features 9
Table 3-2 Not supported AUTOSAR standard conform features 10
Table 4-1 Implementation files 11
Table 5-1 SomeIpXf_InitMemory 12
Table 5-2 SomeIpXf_Init 13
Table 5-3 SomeIpXf_DeInit 13
Table 5-4 SomeIpXf_GetVersionInfo 14
Table 5-5 SomeIpXf_<transformerId> 15
Table 5-6 SomeIpXf_Inv_<transformerId> 16
Table 5-7 SomeIpXf_<transformerId> 17
Table 5-8 SomeIpXf_Inv_<transformerId> 18
Table 7-1 Glossary 20
Table 8-1 Free and Open Source Software Licenses 21
Table 8-2 Abbreviations 21

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	Fixed union length access
1.2.0	Update to AR 4.2.2 > Corrected MessageType for Application Error > Support length of length field configuration > Support Message Type "Notification" > Support for new return codes
1.3.0	Support autonomous error responses
1.4.0	Fixed length checks for dynamic data Support configuration of interface version
1.5.0	MISRA enhancements
1.6.0	Version update only
1.7.0	Version update only
1.8.0	Minor improvements
1.9.0	Version update only
1.10.0	Adapted transformer length parameters according to AUTOSAR 4.3.0
1.11.0	Version update only
1.12.0	Version update only
1.13.0	Code was reworked for MISRA-C2012 compliance Support TLV with non-Autosar-conform TlvDataIdDefinition
1.14.0	Support session handling for sender-receiver communication Support fire and forget methods without parameters Support optional TLV data elements with non-Autosar-conform definition of OptionalElementStructure
1.15.0	Removed 32-bit build Support for UTF-8 Strings without BOM

Table 1-1 Component history

2 Introduction

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

Supported AUTOSAR Release*:	4	
Supported Configuration Variants:	pre-compile	
Vendor ID:	SOMEIPXF_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	SOMEIPXF_MODULE_ID	174 decimal (according to ref. [2])

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

The SomeIpXf module provides the functionality to serialize data in the SOME/IP on-the-wire format.

2.1 Architecture Overview

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

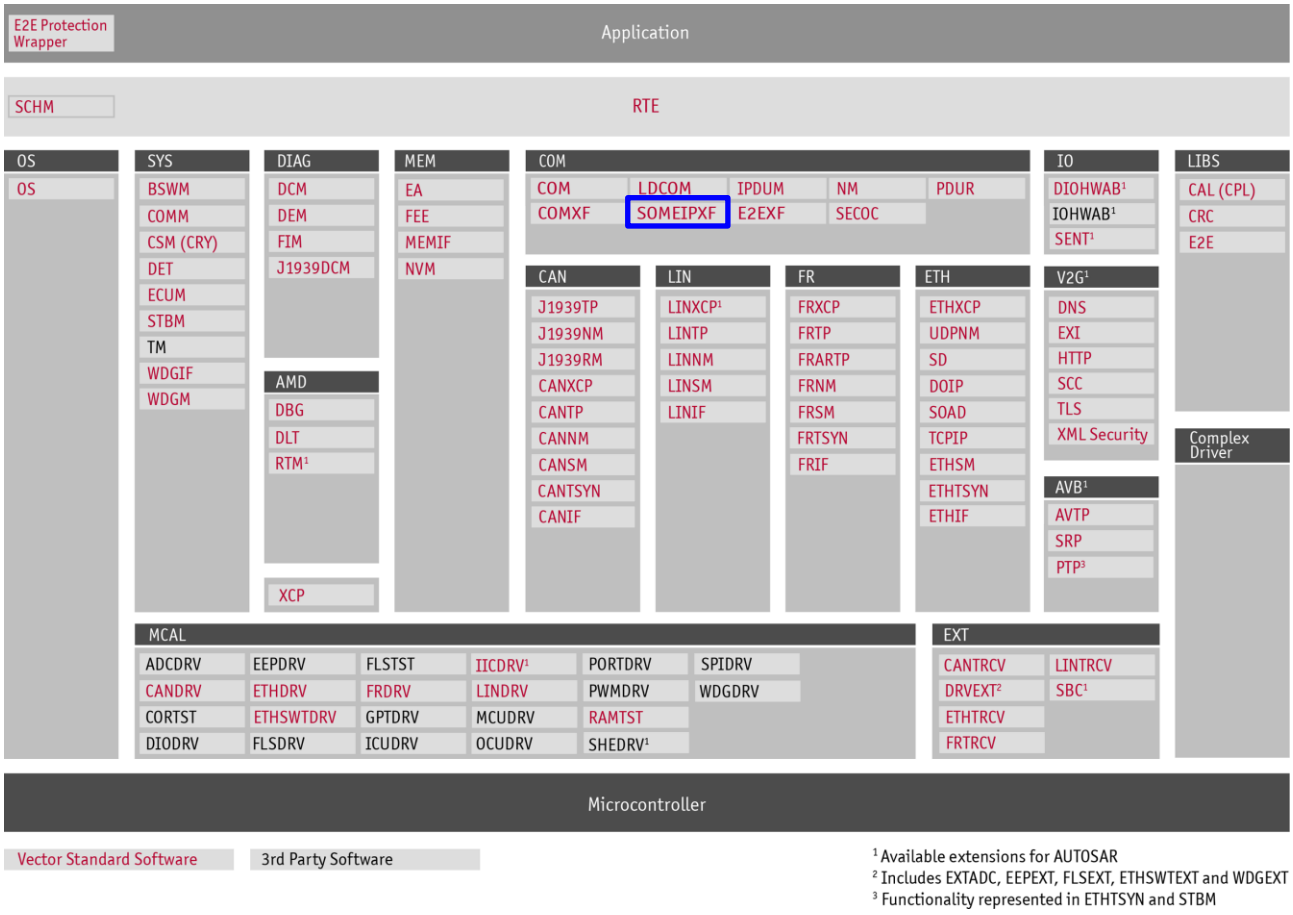

Figure 2-1 AUTOSAR 4.2 Architecture Overview

3 Functional Description

3.1 Features

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

The AUTOSAR standard functionality is specified in [1], the corresponding features are listed in the tables

> Table 3-1 Supported AUTOSAR standard conform features

> Table 3-2 Not supported AUTOSAR standard conform features

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Serialization / Deserialization of complex data for S/R communication.
Serialization / Deserialization of complex data for C/S communication.

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

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

Not Supported AUTOSAR Standard Conform Features
The serialization / deserialization of the following data types: - Bitfields - Extensible structs
Development error detection.
Message type for an autonomous error response is still set to "ERROR (0x81)" as specified in AR 4.2.2 and before.
If message type of transformed Sender/Receiver communication is not set, “REQUEST_NO_RETURN (0x01)” is used as default.
If less data than expected are handed over the SOME/IP transformer during deserialization of data, initial value is not considered and deserialization will always abort with E_SER_MALFORMED_MESSAGE.
If the length is greater than the expected length of a struct (as specified in the data type definition), the unexpected data is not skipped and deserialization aborts with E_SER_MALFORMED_MESSAGE.
Endianness for UTF-16 strings is not evaluated at the base type. Performing a conversion if platform endianness and SOMEIPTransformerDescritption.byteOrder have different values.
DataPrototypeTransformationProps
TLV with optional unions
Dynamic length array of structures with optional struct elements

Not Supported AUTOSAR Standard Conform Features
Optional acknowledgment message types
SomeIP transformer attributes: - SOMEIPTransformationProps: o alignment o sizeOfStructLengthField o sizeOfArrayLengthField o sizeOfUnionLengthField - SOMEIPTransformationISignalProps: o isDynamicLengthFieldSize

Table 3-2 Not supported AUTOSAR standard conform features

3.2 Initialization

The SomeIpXf only needs to be initialized when session handling for sender-receiver communication is configured. Otherwise, calls to `SomeIpXf_InitMemory()`, `SomeIpXf_Init()` and `SomeIpXf_DeInit()` can be omitted. On platforms in which the Random Access Memory (RAM) is not initialized by the startup code, the function `SomeIpXf_InitMemory` has to be called first and then a call to `SomeIpXf_Init` can be realized.

3.3 States

The transformer internally stores the sequence counter when session handling for sender-receiver communication is configured.

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 the SomeIpXf.

3.5.2 Production Code Error Reporting

No production errors are specified for the SomeIpXf.

4 Integration

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

4.1 Embedded Implementation

The delivery of the SomeIpXf consists of:

File Name	Description	Integration Tasks
SomeIpXf.c	Generated source file of the SomeIpXf module.	-
SomeIpXf.h	Generated main header file which shall be included by modules using the SomeIpXf module.	-
SomeIpXf_MemMap.h	Generated file with template areas that can be adapted by the user. It contains the SomeIpXf specific part of the memory mapping.	Adapt the dedicated code areas within that file. See hints within that file.
SomeIpXf_Compiler_Cfg.h	Generated file with template areas that can be adapted by the user. It contains the SomeIpXf specific part of the compiler abstraction.	Adapt the dedicated code areas within that file. See hints within that file.

Table 4-1 Implementation files

5 API Description

5.1 Services provided by SomeIpXf

5.1.1 SomeIpXf_InitMemory

Prototype	
`void` **`SomeIpXf_InitMemory`** `(void)`	
Parameter	
`void`	none
Return code	
`void`	none
Functional Description	
This service initializes the transformer internal state variables if the compiler does not support initialized variables.	
Particularities and Limitations	
> This function is synchronous. > This function is non-reentrant. > If this function is used it shall be called before any other SomeIpXf function after startup.	
Expected Caller Context	
This function needs to be called either from trusted context or before the MPU is activated.	

Table 5-1 SomeIpXf_InitMemory

5.1.2 SomeIpXf_Init

Prototype	
`void` **`SomeIpXf_Init`** `(const SomeIpXf_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 5-2 SomeIpXf_Init

5.1.3 SomeIpXf_DeInit

Prototype	
`void SomeIpXf_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 5-3 SomeIpXf_DeInit

5.1.4 SomeIpXf_GetVersionInfo

Prototype	
`void` **`SomeIpXf_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 5-4 SomeIpXf_GetVersionInfo

5.1.5 Sender / Receiver communication

5.1.5.1 SomeIpXf_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_<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.
`SOMEIPXF_E_SER_GENERIC_ERROR`	A generic error occurred.
Functional Description	
Serialization of data element based on the SOME/IP on the wire format for S/R communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-5 SomeIpXf_<transformerId>

5.1.5.2 SomeIpXf_Inv_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_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.
`SOMEIPXF_E_SER_GENERIC_ERROR`	A generic error occurred.
`SOMEIPXF_E_SER_WRONG_PROTOCOL_VERSION`	The version of the receiving transformer did not match to the version of the sending transformer.
`SOMEIPXF_E_SER_WRONG_INTERFACE_VERSION`	Interface version of serialized data is not supported.
`SOMEIPXF_E_SER_MALFORMED_MESSAGE`	The received data was malformed. No valid output could be produced.
`SOMEIPXF_E_SER_WRONG_MESSAGE_TYPE`	The received message type was not expected.
Functional Description	
Deserialization of data element based on the SOME/IP on the wire format for S/R communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-6 SomeIpXf_Inv_<transformerId>

5.1.6 Client / Server communication

5.1.6.1 SomeIpXf_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_<transformerId>`** `(const Rte_Cs_TransactionHandleType *transactionHandle, uint8 *buffer, uint32 *bufferLength, [Std_ReturnType returnValue], [<type> data_1] ... [<type> data_n])`	
Parameter	
`transactionHandle`	Transaction handle (clientId and sequenceCounter) needed to differentiate between multiple requests.
`buffer`	Buffer allocated by the RTE, where the transformed data has to be stored by the transformer.
`bufferLength`	Used length of the buffer.
`returnValue`	Return value of the server runnable which needs to be serialized on server side for transmission to the calling client. This argument is only available for serializers of the response of a Client/Server communication.
`data_1`	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
`data_n`	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
Return code	
`E_OK`	Serialization successful.
`SOMEIPXF_E_SER_GENERIC_ERROR`	A generic error occurred.
Functional Description	
Serialization of data element based on the SOME/IP on the wire format for C/S communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-7 SomeIpXf_<transformerId>

5.1.6.2 SomeIpXf_Inv_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_Inv_<transformerId>`** `(Rte_Cs_TransactionHandleType *transactionHandle, const uint8 *buffer, uint32 bufferLength, [Std_ReturnType *returnValue], [<type> *data_1] ... [<type> *data_n])`	
Parameter	
`transactionHandle`	Transaction handle (clientId and sequenceCounter) needed to differentiate between multiple requests.
`buffer`	Buffer allocated by the RTE, where the serialized data is stored by the Rte.
`bufferLength`	Used length of the buffer.
`returnValue`	Return value of the server runnable which needs to be serialized on server side for transmission to the calling client. This argument is only available for deserializers of the response of a Client/Server communication.
`data_1`	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
`data_n`	Client/Server operation argument which shall be transformed (in the same order as in the corresponding interface).
Return code	
`E_OK`	Deserialization successful.
`SOMEIPXF_E_SER_GENERIC_ERROR`	A generic error occurred.
`SOMEIPXF_E_SER_WRONG_PROTOCOL_VERSION`	The version of the receiving transformer did not match to the version of the sending transformer.
`SOMEIPXF_E_SER_WRONG_INTERFACE_VERSION`	Interface version of serialized data is not supported.
`SOMEIPXF_E_SER_MALFORMED_MESSAGE`	The received data was malformed. No valid output could be produced.
`SOMEIPXF_E_SER_WRONG_MESSAGE_TYPE`	The received message type was not expected.
Functional Description	
Deserialization of data element based on the SOME/IP on the wire format for C/S communication.	
Particularities and Limitations	
none	
Expected Caller Context	
This function can be called in any context.	

Table 5-8 SomeIpXf_Inv_<transformerId>

6 Configuration

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

- Configuration in DaVinci Configuration

Currently, only the GetVersionInfo API can be enabled / disabled in the SomeIpXf Ecu configuration.

6.1 Configuration Variants

The SomeIpXf supports the configuration variants

- `VARIANT-PRE-COMPILE`

The configuration classes of the SomeIpXf parameters depend on the supported configuration variants. For their definitions please see the `SomeIpXf_bswmd.arxml` file.

6.2 Enabling / Disabling of data transformation

It is possible to enable the SomeIpXf through the configuration of a transformer chain in the system description according to AUTOSAR.

6.3 Configuration of Sender / Receiver Communication

The message types of sender / receiver communication can be `REQUEST_NO_RETURN (0x01)` or `NOTIFICATION (0x02)` according to AUTOSAR. The specification allows this parameter to be undefined without specifying a default value. Currently, if the parameter is not set `REQUEST_NO_RETURN` is used as default.

7 Glossary and Abbreviations

7.1 Glossary

Term	Description
DaVinci Configurator	Configuration and generation tool for MICROSAR components

Table 7-1 Glossary

8 Additional Copyrights

The MICROSAR SOMEIPXF 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
MicrosarSomeIpXfGen64.exe	Perl 5.20	Artistic License	License_Artistic.txt

Table 8-1 Free and Open Source Software Licenses

8.1 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 8-2 Abbreviations

9 Contact

Visit our website for more information on

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

www.vector.com