MICROSAR SOME/IP Transformer

Technical Reference

Version 1.8.0

Authors	Cornelius Reuss, Sascha Sommer, Patrick Alschbach
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

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_SOMEIPTransformer.pdf	4.2.2
[2]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	4.2.2

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

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	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

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.

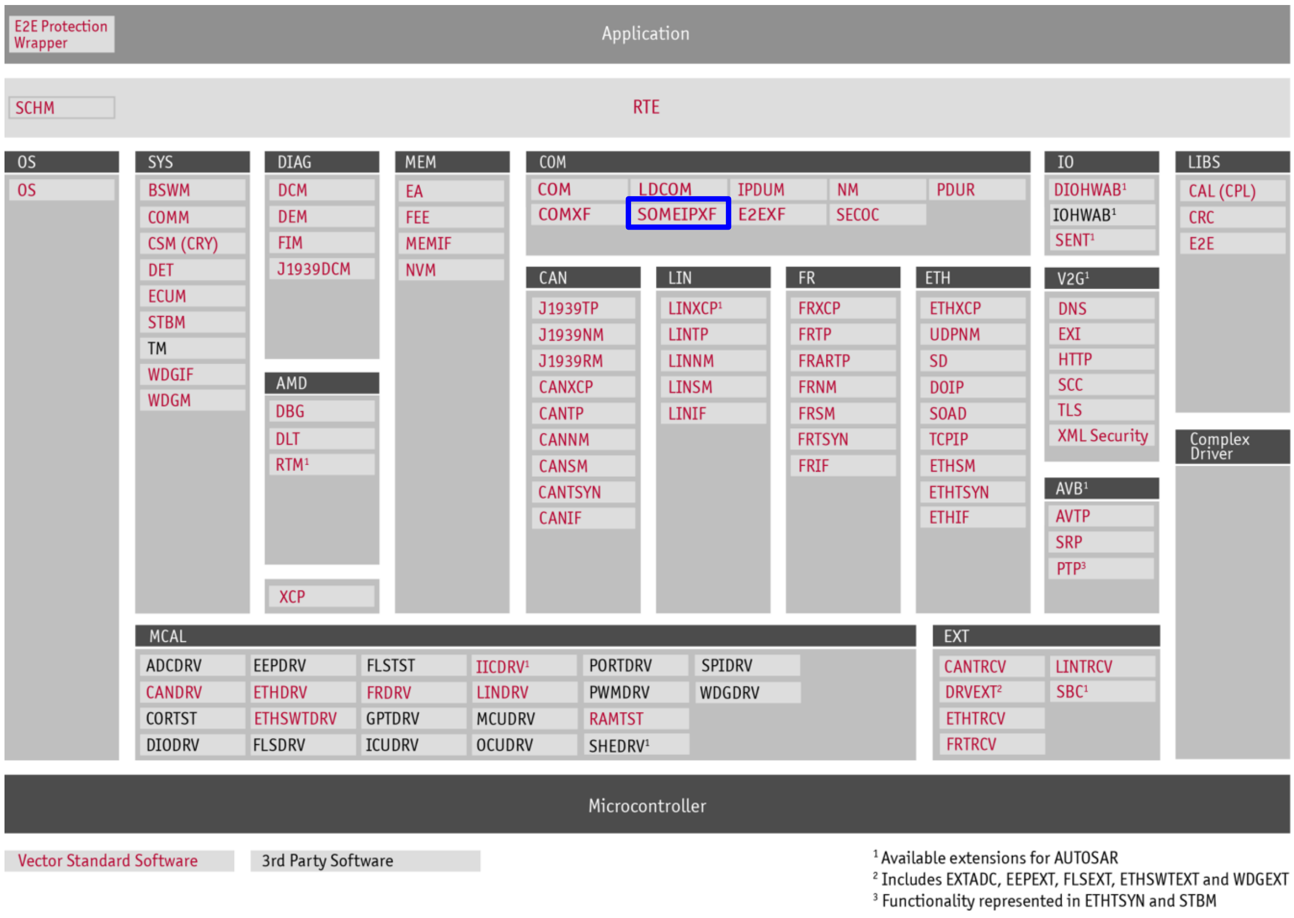


Figure 2-1 AUTOSAR 4.2 Architecture Overview

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 is not supported: - Bitfields - Extensible structs
Development error detection.

Table 3-2 Not supported AUTOSAR standard conform features

3.2 Initialization

The SomeIpXf does not have to be initialized or deinitialized. Calls to `SomeIpXf_Init()` and `SomeIpXf_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 SomeIpXf.

3.5.2 Production Code Error Reporting

No production errors are specified for SomeIpXf.

4 Integration

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

4.1 Scope of Delivery

The delivery of the SomeIpXf contains the files which are described in the chapters 4.1.1 and 4.1.2.

4.1.1 Static Files

File Name	Description
-	-

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool DaVinci Configurator.

File Name	Description
SomeIpXf.c	Source file of the SomeIpXf module.
SomeIpXf.h	Main header file which shall be included by modules using the SomeIpXf module.
SomeIpXf_MemMap.h	Template contains SomeIpXf specific part of the memory mapping.
SomeIpXf_Compiler_Cfg.h	Template contains SomeIpXf specific part of the compiler abstraction.
SomeIpXf_rules.mak, SomeIpXf_defs.mak, SomeIpXf_check.mak, SomeIpXf_cfg.mak	Make files according to the AUTOSAR make environment proposal are generated into the mak subdirectory.

Table 4-2 Generated files

5 API Description

5.1 Services provided by SomeIpXf

5.1.1 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 5-1 SomeIpXf_Init

5.1.2 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 5-2 SomeIpXf_DeInit

5.1.3 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 5-3 SomeIpXf_GetVersionInfo

5.1.4 Sender / Receiver communication

5.1.4.1 SomeIpXf_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_<transformerId>`** `(uint8 *buffer, uint16 *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 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 5-4 SomeIpXf_<transformerId>

5.1.4.2 SomeIpXf_Inv_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_Inv_<transformerId>`** `(const uint8 *buffer, uint16 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 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_Inv_<transformerId>

5.1.5 Client / Server communication

5.1.5.1 SomeIpXf_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_<transformerId>`** `(const Rte_Cs_TransactionHandleType *transactionHandle, uint8 *buffer, uint16 *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 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 5-6 SomeIpXf_<transformerId>

5.1.5.2 SomeIpXf_Inv_<transformerId>

Prototype	
`Std_ReturnType` **`SomeIpXf_Inv_<transformerId>`** `(Rte_Cs_TransactionHandleType *transactionHandle, const uint8 *buffer, uint16 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 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_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

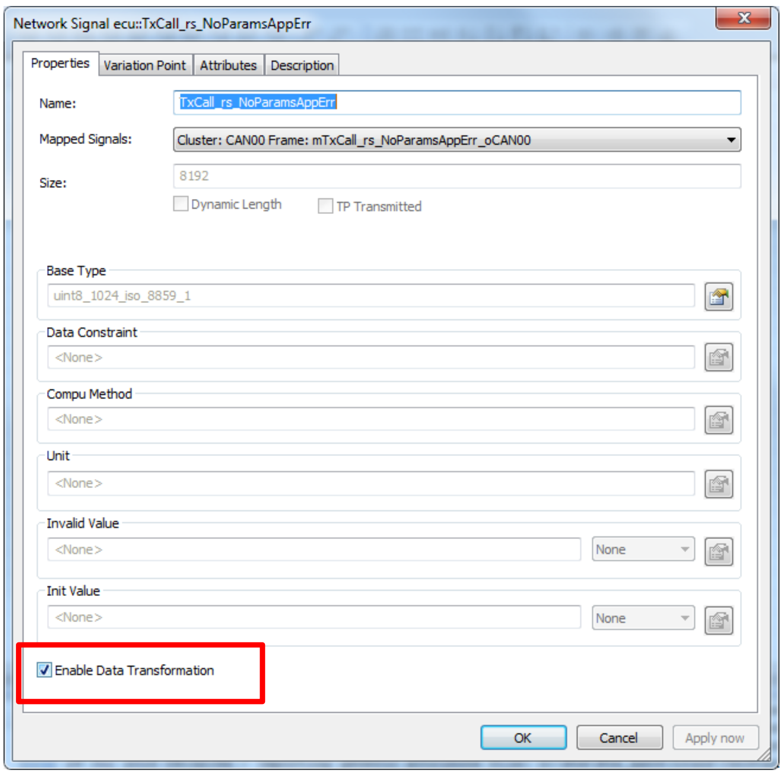
The SomeIpXf supports the configuration variants

> `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

If a signal shall be handled by the SomeIpXf, the attribute “Enable Data Transformation” has to be set in the “Signal Properties” dialog in the DaVinci Developer (see Figure 6-1 Enable Data Transformation).



Name: TxCall_rs_NoParamsAppErr
Mapped Signals: Cluster: CAN00 Frame: mTxCall_rs_NoParamsAppErr_oCAN00
Size: 8192
☐ Dynamic Length ☐ TP Transmitted
Base Type: uint8_1024_iso_8859_1
Data Constraint: <None>
Compu Method: <None>
Unit: <None>
Invalid Value: <None> None
Init Value: <None> None
☑ Enable Data Transformation

Figure 6-1 Enable Data Transformation

It is also 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

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 Contact

Visit our website for more information on

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

www.vector.com