

MICROSAR SOME/IP Transport Protocol

Technical Reference

Version 4.0.0

Authors	Supraja Kovuru, Anthony Thomas
Status	Released

Document Information

History

Author	Date	Version	Remarks
Supraja Kovuru	2018-01-19	1.0.0	Initial Version
Anthony Thomas	2018-02-13	1.0.1	Added section showing how to transmit and receive data without the RTE API.
Supraja Kovuru	2018-10-22	2.0.0	Updated SomelpTpRxTimeoutTime according to AR 4.3.1
Supraja Kovuru	2020-04-03	3.0.0	Added Section 3.1.2.3 and Section 6.2.3
Supraja Kovuru	2020-06-18	3.1.0	Support of metadata. Updated Section 3.1
Supraja Kovuru	2020-06-25	3.2.0	Updated Section 3.5.2
Anthony Thomas	2020-11-25	3.2.1	Clarification of the metadata handling.
Supraja Kovuru	2021-12-16	4.0.0	Updated Figure 2-2

Reference Documents

No.	Source	Title	Version
[1]	AUTOSAR	AUTOSAR_SWS_SOMEIPTransportProtocol.pdf	4.3.1
[2]	AUTOSAR	AUTOSAR_SWS_DET.pdf	4.3.1
[3]	AUTOSAR	AUTOSAR_TR_SomelpExample.pdf	4.2.1
[4]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	4.3.1
[5]	Vector	TechnicalReference_SocketAdaptor.pdf	8.4.0 or later

Scope of the Document

This technical reference describes the general use of the SomelpTp basis software.



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	Introduction.....	6
1.1	Architecture Overview	7
2	Functional Description	9
2.1	Features	9
2.1.1	Deviations	9
2.1.2	Additions/ Extensions.....	9
2.1.2.1	Transmit Queue	10
2.1.2.2	Monitoring Transmission Confirmation timeout.....	10
2.1.2.3	Burst Transmission of TP Segments	10
2.2	Initialization	10
2.3	States	10
2.4	Main Functions	10
2.5	Error Handling.....	11
2.5.1	Development Error Reporting.....	11
2.5.2	Runtime Error Reporting	12
3	Integration.....	13
3.1	Scope of Delivery.....	13
3.1.1	Static Files	13
3.1.2	Dynamic Files	13
3.1.3	User modifiable templates.....	13
3.2	Include Structure.....	14
3.3	Critical Sections	14
4	API Description.....	16
4.1	Services provided by SomelpTp	16
4.1.1	SomelpTp_GetVersionInfo	16
4.1.2	SomelpTp_InitMemory	16
4.1.3	SomelpTp_Init.....	17
4.1.4	SomelpTp_Transmit.....	17
4.1.5	SomelpTp_MainFunctionTx	18
4.1.6	SomelpTp_MainFunctionRx.....	18
4.2	Services used by SomelpTp	19
4.3	Callback Functions.....	19
4.3.1	SomelpTp_TriggerTransmit.....	19
4.3.2	SomelpTp_TxConfirmation	20
4.3.3	SomelpTp_RxIndication	21
4.4	Configurable Interfaces	21

5	Configuration	22
5.1	Configuration Variants	22
5.2	Additional configuration hints	22
5.2.1	SomelpTpRxTimeoutTime	22
5.2.2	SomelpTpTxConfirmationTimeoutTime	22
5.2.3	SomelpTpTxBurstSize	22
5.2.4	Sending and transmitting data without the RTE API	22
5.2.4.1	Configuring the LdCom with the DaVinci Configurator....	23
5.2.4.2	Adjusting the templates.....	24
5.2.4.3	Reading and Writing data from the SWC	24
6	Glossary and Abbreviations	26
6.1	Glossary	26
6.2	Abbreviations	26
7	Contact.....	27

Illustrations

Figure 1-1	AUTOSAR 4.3 Architecture Overview	7
Figure 1-2	Interfaces to adjacent modules of the SomelpTp	8
Figure 3-1	Include Structure.....	14
Figure 5-1	Configuration of the LdCom's I-PDU used during reception.	23
Figure 5-2	Configuration of the LdCom's I-PDU used during transmission.	24

Tables

Table 2-1	Supported AUTOSAR standard conform features	9
Table 2-2	Not supported AUTOSAR standard conform features	9
Table 2-3	Features provided beyond the AUTOSAR standard	10
Table 2-4	Service IDs	11
Table 2-5	Development errors reported to DET	11
Table 2-6	Runtime errors reported to DET	12
Table 3-1	Static files	13
Table 3-2	Generated files	13
Table 3-3	User modifiable templates	13
Table 4-1	SomelpTp_GetVersionInfo.....	16
Table 4-2	SomelpTp_InitMemory	17
Table 4-3	SomelpTp_Init	17
Table 4-4	SomelpTp_Transmit	18
Table 4-5	SomelpTp_MainFunctionTx.....	18
Table 4-6	SomelpTp_MainFunctionRx	19
Table 4-7	Services used by the SomelpTp	19
Table 4-8	SomelpTp_TriggerTransmit	20
Table 4-9	SomelpTp_TxConfirmation	20
Table 4-10	SomelpTp_RxIndication.....	21
Table 5-1	Functions intended to be used by the SWC.	25
Table 6-1	Glossary	26
Table 6-2	Abbreviations.....	26

1 Introduction

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

Supported AUTOSAR Release*:	4.3.1	
Supported Configuration Variants:	pre-compile, post-build	
Vendor ID:	SomelpTp_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	SomelpTp_MODULE_ID	177 decimal (according to ref. [4])

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

The SomelpTp module based on AUTOSAR SOME/IP Transport Protocol provides the following services :

- > Segments SOME/IP packets which do not fit into one single UDP packet.
- > Reassembles the received SOME/IP segments.
- > Detection of errors during segmentation and reassembly.

1.1 Architecture Overview

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

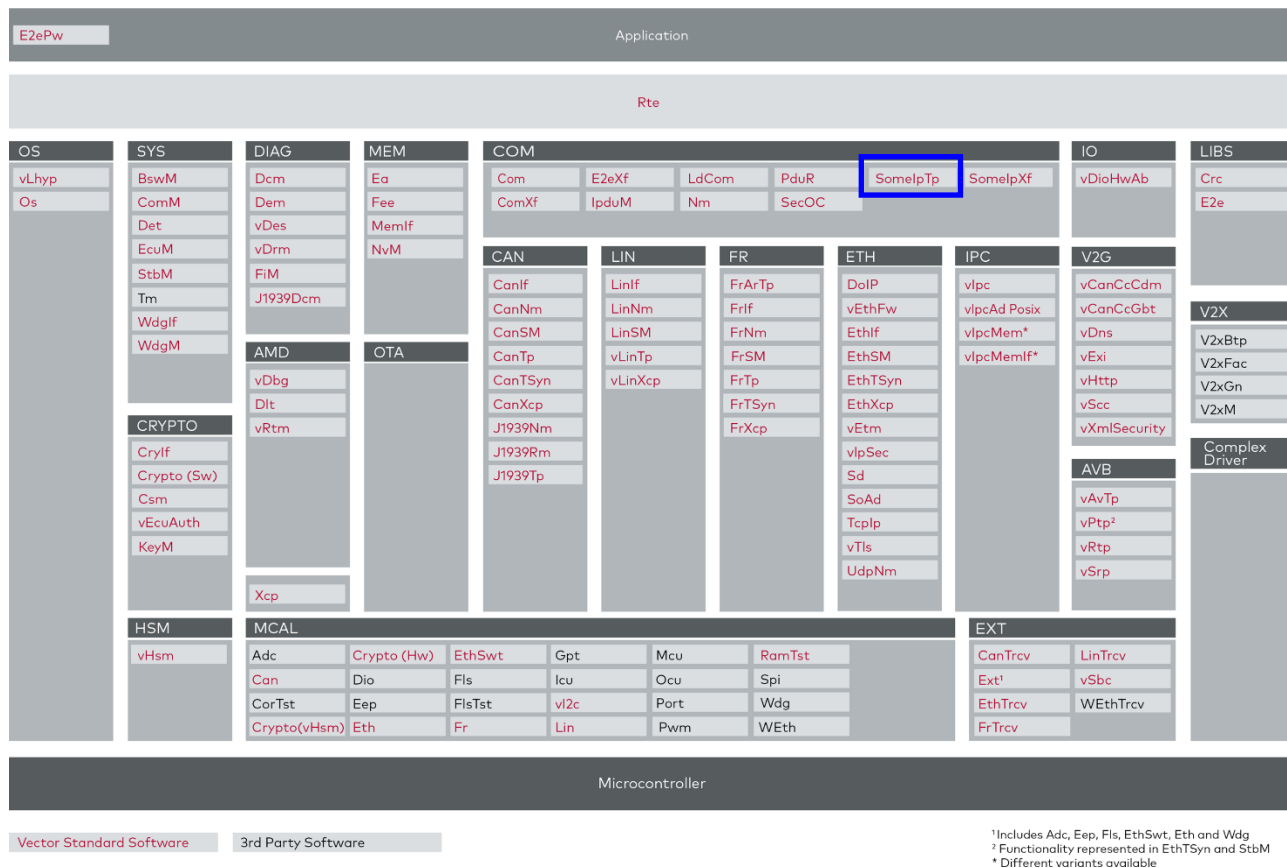


Figure 1-1 AUTOSAR 4.3 Architecture Overview

The next figure shows the interfaces to adjacent modules of the `SomelpTp`. These interfaces are described in chapter 4.

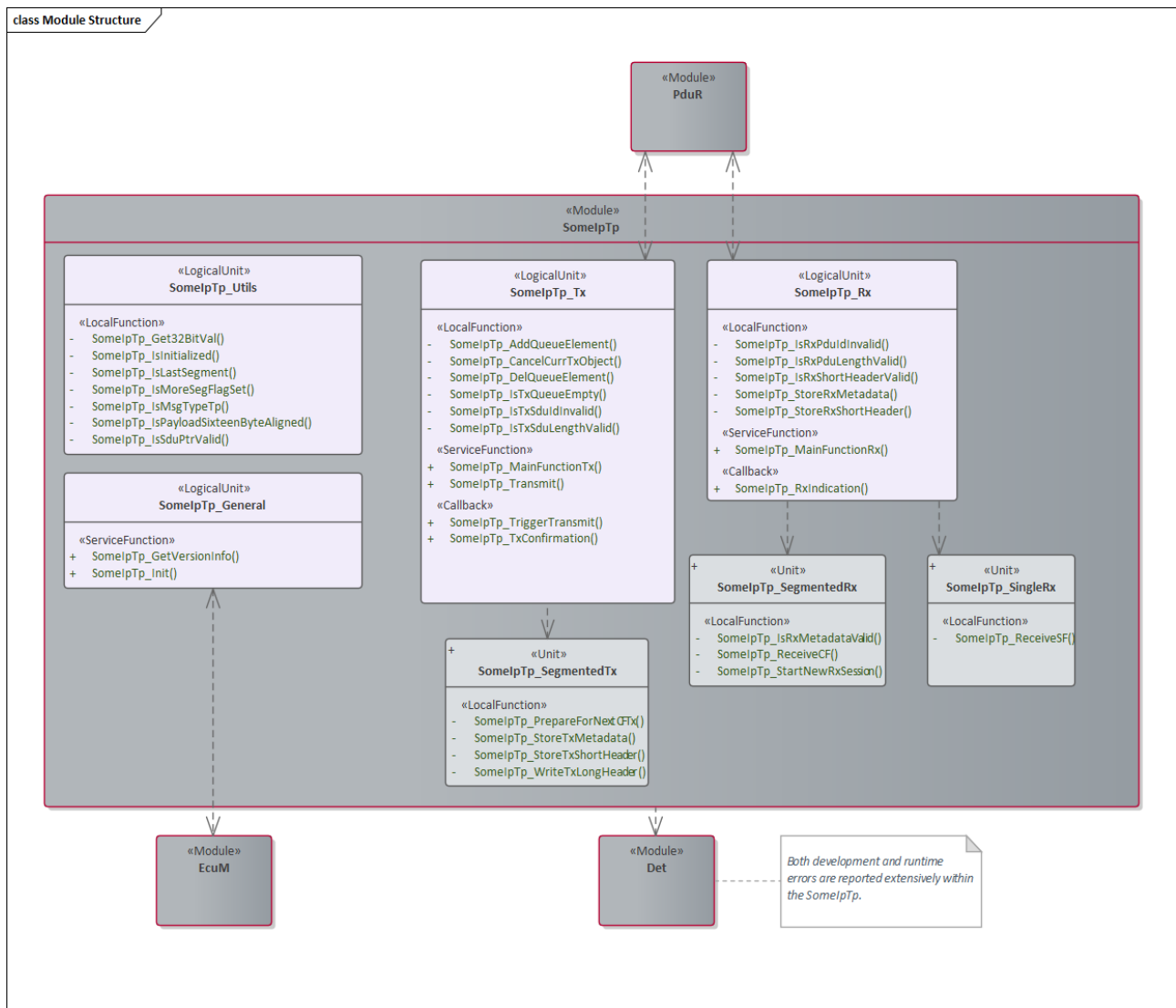


Figure 1-2 Interfaces to adjacent modules of the SomelpTp

2 Functional Description

2.1 Features

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

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

> Table 2-1 Supported AUTOSAR standard conform features

> Table 2-2 Not supported AUTOSAR standard conform features

Vector Informatik provides further SomelpTp functionality beyond the AUTOSAR standard. The corresponding features are listed in the table

> Table 2-3 Features provided beyond the AUTOSAR standard

The following features specified in [1] are supported:

Supported AUTOSAR Standard Conform Features
Segmented and unsegmented data transmission
Segmented and unsegmented reception
Supervision of timeouts
Detection of errors during segmentation and reassembly
Forwarding of metadata during transmission and reception

Table 2-1 Supported AUTOSAR standard conform features

2.1.1 Deviations

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

Not Supported AUTOSAR Standard Conform Features	ASR Version
Negative TX Confirmation is not supported. For more details refer Monitoring Transmission Confirmation timeout	4.3.1

Table 2-2 Not supported AUTOSAR standard conform features

2.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
Transmit queue
Monitoring Transmission Confirmation Timeout

Features Provided Beyond The AUTOSAR Standard

Burst Transmission of TP Segments

Table 2-3 Features provided beyond the AUTOSAR standard

2.1.2.1 Transmit Queue

If several TX SDUs are requested by the upper layer, they will be added to a queue. Supervision of timeouts and transmission of further segments of these SDUs is based on the order of elements stored in the Queue.

2.1.2.2 Monitoring Transmission Confirmation timeout

To be compatible with the other modules, `SomeIpTp_TxConfirmation()` is implemented only with the parameter `TxPduId`. Since negative TX Confirmation is currently not supported, if there is no call to `SomeIpTp_TxConfirmation()` within a specified time, the segmentation process for the corresponding TX SDU is aborted.

In the Vector `SomelpTp`, this timeout is configurable by the parameter `SomeIpTpTxConfirmationTimeoutTime` per `SomeIpTpChannel`.

2.1.2.3 Burst Transmission of TP Segments

This feature is implemented to increase the transmission speed of `SomelpTp` frames where the transmission of TP segments of the frames with separation time takes longer than required, Segmented messages will be transmitted in a burst in each main function cycle according to the configured burst size.

2.2 Initialization

The API `SomeIpTp_Init()` uses the given configuration set to initialize all global variables of `SomelpTp` and brings each SDU to state `Idle`, where segmentation and reassembly can be started.

2.3 States

After initialization, all the TX and RX SDUs are set to `Idle` (`SOMEIPTP_TX_STATUS_IDLE`, `SOMEIPTP_RX_STATUS_IDLE`). Segmentation or reassembly is possible only in this state.

2.4 Main Functions

The functions `SomeIpTp_MainfunctionTx()` and `SomeIpTp_MainfunctionRx()` perform all the tasks that should be done cyclically.

The timing parameters like `SomeIpTpSeparationTime` and `SomeIpTpRxTimeoutTime` are calculated based on the configured `SomeIpTpTxMainfunctionPeriod` and `SomeIpTpRxMainfunctionPeriod` respectively.

2.5 Error Handling

2.5.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [2], if development error reporting is enabled (i.e. pre-compile parameter `SOMEIPTP_DEV_ERROR_REPORT==STD_ON`).

If another module is used for development error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportError()`.

The reported `SomelpTp` ID is 177.

The reported service IDs identify the services which are described in 4.1. The following table presents the service IDs and the related services:

Service ID	Service
0x01	<code>SomelpTp_GetVersionInfo</code>
0x02	<code>SomelpTp_Init</code>
0x03	<code>SomelpTp_MainfunctionTx</code>
0x04	<code>SomelpTp_MainfunctionRx</code>
0x40	<code>SomelpTp_TxConfirmation</code>
0x41	<code>SomelpTp_TriggerTransmit</code>
0x42	<code>SomelpTp_RxIndication</code>
0x49	<code>SomelpTp_Transmit</code>

Table 2-4 Service IDs

The errors reported to DET are described in the following table:

Error Code		Description
0x01	<code>SOMEIPTP_E_NOTINIT</code>	<code>SomelpTp</code> not Initialized
0x02	<code>SOMEIPTP_E_PARAM_POINTER</code>	API called with wrong <code>PduInfo</code> pointer
0x03	<code>SOMEIPTP_E_PARAM</code>	API called with wrong parameters
0xFF	<code>SOMEIPTP_E_FATAL_ERROR</code>	Transmit Queue is full

Table 2-5 Development errors reported to DET

2.5.2 Runtime Error Reporting

By default, Runtime related errors are reported to the to the DET using the service `Det_ReportRuntimeError()` as specified in [2], if runtime error reporting is enabled (i.e. pre-compile parameter `SOMEIPTP_RUNTIME_ERROR_REPORT==STD_ON`).

If another module is used for runtime error reporting, the function prototype for reporting the error can be configured by the integrator, but must have the same signature as the service `Det_ReportRuntimeError()`.

The runtime errors reported to DET are described in the following table:

Error Code		Description
0x04	SOMEIPTP_E_MESSAGE_TYPE	TP Flag not set
0x05	SOMEIPTP_E_INCONSISTENT_SEQUENCE	Out of order segment received
0x06	SOMEIPTP_E_INCONSISTENT_HEADER	Different header compared to previous segments
0x07	SOMEIPTP_E_DISASSEMBLY_INTERRUPT	Segment received with incorrect alignment or insufficient buffer reported by upper layer
0x08	SOMEIPTP_E_ASSEMBLY_INTERRUPT	Requested payload not available by the upper layer
0x09	SOMEIPTP_E_INCONSISTENT_METADATA	Different metadata compared to previous segments

Table 2-6 Runtime errors reported to DET

3 Integration

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

3.1 Scope of Delivery

The delivery of the SomelpTp contains the files which are described in the chapters 3.1.1 and 3.1.2:

3.1.1 Static Files

File Name	Description
SomelpTp.c	Source file of SomelpTp
SomelpTp.h	Header file of SomelpTp
SomelpTp_Cbk.h	Header file with SomelpTp callback function prototypes
SomelpTp_Priv.h	Header file with internal SomelpTp definitions.

Table 3-1 Static files

3.1.2 Dynamic Files

The dynamic files are generated by the configuration tool [config tool].

File Name	Description
SomelpTp_Cfg.h	Header file for configuration parameters (e.g. Pre-compile switches)
SomelpTp_LCfg.c	Source file for generated source code
SomelpTp_LCfg.h	Header file for generated source code
SomelpTp_PBCfg.c	Header file for generated source code (post-build-time configurable parameters)
SomelpTp_PBCfg.h	Header file for generated source code (post-build-time configurable parameters)
SomelpTp_Types.h	Header file containing SomelpTp type definitions

Table 3-2 Generated files

3.1.3 User modifiable templates

File Name	Description
Cdd_SomelpTp.c	Example implementation of a fictive component located between the LdCom and the SWC, that allows the transmission and reception of byte streams bypassing the RTE. This file can be adapted as required.
Cdd_SomelpTp.h	Example header of the fictive component. This file can be adapted as required.

Table 3-3 User modifiable templates

3.2 Include Structure

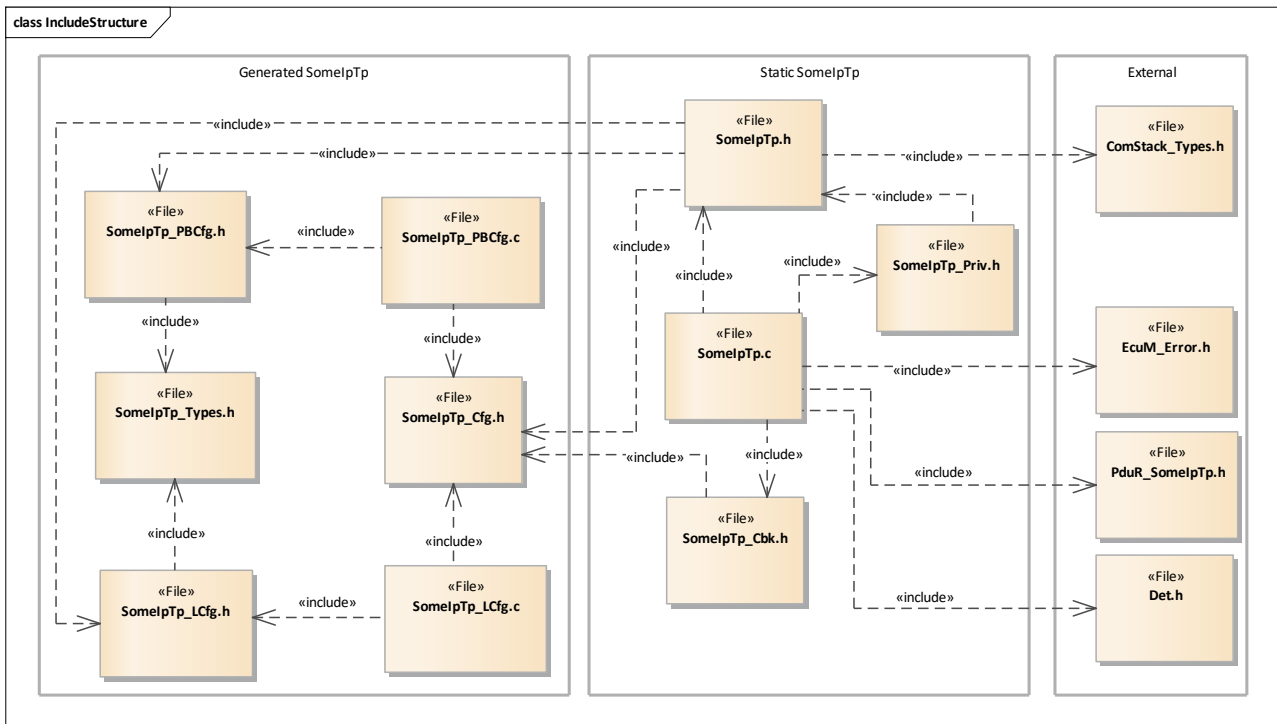


Figure 3-1 Include Structure



Note

The only files that need to be included by other modules and software components are `SomeIpTp.h` and `SomeIpTp_Cbk.h`. These files contain all definitions and inclusions required to use the `SomelPtp`.

3.3 Critical Sections

All services and callbacks for transmission, reception and state changes of `SomelPtp` may be called in interrupt or task level. Thus, a synchronization mechanism is implemented to guarantee data consistency.

The synchronization mechanism defined by AUTOSAR covers the entering and leaving of so called critical sections.

Relevant interrupt services in the `SomelPtp` context are interrupt services originated from physical bus events (Ethernet, CAN, FlexRay, LIN etc.) which may effect `SomelPtp` operation.

The `SomelPtp` defines the following exclusive area.

- > `SOMEIPTP_EXCLUSIVE_AREA_0` is used to protect Transmit Queue access and memory access to TX SDU information from interrupting calls to services and callbacks of `SomelPtp`.

For implementing this critical section it could be sufficient to

- > Disable all bus relevant interrupts related to calls of SomeIpTp APIs.
- > Disable all Ethernet bus relevant interrupts if all the modules calling SomeIpTp APIs are mapped to one task.

4 API Description

For an interfaces overview please see Figure 1-2.

4.1 Services provided by SomelpTp

4.1.1 SomelpTp_GetVersionInfo

Prototype	
void SomelpTp_GetVersionInfo(StdVersionInfoType *VersionInfo)	
Parameter	
Parameter	Pointer to store the version information.Parameter must not be NULL.
Return code	
Void	None
Functional Description	
Returns the version information.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > TASK ISR2 	

Table 4-1 SomelpTp_GetVersionInfo

4.1.2 SomelpTp_InitMemory

Prototype	
void SomelpTp_InitMemory(void)	
Parameter	
Void	None
Return code	
Void	none
Functional Description	
Initializes *_INIT_* variables. This is used in case they are not initialized by startup code.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is Non-reentrant. 	

Expected Caller Context
> TASK

Table 4-2 SomelpTp_InitMemory

4.1.3 SomelpTp_Init

Prototype	
void SomelpTp_Init(const SomeIpTp_ConfigType *SomeIpTp_ConfigPtr)	
Parameter	
SomeIpTp_ConfigPtr [in]	Configuration structure for initializing the module.
Return code	
Void	none
Functional Description	
This function initializes the module SomelpTp. It initializes all the variables and sets the module to initialized.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > Interrupts are disabled and SomelpTp_InitMemory has been called unless SomelpTp_ModuleInitialized is initialized by startup code. > This function is synchronous. > This function is non-reentrant. 	
Expected Caller Context	
> TASK	

Table 4-3 SomelpTp_Init

4.1.4 SomelpTp_Transmit

Prototype	
Std_ReturnType SomelpTp_Transmit(PduIdType TxPduId, const PduInfoType *PduInfoPtr)	
Parameter	
TxPduId [in]	Identifier of the PDU to be transmitted.
PduInfoPtr [in]	Contains the whole data's length plus the metadata's length at PduInfoPtr->SduLength, as well as a pointer to the whole data, followed by the metadata at PduInfoPtr->SduDataPtr.
Return code	
Std_ReturnType	E_OK Transmit request was accepted.
	E_NOT_OK Transmit request was not accepted.

Functional Description
Requests transmission of a PDU.
Particularities and Limitations
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant.
Expected Caller Context
> TASK ISR2

Table 4-4 SomelpTp_Transmit

4.1.5 SomelpTp_MainFunctionTx

Prototype	
void SomelpTp_MainFunctionTx(void)	
Parameter	
Void	none.
Return code	
void	none.
Functional Description	
Schedules transmission of consecutive segments and monitors TX Confirmation timeout	
Particularities and Limitations	
<ul style="list-style-type: none">> Service ID: see table 'Service IDs'> This function is synchronous.> This function is non-reentrant.	
Expected Caller Context	
<ul style="list-style-type: none">> TASK	

Table 4-5 SomelpTp_MainFunctionTx

4.1.6 SomelpTp_MainFunctionRx

Prototype	
void SomelpTp_MainFunctionRx(void)	
Parameter	
Void	none.
Return code	
void	none.

Functional Description
This function monitors RxTimeoutTime
Particularities and Limitations
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant.
Expected Caller Context
> TASK

Table 4-6 SomelpTp_MainFunctionRx

4.2 Services used by SomelpTp

In the following table services provided by other components, which are used by the SomelpTp are listed. For details about prototype and functionality refer to the documentation of the providing component.

Component	API
DET	Det_ReportError
DET	Det_ReportRuntimeError
PDUR	PduR_SomelpTpCopyRxData
PDUR	PduR_SomelpTpCopyTxData
PDUR	PduR_SomelpTpRxIndication
PDUR	PduR_SomelpTpStartOfReception
PDUR	PduR_SomelpTpTransmit
PDUR	PduR_SomelpTpTxConfirmation
ECUM	EcuM_BswErrorHook

Table 4-7 Services used by the SomelpTp

4.3 Callback Functions

This chapter describes the callback functions that are implemented by the SomelpTp and can be invoked by other modules. The prototypes of the callback functions are provided in the header file `SomeIpTp_Cbk.h` by the SomelpTp.

4.3.1 SomelpTp_TriggerTransmit

Prototype	
Std_ReturnType SomelpTp_TriggerTransmit (PduIdType TxPduId, PduInfoType* *PduInfoPtr)	
Parameter	
TxPduId [in]	ID of the PDU requested to be transmitted.

PduInfoPtr [in][out]	Contains a pointer to the buffer where the SDU data shall be copied. On return, the service will indicate the length of the data copied.
Return code	
Std_ReturnType	E_OK
	E_NOT_OK : SDU data has not been copied.
Functional Description	
Within this API, the called upper layer module shall check whether the available data fits into the buffer size reported by PduInfoPtr->SduLength. If it fits, it shall copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
> TASK ISR2	

Table 4-8 SomelpTp_TriggerTransmit

4.3.2 SomelpTp_TxConfirmation

Prototype	
void SomelpTp_TxConfirmation (PduIdType TxPduId)	
Parameter	
TxPduId [in]	ID of the PDU that has been transmitted.
Return code	
Void	none
Functional Description	
The lower layer communication interface module confirms the transmission of a PDU.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
> TASK ISR2	

Table 4-9 SomelpTp_TxConfirmation

4.3.3 SomelpTp_RxIndication

Prototype	
void SomelpTp_RxIndication (PduIdType RxPduId, const PduInfoType* PduInfoPtr)	
Parameter	
RxPduId [in]	ID of the received PDU.
PduInfoPtr [in]	Contains the length(SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU and metadata.
Return code	
Void	none
Functional Description	
Indication of a received PDU from a lower layer communication interface module.	
Particularities and Limitations	
<ul style="list-style-type: none"> > Service ID: see table 'Service IDs' > This function is synchronous. > This function is reentrant. 	
Expected Caller Context	
<ul style="list-style-type: none"> > TASK ISR2 	

Table 4-10 SomelpTp_RxIndication

4.4 Configurable Interfaces

None.

5 Configuration

The SomelpTp attributes can be configured with the following tool:

- > Configuration in DaVinci Configurator Pro 5

5.1 Configuration Variants

The SomelpTp supports the configuration variants

- > VARIANT-PRE-COMPILE
- > VARIANT-POST-BUILD-LOADABLE
- > VARIANT-POST-BUILD-SELECTABLE

The configuration classes of the SomelpTp parameters depend on the supported configuration variants. For their definitions please see the SomelpTp_bswmd.arxml file.

5.2 Additional configuration hints

5.2.1 SomelpTpRxTimeoutTime

The SomelpTp module starts the timer after the first segment is received and restarts after reception of intermediate segmented PDUs of an SDU and stops when the last segment is received.

To avoid unpredictable interruption, this configured time should be sufficient to receive the next segmented PDU of an SDU.

5.2.2 SomelpTpTxConfirmationTimeoutTime

The SomelpTp module starts the timer immediately after a PDU is requested for transmission to the lower layer communication module and waits for transmission confirmation until this period before aborting the disassembly process.

To avoid unpredictable interruption, this configured time should be sufficient to transmit and process the confirmation of a segmented PDU.

5.2.3 SomelpTpTxBurstSize

During the disassembly process, SomelpTp triggers consecutive segments in a burst of `SomelpTpTxBurstSize` size in each main function cycle without waiting for the separation time to expire. If the burst size is configured to default i.e. 1, separation time is maintained between the consecutive segments.

5.2.4 Sending and transmitting data without the RTE API

This guide is based on the provided templates listed in Table 3-3, and assumes that the routing paths between SoAd/SomelpTp and LdCom/SomelpTp are properly configured.

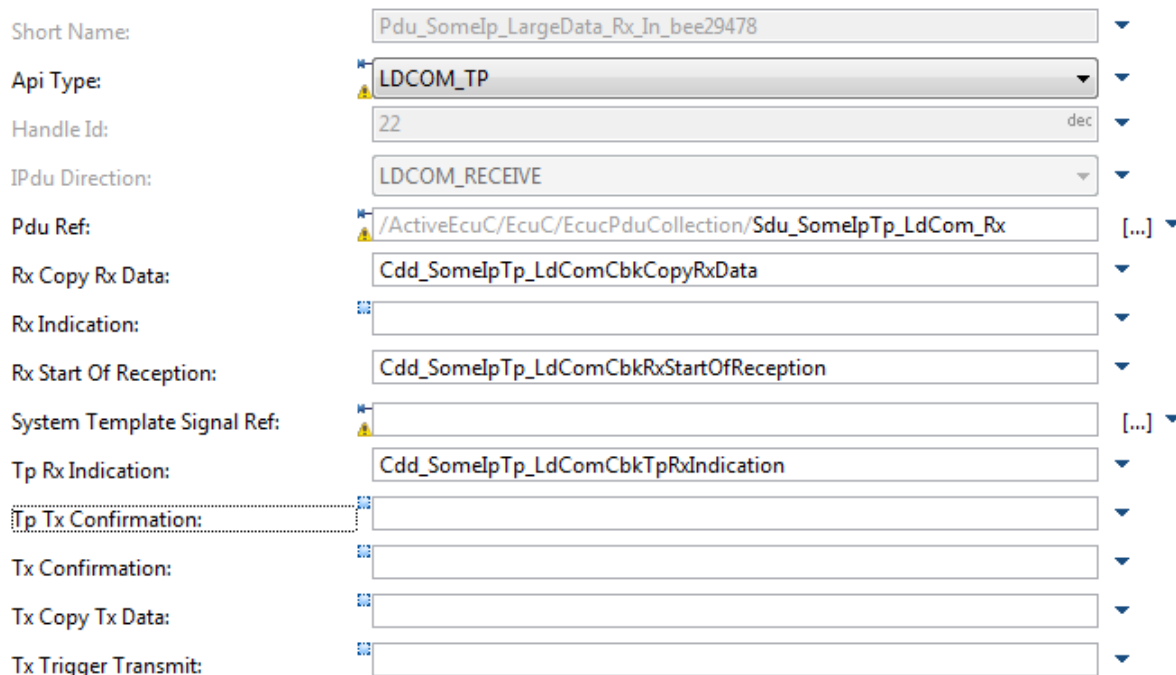
Make sure none of the LdCom's I-PDUs related to the SomeIpTp's SDUs have been mapped to the RTE, since this could trigger validation errors in the DaVinci Configurator and generation won't be possible.

5.2.4.1 Configuring the LdCom with the DaVinci Configurator

To receive data using the provided fictive component, the following parameters of the LdCom's Rx I-PDU related to the SomeIpTp's Rx SDU should be modified:

- > LdComApiType: Set to LDCOM_TP.
- > LdComRxCopyRxData: Set to the proper callback function. If the provided code is used, this would be Cdd_SomeIpTp_LdComCbKCopyRxData.
- > LdComRxStartOfReception: Set to the proper callback function. If the provided code is used, this would be Cdd_SomeIpTp_LdComCbKRxStartOfReception.
- > LdComTpRxIndication: Set to the proper callback function. If the provided code is used, this would be Cdd_SomeIpTp_LdComCbKTpRxIndication.

Figure 5-1 shows how the configuration should look like.



Short Name:	Pdu_SomeIp_LargeData_Rx_In_bee29478	▼
Api Type:	LDCOM_TP	▼
Handle Id:	22	dec ▼
IPdu Direction:	LDCOM_RECEIVE	▼
Pdu Ref:	/ActiveEcuC/EcuC/EcucPduCollection/Sdu_SomeIpTp_LdCom_Rx	[...] ▼
Rx Copy Rx Data:	Cdd_SomeIpTp_LdComCbKCopyRxData	▼
Rx Indication:		▼
Rx Start Of Reception:	Cdd_SomeIpTp_LdComCbKRxStartOfReception	▼
System Template Signal Ref:		[...] ▼
Tp Rx Indication:	Cdd_SomeIpTp_LdComCbKTpRxIndication	▼
Tp Tx Confirmation:		▼
Tx Confirmation:		▼
Tx Copy Tx Data:		▼
Tx Trigger Transmit:		▼

Figure 5-1 Configuration of the LdCom's I-PDU used during reception.

To transmit data using the provided fictive component, the following parameters of the LdCom's Tx I-PDU related to the SomeIpTp's Tx SDU should be modified:

- > LdComApiType: Set to LDCOM_TP.
- > LdComTpTxConfirmation: Set to the proper callback function. If the provided code is used, this would be Cdd_SomeIpTp_LdComCbKTpTxConfirmation.
- > LdComTxCopyTxData: Set to the proper callback function. If the provided code is used, this would be Cdd_SomeIpTp_LdComCbKCopyTxData.

Figure 5-2 shows how the configuration should look like.

Short Name:	Pdu_SomeIp_LargeData_Tx_Out_e2acea87	▼
Api Type:	LDCOM_TP	▼
Handle Id:	25	dec ▼
IPdu Direction:	LDCOM_SEND	▼
Pdu Ref:	/ActiveEcuC/EcuC/EcucPduCollection/Sdu_LdCom_SomeIpTp_Tx	[...] ▼
Rx Copy Rx Data:		▼
Rx Indication:		▼
Rx Start Of Reception:		▼
System Template Signal Ref:		[...] ▼
Tp Rx Indication:		▼
Tp Tx Confirmation:	Cdd_SomeIpTp_LdComCbKTPTxConfirmation	▼
Tx Confirmation:		▼
Tx Copy Tx Data:	Cdd_SomeIpTp_LdComCbKCopyTxData	▼
Tx Trigger Transmit:		▼

Figure 5-2 Configuration of the LdCom's I-PDU used during transmission.

Additionally, the header containing the declaration of the configured callback functions must be configured as an LdCom user config file (LdCom/LdComGeneral/LdComVersionInfoApi).

5.2.4.2 Adjusting the templates

The provided fictive component allows to transmit a stream of bytes over an LdCom Tx I-PDU. For this simple use case, the only mandatory change is to define the LdCom Tx I-PDU that shall be used for transmission in Cdd_SomeIpTp.h as follows:

```
#define CDD_SOMEIPTP_LDCOM_IPDU_TX LdComConf_LdComIPdu_Pdu_Symbolic_Name
```

LdCom_Cfg.h contains the symbolic names of the available LdCom I-PDUs.

Cdd_SomeIpTp.h also provides the array Cdd_SomeIpTp_xfHeader, which contains the SOME/IP header fields Request ID, Protocol Version, Interface Version, Message Type and Return Code of message to be sent. This fields can also be modified if required.

The comments in the templates provide more information.

5.2.4.3 Reading and Writing data from the SWC

Table 5-1 provides an overview of the functions provided by the fictive component that are intended to be used by the SWC. The functions are supposed to emulate the functionality provided by the RTE.

Function Name	Description
Cdd_SomeIpTp_Read	Reads the data received over the Rx I-PDU configured in 5.2.4.1.
Cdd_SomeIpTp_IsUpdated	Checks if the received data has been updated since the last read.
Cdd_SomeIpTp_Write	Writes data to be sent over the Tx I-PDU configured in 5.2.4.1.

Table 5-1 Functions intended to be used by the SWC.

The following example shows how the provided functions could be used within a runnable entity to receive, modify and send back a byte stream over the SomeIpTp.



Example

```
/* *****  
 * DON'T CHANGE COMMENT!  <<Start of runnable>>  DON'T CHANGE COMMENT!  
***** */  
  
#include "Cdd_SomeIpTp.h"  
  
if (Cdd_SomeIpTp_IsUpdated() == TRUE)  
{  
    uint8 dst[CDD_SOMEIPTP_MAX_SDU_LENGTH];  
    uint16 length = 0;  
  
    if (Cdd_SomeIpTp_Read(dst, &length) == RTE_E_OK)  
    {  
        uint16 i;  
  
        for (i = 0; i < length; i++)  
        {  
            dst[i]++;  
        }  
        Cdd_SomeIpTp_Write(dst, length);  
    }  
}  
/* *****  
 * DON'T CHANGE COMMENT!  <<End of runnable>>  DON'T CHANGE COMMENT!  
***** */
```

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
DaVinci Configurator Pro 5	Configuration and code generation tool for MICROSAR components

Table 6-1 Glossary

6.2 Abbreviations

Abbreviation	Description
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basis Software
DET	Development Error Tracer
HIS	Hersteller Initiative Software
ISR	Interrupt Service Routine
MICROSAR	Microcontroller Open System Architecture (the Vector AUTOSAR solution)
SRS	Software Requirement Specification
SWS	Software Specification
PduR	PDU Router
PDU	Protocol Data Unit
SDU	Service Data Unit
EcuM	ECU State Manager
ECU	Electronic Control Unit

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com