MICROSAR FrTp

Technical Reference

Version 2.00.04

Authors	Knut Winkelbach, Oliver Reineke
Status	Released

Document Information

History

Author	Date	Version	Remarks
Knut Winkelbach	2008-09-05	0.1	First version for Pre-Release.
Klaus Bergdolt	2008-09-16	0.2	Manual configuration when using ECUC files.
Knut Winkelbach	2009-01-26	1.0	Adapted to new features, explained limitations in more detail.
Knut Winkelbach	2009-03-28	1.1	Clarified interaction with FrIf. Removed feature overview.
Knut Winkelbach	2009-06-29	1.2	Added description of new features for code size optimization.
Knut Winkelbach	2009-07-07	1.3	Rework after review.
Knut Winkelbach	2009-11-20	1.4	Described new code optimizations in conjunction with their use cases. Adapted erroneous prototype of FrTp_Transmit() Extended description of GENy configurable options.
Knut Winkelbach	2009-12-01	1.5	Rework after review.
Knut Winkelbach	2010-04-22	1.6	Straightened description of "Multi Purpose Tp" Added error detection hints and support for "Multiple Identity" of Ecus.. Added description of feature "FrTp_ChangeParameterRequest"_(extended parameter change) and "FrTp_ReadParameterRequest".
Knut Winkelbach	2010-05-07	1.7	Rework after review.
Knut Winkelbach	2010-05-21	1.8	Corrected prototype of FrTp_ReadParameterRequest.
Knut Winkelbach	2010-08-20	1.9	Added description of optimizations for GW-operation.
Knut Winkelbach	2010-08-26	1.10	Rework after review.
Knut Winkelbach	2011-02-07	1.11	Clarified FrIf_Transmit(), FrTp_TriggerTransmit() and description of E_PENDING Optimizations for faster Rx- and Tx-Buffer-Retrieval now controlled by 'FrTp Disable Fast Buf Retrieval' instead of 'Have Fast Segm Stf Seg Rx:' and 'Have Fast Unsegm Stf Unseg Rx'.
Knut Winkelbach	2011-02-21	1.12	Corrected description of 'FrTp Disable Fast Buf Retrieval'.
Knut Winkelbach	2011-05-23	1.13	Corrected revision history (aligned version with ALM tool).

Knut Winkelbach	2011-07-27	1.14	Removed description of FrIf_Transmit return code E_PENDING. Removed feature providing AUTOSAR 4 prototypes & corrected all features depending on the feature removed. Overworked chapter Multi Identity (new feature Multi Config added).
Knut Winkelbach	2012-01-11	1.15	Added description of FrTp_CancelReceive, updated description of FrTp_CancelTransmit, FrTp_ChangeParameter, removed CancelReason Removed section “Configuration with *.GNY project files” because configuration with FIBEX files is no longer supported by GENy. Overworked section “Configuration” and “Important Hints” as a consequence of this.
Knut Winkelbach	2012-07-10	1.16	Corrected “Contents”.
Knut Winkelbach	2013-02-21	1.17	Update to single source concept to be able to generate separate AUTOSAR 3 / 4 documents.
Knut Winkelbach	2013-05-17	1.18	No support of Acknowledge or Acknowledge and Retry.
Knut Winkelbach	2014-01-20	1.19	1. No ChangeParameter Confirmation Callback 2. AUTOSAR-compliant prototype of FrTp_ChangeParameter 3. FrTp_Shutdown cancelling all transfers optionally
Knut Winkelbach	2014-05-21	1.20	AUTOSAR 4.1.2 compliant PduR-API (Replacement of NotifResultType by Std_ReturnType) Check separation cycles in FrTp_ChangeParameter() vs. FrTpTimeoutCr.
Knut Winkelbach	2014-11-06	2.00.00	Separate development of AUTOSAR4 version. Support of postbuild selectable configurations. Support of runtime measurement.
Knut Winkelbach	2015-08-14	2.00.01	Added link to Technical Reference “Postbuild Selectable”, introduced new critical section.
Knut Winkelbach	2015-12-17	2.00.02	Updated Deviations Chapter.
Knut Winkelbach	2016-08-04	2.00.03	Re-creation based on newest Template.
Knut Winkelbach	2016-11-30	2.00.04	Added hint regarding unsupported config. variant “no Tx-Pdu-Pools at all”.

Referenced Documents

No.	Source	Title	Version
[1]	AUTOSAR	ISO_IS_10681-2_(E).doc	2009-12-10
[2]	AUTOSAR	AUTOSAR_SWS_FlexRayISOTransportLayer.pdf	V4.0.0
[3]	AUTOSAR	AUTOSAR_SWS_DevelopmentErrorTracer.pdf	V3.2.0

[4]	AUTOSAR	AUTOSAR_SWS_PDURouter.pdf	V3.2.0
[5]	Vector	TechnicalReference_Asr_EcuM.pdf	See delivery
[6]	Vector	TechnicalReference_Asr_Dbg.pdf	See delivery
[7]	Vector	TechnicalReference_PostBuildLoadable.pdf	See delivery
[8]	Vector	TechnicalReference_IdentityManager.pdf	See delivery
[9]	Vector	TechnicalReference_Asr4Rtm.pdf	See delivery
[10]	Vector	TechnicalReference_PostBuildSelectable.pdf	See delivery
[11]	AUTOSAR	AUTOSAR_TR_BSWModuleList.pdf	V1.6.0
[12]	AUTOSAR	AUTOSAR_SWS_CompilerAbstraction.pdf	V3.2.0
[13]	Vector	AN-ISC-8-1140_FrIf_JLE_Configuration.pdf	See delivery

Scope of the Document:

This technical reference describes the general use of the FrTp 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

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
FRTP_SW_MAJOR_VERSION: 2 FRTP_SW_MINOR_VERSION: 4	SafeBSW

Table 1-1 Component history

2 Introduction

This document describes the functionality, API and configuration of the AUTOSAR BSW module FrTp as specified in [2] and [2].

Supported AUTOSAR Release*:	4.0.3 (with deviations), 4.2.1	
Supported Configuration Variants:	pre-compile, link-time, post-build	
Vendor ID:	FrTp_VENDOR_ID	30 decimal (= Vector-Informatik, according to HIS)
Module ID:	FrTp_MODULE_ID	36 decimal (according to ref. [11])

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

The FrTp component implements a transport protocol according to ISO10681-2 to be used in AUTOSAR communication stacks.

2.1 Architecture Overview

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

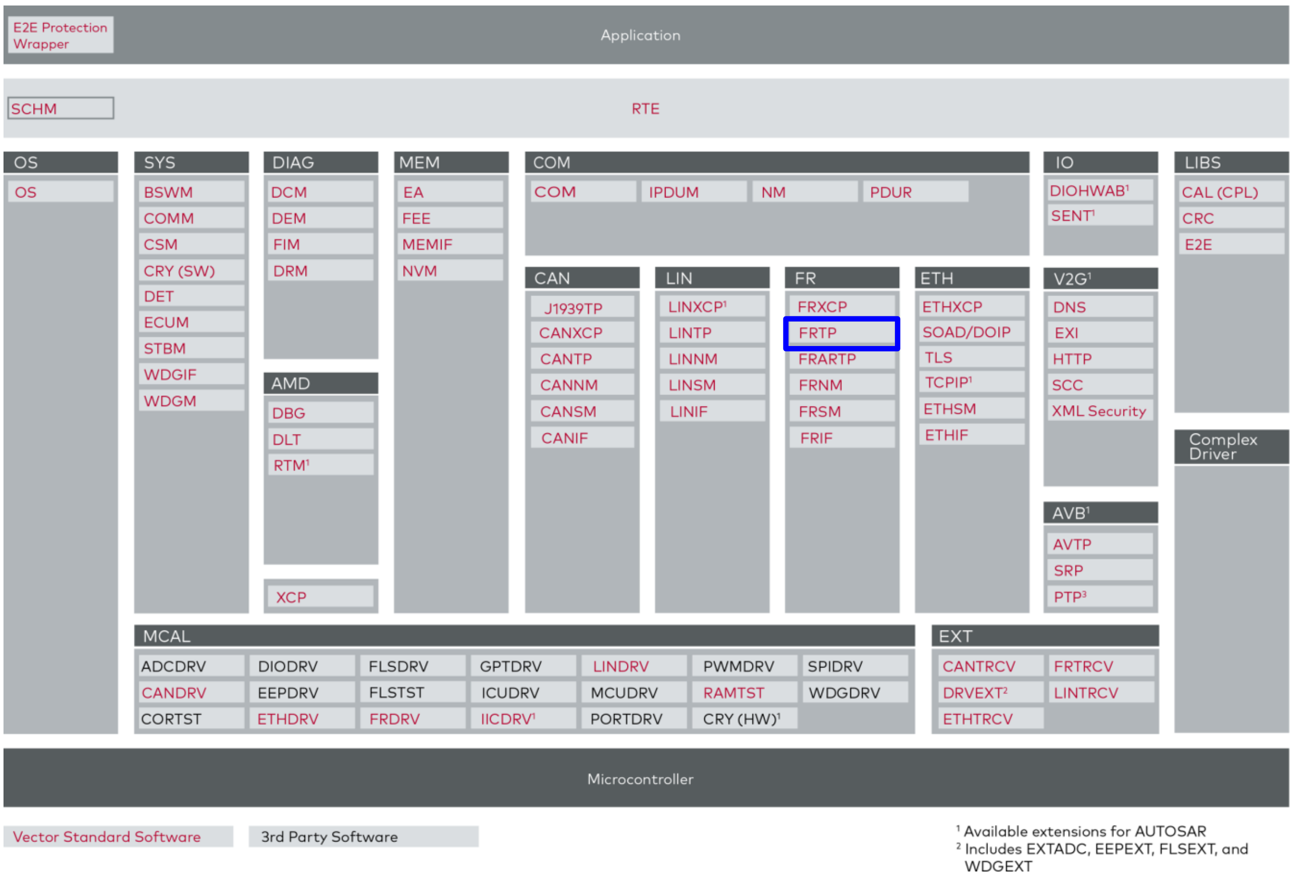

Figure 2-1 AUTOSAR 4.x Architecture Overview

The next figure shows the interfaces to adjacent modules of the FrTp. These interfaces are described in chapter 5.

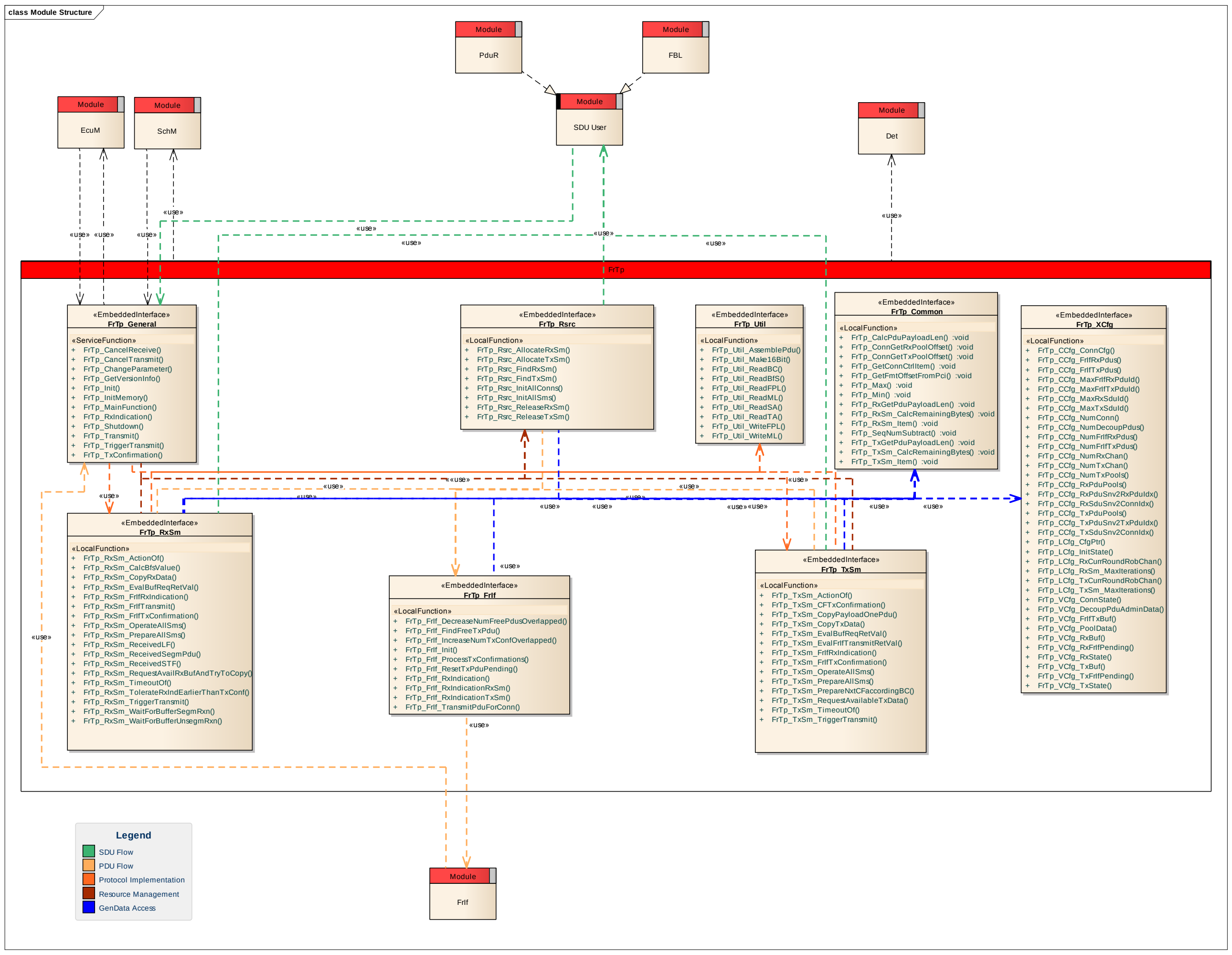
class Module Structure
Module
PduR
Module
FBL
Module
SDU User
Module
EcuM
Module
SchM
Module
Det
«use»
FrTp
«EmbeddedInterface»
FrTp_General
«ServiceFunction»
+ FrTp_CancelReceive()
+ FrTp_CancelTransmit()
+ FrTp_ChangeParameter()
+ FrTp_GetVersionInfo()
+ FrTp_Init()
+ FrTp_InitMemory()
+ FrTp_MainFunction()
+ FrTp_RxIndication()
+ FrTp_Shutdown()
+ FrTp_Transmit()
+ FrTp_TriggerTransmit()
+ FrTp_TxConfirmation()
«EmbeddedInterface»
FrTp_Rsrc
«LocalFunction»
+ FrTp_Rsrc_AllocateRxSm()
+ FrTp_Rsrc_AllocateTxSm()
+ FrTp_Rsrc_FindRxSm()
+ FrTp_Rsrc_FindTxSm()
+ FrTp_Rsrc_InitAllConns()
+ FrTp_Rsrc_InitAllSms()
+ FrTp_Rsrc_ReleaseRxSm()
+ FrTp_Rsrc_ReleaseTxSm()
«EmbeddedInterface»
FrTp_Util
«LocalFunction»
+ FrTp_Util_AssemblePdu()
+ FrTp_Util_Make16Bit()
+ FrTp_Util_ReadBC()
+ FrTp_Util_ReadBfS()
+ FrTp_Util_ReadFPL()
+ FrTp_Util_ReadML()
+ FrTp_Util_ReadSA()
+ FrTp_Util_ReadTA()
+ FrTp_Util_WriteFPL()
+ FrTp_Util_WriteML()
«EmbeddedInterface»
FrTp_Common
«LocalFunction»
+ FrTp_CalcPduPayloadLen() :void
+ FrTp_ConnGetRxPoolOffset() :void
+ FrTp_ConnGetTxPoolOffset() :void
+ FrTp_GetConnCtrlItem() :void
+ FrTp_GetFmtOffsetFromPci() :void
+ FrTp_Max() :void
+ FrTp_Min() :void
+ FrTp_RxGetPduPayloadLen() :void
+ FrTp_RxSm_CalcRemainingBytes() :void
+ FrTp_RxSm_Item() :void
+ FrTp_SeqNumSubtract() :void
+ FrTp_TxGetPduPayloadLen() :void
+ FrTp_TxSm_CalcRemainingBytes() :void
+ FrTp_TxSm_Item() :void
«EmbeddedInterface»
FrTp_XCfg
«LocalFunction»
+ FrTp_CCfg_ConnCfg()
+ FrTp_CCfg_FrIfRxPdus()
+ FrTp_CCfg_FrIfTxPdus()
+ FrTp_CCfg_MaxFrIfRxPduId()
+ FrTp_CCfg_MaxFrIfTxPduId()
+ FrTp_CCfg_MaxRxSduId()
+ FrTp_CCfg_MaxTxSduId()
+ FrTp_CCfg_NumConn()
+ FrTp_CCfg_NumDecoupPdus()
+ FrTp_CCfg_NumFrIfRxPdus()
+ FrTp_CCfg_NumFrIfTxPdus()
+ FrTp_CCfg_NumRxChan()
+ FrTp_CCfg_NumTxChan()
+ FrTp_CCfg_NumTxPools()
+ FrTp_CCfg_RxPduPools()
+ FrTp_CCfg_RxPduSnv2RxPduIdx()
+ FrTp_CCfg_RxSduSnv2ConnIdx()
+ FrTp_CCfg_TxPduPools()
+ FrTp_CCfg_TxPduSnv2TxPduIdx()
+ FrTp_CCfg_TxSduSnv2ConnIdx()
+ FrTp_LCfg_CfgPtr()
+ FrTp_LCfg_InitState()
+ FrTp_LCfg_RxCurrRoundRobChan()
+ FrTp_LCfg_RxSm_MaxIterations()
+ FrTp_LCfg_TxCurrRoundRobChan()
+ FrTp_LCfg_TxSm_MaxIterations()
+ FrTp_VCfg_ConnState()
+ FrTp_VCfg_DecoupPduAdminData()
+ FrTp_VCfg_FrIfTxBuf()
+ FrTp_VCfg_PoolData()
+ FrTp_VCfg_RxBuf()
+ FrTp_VCfg_RxFrIfPending()
+ FrTp_VCfg_RxState()
+ FrTp_VCfg_TxBuf()
+ FrTp_VCfg_TxFrIfPending()
+ FrTp_VCfg_TxState()
«EmbeddedInterface»
FrTp_RxSm
«LocalFunction»
+ FrTp_RxSm_ActionOf()
+ FrTp_RxSm_CalcBfsValue()
+ FrTp_RxSm_CopyRxData()
+ FrTp_RxSm_EvalBufReqRetVal()
+ FrTp_RxSm_FrIfRxIndication()
+ FrTp_RxSm_FrIfTransmit()
+ FrTp_RxSm_FrIfTxConfirmation()
+ FrTp_RxSm_OperateAllSms()
+ FrTp_RxSm_PrepareAllSms()
+ FrTp_RxSm_ReceivedLF()
+ FrTp_RxSm_ReceivedSegmPdu()
+ FrTp_RxSm_ReceivedSTF()
+ FrTp_RxSm_RequestAvailRxBufAndTryToCopy()
+ FrTp_RxSm_TimeoutOf()
+ FrTp_RxSm_TolerateRxIndEarlierThanTxConf()
+ FrTp_RxSm_TriggerTransmit()
+ FrTp_RxSm_WaitForBufferSegmRxn()
+ FrTp_RxSm_WaitForBufferUnsegmRxn()
«EmbeddedInterface»
FrTp_FrIf
«LocalFunction»
+ FrTp_FrIf_DecreaseNumFreePdusOverlapped()
+ FrTp_FrIf_FindFreeTxPdu()
+ FrTp_FrIf_IncreaseNumTxConfOverlapped()
+ FrTp_FrIf_Init()
+ FrTp_FrIf_ProcessTxConfirmations()
+ FrTp_FrIf_ResetTxPduPending()
+ FrTp_FrIf_RxIndication()
+ FrTp_FrIf_RxIndicationRxSm()
+ FrTp_FrIf_RxIndicationTxSm()
+ FrTp_FrIf_TransmitPduForConn()
«EmbeddedInterface»
FrTp_TxSm
«LocalFunction»
+ FrTp_TxSm_ActionOf()
+ FrTp_TxSm_CFTxConfirmation()
+ FrTp_TxSm_CopyPayloadOnePdu()
+ FrTp_TxSm_CopyTxData()
+ FrTp_TxSm_EvalBufReqRetVal()
+ FrTp_TxSm_EvalFrIfTransmitRetVal()
+ FrTp_TxSm_FrIfRxIndication()
+ FrTp_TxSm_FrIfTxConfirmation()
+ FrTp_TxSm_OperateAllSms()
+ FrTp_TxSm_PrepareAllSms()
+ FrTp_TxSm_PrepareNxtCFaccordingBC()
+ FrTp_TxSm_RequestAvailableTxData()
+ FrTp_TxSm_TimeoutOf()
+ FrTp_TxSm_TriggerTransmit()
Module
FrIf
Legend
🟩 SDU Flow
🟧 PDU Flow
🟧 Protocol Implementation
🟫 Resource Management
🟦 GenData Access

Figure 2-2 Interfaces to adjacent modules of the FrTp

3 Functional Description

3.1 Features

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

The AUTOSAR standard functionality is specified in [2], 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

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

> Table 3-3 Features provided beyond the AUTOSAR standard

The following features specified in [2] are supported:

Supported AUTOSAR Standard Conform Features
AUTOSAR Debugging: Internal variables can be accessed through the Vector MICROSAR DBG module
Postbuild Loadable: Configuration parameters can be changed and new connections can be added at post-build time.
Postbuild Selectable: Support of multiple ECU variants; the currently active variant is selected during initialization.
AUTOSAR Runtime Measurement: Runtime of functions FrTp_TxConfirmation, FrTp_RxIndication, FrTp_TriggerTransmit, FrTp_MainFunction can be measured

Table 3-1 Supported AUTOSAR standard conform features

3.1.1 Deviations

The following features are not supported or deviate from document [2]:

Category	Description	Version
Functional	"Acknowledgement and Retry" is not supported.	4.0.3
Functional	"FrIf Retry" i.e. retry of calls to FrIf_Transmit() after FrIf errors is not supported.	4.0.3
Functional	The FrTp does not implement the buffer handling of AUTOSAR SWS FrTp 4.0.3 (Version 4.0.3 specifies using parameters FrTpTimeBuffer and FrTpMaxFcWait as a timing basis for data exchange with the PduR. Instead the FrTp tries to copy Rx- or Tx-data for a duration defined by parameters FrTpTimeBr and FrTpTimeCs as specified by version 4.2.1).	4.0.3 < 4.2.1
Config	Parameter FrTpMaxFCWait of container FrTpConnectionControl: The full range of values: [0..255] of that parameter is not supported (Instead the FrTp supports a range of [0..254]).	4.0.3

Category	Description	Version
Config	Parameter FrTpTimeBuffer of container FrTpConnectionControl: This parameter is ignored (Instead the FrTp supports the asking for Tx-data or Rx-buffer at each call to FrTp_MainFunction() as described in AUTOSAR SWS FrTp 4.2.1).	4.0.3
Config	Parameter FrTpMaxFrIf of container FrTpConnectionControl: This parameter is ignored because the related feature is not implemented. In addition the full range of values: [0..255] of that parameter is not supported (Instead the FrTp supports a range of [1..255]).	4.0.3
Config	FrTpTxPdu The maximum number of containers of type FrTpTxPdu is not supported (Instead the FrTp supports 254 Tx-Pdus).	4.0.3
Config	FrTpTxPduPool The maximum number of containers of type FrTpTxPduPool is not supported (Instead the FrTp supports 254 Tx-Pdu-Pools).	4.0.3
Config	FrTpRuntime The maximum number of runtime ressources of type FrTpChannel is not supported (Instead the FrTp supports 254 FrTpChannels).	4.0.3

Table 3-2 Not supported AUTOSAR standard conform features

3.1.2 Additions/ Extensions

The following features are provided beyond the AUTOSAR standard:

Features Provided Beyond The AUTOSAR Standard
n/a

Table 3-3 Features provided beyond the AUTOSAR standard

3.2 Error Handling

3.2.1 Development Error Reporting

By default, development errors are reported to the DET using the service `Det_ReportError()` as specified in [3], if development error reporting is enabled (i.e. pre-compile parameter `FrTp_DEV_ERROR_DETECT==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 FrTp ID is 36.

The reported service IDs identify the services which are described in 5.2. The following table presents the service IDs and the related services:

Service ID	Service
0x00	FrTp_Init_ServiceId

Service ID	Service
0x01	FrTp_Shutdown_ServiceId
0x02	FrTp_Transmit_ServiceId
0x03	FrTp_CancelTransmit_ServiceId (#if (FRTP_HAVE_TC == STD_ON))
0x04	FrTp_ChangeParameter_ServiceId
0x08	FrTp_CancelReceive_ServiceId
0x10	FrTp_MainFunction_ServiceId
0x27	FrTp_GetVersionInfo_ServiceId
0x40	FrTp_TxConfirmation_ServiceId
0x41	FrTp_TriggerTransmit_ServiceId
0x42	FrTp_RxIndication_ServiceId
0x81	FrTp_ReadParameter_ServiceId

Table 3-4 Service IDs

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

Error Code	Description
0x01	FRTP_E_NOT_INIT
0x02	FRTP_E_NULL_PTR
0x03	FRTP_E_INVALID_PDU_SDU_ID
0x04	FRTP_E_INVALID_PARAMETER (FRTP_WRONG_PARAM_VAL)
0x05	FRTP_E_SEG_ERROR
0x06	FRTP_E_UMSG_LENGTH_ERROR
0x07	FRTP_E_NO_CHANNEL
0xff	FRTP_E_NO_ERROR

Table 3-5 Errors reported to DET

4 Integration

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

4.1 Scope of Delivery

The delivery of the FrTp contains the files which are described in the chapters 4.1.1 and 4.1.2:

4.1.1 Static Files

File Name	Description
FrTp.c, FrTp_Rscr.c, FrTp_FrIf.c, FrTp_RxSm.c, FrTp_TxSm.c, FrTp_Util.c	These are the source files of the FrTp
FrTp_Types.h	This header file defines the data types used by the FrTp
FrTp_Common.h	This is an internal header file of the FrTp
FrTp_Lcfg.h	This header file declares the Link-Time configuration in case the variant Link-Time-configuration is selected
FrTp_PBcfg.h	This header file declares the PB configuration in case the variant PB is selected
FrTp_Cbk.h	This header declares callback functions.
FrTp.h	This header declares the FrTp function prototypes

Table 4-1 Static files

4.1.2 Dynamic Files

The dynamic files are generated by the configuration tool.

File Name	Description
FrTp_PBcfg.c	This file contains generated data defining the PB configuration in case the variant PB is selected
FrTp_Lcfg.c	This file contains generated data defining the Link-Time configuration in case the variant Link-Time-configuration is selected
FrTp_cfg.h	This file is a generated header file defining the FrTp-specific settings of the FrTp
FrTp_GlobCfg .h	This file is a generated header file declaring the global configuration struct of the FrTp.

Table 4-2 Generated files

4.2 Critical Sections

> The FrTp uses the critical section FRTP_EXCLUSIVE_AREA_0.

- The purpose of this critical section is to protect the rx- and tx-statemachines from inconsistencies during runtime.
- The critical section FRTP_EXCLUSIVE_AREA_0 is used by all API functions of the FrTp. In detail these are:
  - FrTp_MainFunction(): It is called synchronous to the FlexRay-bus, once each FlexRay-cycle. The locking time is long.

    In detail its runtime expands with the number of active TP message transfers and as the runtime of the functions PduR_FrTpCopyTxData() and PduR_FrTpCopyRxData() expands. Seldom, also calls to PduR_FrTpTxConfirmation() and PduR_FrTpRxIndication() are done.
  - FrTp_RxIndication(): The locking time is short.

    This function incorporates callouts to the PduR. Its runtime expands, as the runtime of the functions PduR_FrTpStartOfReception() and PduR_FrTpCopyRxData() expands. Seldom, also a call to PduR_FrTpRxIndication() is done.
  - FrTp_TxConfirmation():The locking time is short.

    This function incorporates callouts to the PduR. Its runtime expands, as the runtime of the functions PduR_CopyTxData() expands. Seldom, also a call to PduR_FrTpTxConfirmation() is done.
  - FrTp_TriggerTransmit():The locking time is short.

    This function incorporates callouts to the PduR. Its runtime expands, as the runtime of the functions PduR_CopyTxData() expands.
  - FrTp_Transmit():The locking time is short.

    This function solely allocates a tx-statemachine.

Note
Implementation of the critical section

By default the demo application that is delivered with your software locks all interrupts of the processor in case the "Enter" macro / function is called and releases all interrupts of the processor in case the "Exit" macro is called:

It is recommended to define the exclusive area in a way that it locks the communication interrupts of the bus systems affected. These depend on the use case the FrTp is used in:

For use case “Standard Ecu” the critical sections should lock FlexRay-Timer interrupt because this interrupt drives the FlexRay-interfaces JobListExecution (JLE) which issues the FrTp-related communication events FrTp_RxIndication, FrTp_TxConfirmation and FrTp_TriggerTransmit.

For use case “Gateway Ecu” the critical section should lock the interrupts of all bus-systems that affect the FrTp, e.g:

In case there are CanTp-FrTp-Routings configured then the interrupts that lead to calls of CanTp_RxIndication, CanTp_TxConfirmation should also be locked (in addition to FlexRay-timer-interrupts).

In case there are LinTp-FrTp-Routings configured then the interrupts that lead to calls of LinTp_RxIndication, LinTp_TxConfirmation should also be locked (in addition to FlexRay-timer-interrupts).

This assures that no rx- or tx-events from busses that affect FrTp-communication bring the FrTp statemachine into an inconsistent state.

4.3 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 of the FrTp, and illustrates their assignment among each other.

Compiler Abstraction Definitions Memory Mapping Sections	FRTP_CODE	FRTP_CONST	FRTP_APPL_DATA	FRTP_VAR_NOINIT	FRTP_VAR_ZERO_INIT	FRTP_VAR_PBCFG	FRTP_PBCFG	FRTP_PBCFG_ROOT
`FRTP_START_SEC_CONST_32BIT`		■						
`FRTP_START_SEC_CONST_16BIT`		■						
`FRTP_START_SEC_CONST_8BIT`		■						
`FRTP_START_SEC_PBCFG_ROOT`		■						

FRTP_START_SEC_CONST_UNSPECIFIED		■						
FRTP_START_SEC_PBCFG						■	■	
FRTP_START_SEC_VAR_ZERO_INIT_UNSPECIFIED					■	■		
FRTP_START_SEC_VAR_NOINIT_UNSPECIFIED				■		■		
FRTP_START_SEC_VAR_NOINIT_32BIT				■		■		
FRTP_START_SEC_VAR_NOINIT_16BIT				■		■		
FRTP_START_SEC_VAR_NOINIT_8BIT				■		■		
FRTP_START_SEC_CODE	■							

Table 4-3 Compiler abstraction and memory mapping

Please see the document [2] for details about how these definitions are being used.

4.4 Compatibility Checks

The FrTp component conducts a compatibility check during initialization in case the configuration variant is “postbuild”. A failure of this check is caused by generated data that does not fit to the FrTp implementation. In this case the function EcuM_BswErrorHook is called with error code “ECUM_BSWERROR_MAGICNUMBER”.

4.5 Steps to Start Up

In case of problems during starting up the FrTp, please read the following two chapters and check the facts described there.

4.5.1 Checklist

These steps are the "must haves" that have to be fulfilled in order to get the FrTp "up and running":

Caution
“Must Haves” to get the FrTp “up and running”

> In case no reception or transmission can be done using the FrTp the DET should be set active for the FrTp in the configuration tool.

> The functions FrTp_InitMemory() and FrTp_Init() have to be called before operation of the FrTp.

> The FrTp_MainFunction() has to be called:

  > once per FlexRay cycle and also

  > synchronous to the FlexRay time.

  > before or after the time of transmission or reception of the FrIf-Pdus used by the FrTp

> It is **not** possible to mix transmission types (immediate/decoupled) of FrIf-Tx-Pdus that are contained within one Tx-Pdu-Pool.

> It is **mandatory** to configure **at least one** Tx-Pdu-Pool, even though there might be (currently unsupported) use cases where the DCM uses the FrIf_Transmit() API directly (e.g. the so called “periodic transmission” used by some OEMs).

4.5.2 Transmission

In case of problems transmitting a FrTp message please check that these facts are fulfilled:

1. FrTp_Transmit() should return E_OK (‘0’).
2. During the next scheduling of the FrTp_MainFunction() the FrTp should request the upper layer (typically the “PduR”) to copy the Tx-data into the TP buffer by a call to PduR_FrTpCopyTxData().
3. After the successful provision of Tx-data in that function the transmission should begin. You can verify the start of the transmission of TP-Tx-frames by assuring that the function FrIf_Transmit() is called by the FrTp and the functions return value is E_OK..
4. After that (i.e. after the successfull call to FrTp_TriggerTransmit for decoupled Pdus) the function FrTp_TxConfirmation() is called by the FrIf-component that is used. There is only one location in FrTp.c where FrIf_Transmit() is called.
5. Each call to FrIf_Transmit() of the FrTp should be acknowledged by a call to FrTp_TxConfirmation() done by the FrIf component.
6. In any case the callback <Owner>_FrTpTxConfirmation() should be called after the transmission request via FrTp_Transmit() was accepted by the FrTp by returning E_OK. The notification-code given to <Owner>_FrTpTxConfirmation() will provide additional information about success or failure of the transmission.

Please refer to document [2] on how to interpret parameters and notification-codes of the functions mentioned.

4.5.3 Reception

In case of problems receiving a FrTp message, please check that these facts are fulfilled:

1. After the first FrTp -frame has been “detected” on the bus by an appropriate analyser-tool, the function FrTp_RxIndication() should be called by the FrIf-component that is used.
2. During the next scheduling of the FrTp_MainFunction() the FrTp should indicate to the upper layer that new Rx-data is available by calling PduR_FrTpStartOfReception(). Afterwards FrTp calls PduR_FrTpCopyRxData() in order to store the Rx-data.
3. After the successful provision of Rx-buffer-space the reception should begin, e.g. by the transmission of a flow control frame using FrIf_Transmit() in case of an 1:1 connection. You can verify the transmission of flow control frames by assuring that the function FrIf_Transmit() is called by the FrTp and the function FrTp_TxConfirmation() is called by the FrIf-component that is used. There is only one location in FrTp.c where FrIf_Transmit() is called.
4. Each call to FrIf_Transmit() of the FrTp should be acknowledged by a call to FrTp_TxConfirmation() done by the FrIf component.
5. In any case the callback <Owner>_FrTp_RxIndication() should be called after the successful reception of at least one Iso10681-frame. The notification-code given to <Owner>_FrTp_RxIndication() will provide additional information about success or failure of the reception.

Please refer to document [2] on how to interpret parameters and notification-codes of the functions mentioned.

4.5.4 Buffer Handling

During one call of the FrTp_MainFunction() the retrieval of Tx-data or Rx-buffer-space is done. For Rx-buffer-retrieval no limitations apply, because the ISO 10681-2 protocol allows the FrTp to communicate to the sender the exact buffer size available at the receiving application. However for Tx-buffer-retrieval the following restriction applies:

Caution
Rx-Buffer and Tx-data handling for functional communication

Please note, that the FrTp does not support multiple calls to PduR_FrTpCopyTxData / PduR_FrTpCopyRxData in case the application does not supply the Tx-data / Rx-buffer-space of **functional** connections **in one piece**. I.e. in case of a 246 byte unsegmented functional STF the application has to process the complete payload within exactly one call to PduR_FrTpCopyTxData / PduR_FrTpCopyRxData.

4.6 Troubleshooting

Symptom	Solution
An FrTp transmission / reception stops that comes from or goes to a Gateway-ECU before the data is transferred completely.	Check the exchanging of Rx- or Tx-buffer-on the Gateway-ECU. Often the PduR or the transport-layer on the other bus-system has configurative problems (e.g. timing) that lead to the stopping of the Rx or Tx processes on the particular other bus-system.
The FrTp on a gateway Ecu stops transmitting	Check the Std_ReturnType code of the callback PduR_FrTpTxConfirmation() or (better:) the FrTp-internal result of type FrTp_NotifResultType that is used in FrTp-function FrTpIso_TxInit().
The FrTp on a gateway Ecu stops receiving (i.e. another <Bus>Tp stops sending (e.g. CanTp)	Check the Std_ReturnType code of the callback PduR_FrTpRxIndication() or (better:) the FrTp-internal result of type FrTp_NotifResultType that is used in FrTp-function FrTpIso_RxInit().
FrTp_GlobCfg.h	This file is a generated header file declaring the global configuration struct of the FrTp.

4.7 Postbuild Variants

4.7.1 Post-build Selectable

The MICROSAR Identity Manager (refer to [2]) is an implementation of the AUTOSAR 4 post-build selectable concept. It allows the ECU manufacturer to include several FrTp configurations within one ECU. With post-build selectable and the Identity Manager the ECU variants are downloaded within the ECUs non-volatile memory (e.g. flash) at ECU build time. Post-build selectable does not allow modification of FrTp aspects after ECU build time. At the same time, this limitation allows some of the optimization strategies still to be effective - FrTp static code part will be optimized for the variant with maximum configuration size. The variant selection is performed at runtime by passing the corresponding configuration root during the module initialization (refer to chapter 5.2.1).

4.7.2 Post-build loadable

All FrTp configuration parameters, that are classified to be post-build selectable, also do support post-build loadable variant. The differences to the post-build-selectable case relisted upon their qualification:

- advantages:
- The module's configuration can be updated after the module's compile time without reprogramming the whole ECU software.
- disadvantages:
- Since all of the affected configuration-parameters may change after modules compile time, the optimization level of the source code is very low.

> Since no maximum configuration size can be pre-calculated, some scalable RAM blocks are referred not by a direct linker symbol, but through a pointer.

> Only one configuration variant is supported at a time (no variant selection at runtime possible). This disadvantage is avoided if the post -build loadable selectable variant is chosen instead (refer to chapter 4.7.3).

> Greater risks of passing an invalid pointer during module initialization time. For details about the post-build loadable feature, please refer to [2].

4.7.3 Post-build loadable selectable

This variant actually combines both post-build selectable and loadable variants, allowing a variant selection at runtime and at the same time post-build calibration of parameters. For details on the two mentioned variants, please refer correspondingly to chapters 4.7.1 and 4.7.2.

4.7.4 Post-build deleteable

This variant is actually a specific sub-variant of the post-build loadable variants. It allows deleting of containers that were created at link time, by guaranteeing at the same time the preservation of other post-build capable parameters' values. For details about this feature, refer to document [2].

4.8 Recommended compiler switches

Note
Here you can obtain supplemental information.

In case you want to use the configuration variant post build together with a Greenhills® compiler on platform NEC® V850™ and compatible then you should take attention on the following compiler switch combination:

In case you want to use `option -pack=1` to reduce code size you should always use `-misalign_pack` at the same time in order to prevent a strong increase of the size of the static code of the FrTp.

5 API Description

For an interfaces overview please see Figure 2-2.

5.1 Type Definitions

There are no FrTp-specific type-definitions that go beyond the types offered by the AUTOSAR header files Std_Types.h and ComStack_Types.h that would affect the user-application. Thus there are no further types described in this chapter.

5.2 Services provided by FrTp

5.2.1 FrTp_Init

Prototype	
`retCode` **`FrTp_Init`** `(FrTp_ConfigType *CfgPtr)`	
Parameter	
`CfgPtr`	Pointer to the configuration structure of the FrTp.
Return code	
`Void`	n/a
Functional Description	
This method initializes all administration-data of the FrTp and prepares the FrTp component for operation. It has to be called prior to any other function-call to a FrTp -API. Otherwise the respective other API-call will report an error to the DET.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM). This is because this function reuses internal functions of the FrTp that are also used during runtime and that require interrupt-locking. > In case of post build configuration this method checks the validity of the given configuration versus the version of the embedded code of the FrTp component. In case the given configuration is invalid or not suitable for the embedded code then the callout function EcuM_BswErrorHook() is called.	
Expected Caller Context	
> May be called from interrupt context..	

Table 5-1 FrTp_Init

5.2.2 FrTp_InitMemory

Prototype
`void` **`FrTp_InitMemory`** `( void )`

Parameter	
`void`	n/a
Return code	
`void`	n/a
Functional Description	
This method initializes all administration-data of the FrTp that has to be set to zero before FrTp_Init() is called. FrTp_InitMemory() has to be called in case the startup code does not set the RAM used by FrTp to zero.	
Particularities and Limitations	
> Service ID: n/a (not an AUTOSAR compliant method) > This function is synchronous. > This function is non-reentrant. > This function has to be called during the initialization of the FlexRay-Stack, i.e. during ECU-startup in case the startup code does not initialize the RAM.	
Expected Caller Context	
> May be called from interrupt context.	

Table 5-2 FrTp_InitMemory

5.2.3 FrTp_Shutdown

Prototype	
`void` **`FrTp_Shutdown`** `( void )`	
Parameter	
`void`	n/a
Return code	
`void`	n/a
Functional Description	
This method shuts down the FrTp and prevents the initiation of transmissions in future. After the call the FrTp is in a non-initialized state.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This function solely changes a FrTp -internal status-variable. It does not cancel or stop any ongoing transmissions.	

Expected Caller Context
> May be called from interrupt context.

Table 5-3 FrTp_Shutdown

5.2.4 FrTp_Transmit

Prototype	
`Std_ReturnType` **`FrTp_Transmit`** `( PduIdType FrTpTxSduId, const PduInfoType* FrTpTxSduInfoPtr )`	
Parameter	
`FrTpTxSduId`	This parameter identifies the FrTpTxSdu to be used for transmission.
`FrTpTxSduInfoPtr`	This is a pointer to a struct of type PduInfoType whereas the structs parameter PduLength states the length of the data to be transmitted.
Return code	
`Std_ReturnType`	E_OK: This value is returned in case the transmission request is going to be executed by the FrTp component. E_NOT_OK: This value is returned in case the transmission request is not going to be executed by the FrTp component. This can happen under these conditions: 1. The length-value given in the struct-parameter is invalid, i.e. either larger than 64 kBytes or, in case of a 1:n connection, it might be larger than the payload of the smallest FrIf-Pdu that is assigned to the affected connection. 2. A connection identifier is provided that is invalid. Then the value usually is larger or equal than the overall number of connections configured in the current FrTp instance. 3. In case 1. or 2. are not applicable then typically all channels of the FrTp are busy at the moment of the call.
Functional Description	
This method initiates a transmission in case the given parameters are valid (see section “Return code” above). In this case a free channel of the FrTp is reserved to conduct the transmission requested. Within the next call to the FrTp_MainFunction() the steps required for the transmission will be taken, i.e. a buffer with Tx-Data is written by the application via PduR_ FrTpCopyTxData() and after successful transmission of the provided data a PduR_FrTpTxConfirmation()-callback is called. These steps are conducted in the same way as described in document [2].	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > It is not required that the physical layer is in sync with the networks since this function does not check the status of the physical FlexRay bus. > The PduR / caller should be prepared to receive a call to the callback-function PduR_FrTpCopyTxData() afterwards, in order to provide the Tx-data to be transmitted. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM).
Expected Caller Context
> May be called from interrupt context.

Table 5-4 FrTp_Transmit

5.2.5 FrTp_CancelTransmit

Prototype	
`Std_ReturnType` **`FrTp_CancelTransmit`** `(PduIdType FrTpTxPduId )`	
Parameter	
`FrTpTxPduId`	This parameter identifies the connection a cancellation request shall be applied to.
Return code	
`Std_ReturnType`	E_OK: This value is returned in case the request for transmission-cancellation is going to be executed by the component. E_NOT_OK: This value is returned in case the request for transmission-cancellation is invalid. This can happen in case the affected connection is not active at all.
Functional Description	
This method requests the cancellation of an active transmission.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > It is not required that the physical layer is in sync with the networks since this function does not check the status of the physical FlexRay bus. > After successful cancellation of the transmission no additional calls to PduR_FrTp_TxConfirmation() are done by the FrTp in order to finish the transmission. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM). > This API is optional.
Expected Caller Context
> May be called from interrupt context.

Table 5-5 FrTp_CancelTransmit

5.2.6 FrTp_CancelReceive

Prototype	
`Std_ReturnType` **`FrTp_CancelReceive`** (PduIdType FrTpRxPduId)	
Parameter	
`FrTpRxPduId`	This parameter identifies the connection a cancellation request shall be applied to.
Return code	
`Std_ReturnType`	E_OK: This value is returned in case the request for reception-cancellation is going to be executed by the component. E_NOT_OK: This value is returned in case the request for reception-cancellation is invalid e.g. because the affected connection is not active at all.
Functional Description	
This method requests the cancellation of an active reception.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > It is not required that the physical layer is in sync with the networks since this function does not check the status of the physical FlexRay bus. > After successful cancellation of the reception no additional calls to PduR_FrTp_RxIndication() or <Owner>_FrTp_RxIndication() are done by the FrTp in order to finish the reception. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM). > This API is optional.
Expected Caller Context
> May be called from interrupt context.

Table 5-6 FrTp_CancelReceive

5.2.7 FrTp_ChangeParameter

Prototype	
`Std_ReturnType` **`FrTp_ChangeParameter`**`(PduIdType FrTpSduId, TPParameterType FrTpParameterType, uint16 FrTpParameterValue )`	
Parameter	
`FrTpSduId`	This parameter identifies the connection a request to change parameters shall be applied to
`FrTpParameterType`	This parameter contains the information about which parameter of the given connection shall be changed.
`FrTpParameterValue`	This parameter contains the new value to be set as new parameter-value.
Return code	
`retCode`	E_OK: Request was accepted E_NOT_OK: Request was not accepted
Functional Description	
This method changes the parameter of a FrTp connection identified by the type-parameter using the given value-parameter as value. In case the SCexp part of the given BC-parameter leads to a larger inter-pdu-gap than configured in the FrTpTimeoutCr-parameter of the affected connection the parameter will not be changed and E_NOT_OK will be returned.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > Other particularities, limitations, post-conditions, pre-conditions	

Expected Caller Context
> May be called from interrupt context.

Table 5-7 FrTp_ChangeParameter

5.2.8 FrTp_MainFunction

Prototype	
`void` **`FrTp_MainFunction`** `( void )`	
Parameter	
`void`	n/a
Return code	
`void`	n/a
Functional Description	
This method conducts these actions: - It supervises the timers of each TP-channel. - It evaluates the actions that have happened since the last call to FrTp_MainFunction(), i.e. callbacks to the methods FrTp_RxIndication(), FrTp_TxConfirmation(). - It conducts all calls to FrIf_Transmit() required by a FrTp communication.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > This function has to be called synchronous to the FlexRay bus-time. > FrTp_Init() might not be called in postbuild loadable / selectable use-cases. Therefore no DET errror FRTP_E_NOT_INIT is issued by the FrTp_MainFunction(). > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM). > This function has to be called at a different time than the execution-time of the FrIf-job-execution (Both Rx and Tx !!!). Both points in time are possible: Prior or after the point of time of the FrIf-job-execution.	
Expected Caller Context	
> May be called from interrupt context.	

Table 5-8 FrTp_MainFunction

Caution
Rules to call FrTp_MainFunction()

The FrTp_MainFunction() has to be called:

- once per FlexRay cycle and also
- synchronous to the FlexRay time.
- This function should be called at a different time than the execution-time of the FrIf-job-execution that operates the FrIf-Rx- and FrIf-Tx-Pdus, used by the FrTp. Otherwise this will result in poor performance of the FrTp.
- Refer to document [13] for more details about FrIf-JLE.

Example
Call schedule of FrTp_MainFunction()

In case of 5 ms FlexRay cycle length and all FrIf-Tx-Pdus assigned to the FrTp being in the dynamic segment then the FrTp_MainFunction() should be executed in the static segment.

The offset within the cycle should be:

- earlier than the Tx-Job of the FrIf responsible for FrTp FrIf-Tx-Pdus
- later than the Rx-Job of the FrIf responsible for FrTp FrIf-Rx-Pdus

Other offsets are possible but will result in poor performance of the FrTp.

5.2.9 FrTp_GetVersionInfo

Prototype	
`void **Frtp_GetVersionInfo** (Std_VersionInfoType *pVersionInfo)`	
Parameter	
`pVersionInfo`	The version-info-struct the parameter “pVersionInfo” points to is filled with the version-information by this method.
Return code	
`retCode`	description
Functional Description	
FrTp_GetVersionInfo() returns version information, vendor ID and AUTOSAR module ID of the component. The versions are BCD-coded.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > It is not required that the physical layer is in sync with the networks since this function does not check the status of the physical FlexRay bus.
Expected Caller Context
> May be called from interrupt context.

Table 5-9 FrTp_GetVersionInfo

5.3 Services used by FrTp

5.3.1 Services with AUTOSAR compliant prototypes

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

Component	API
Det	Det_ReportError
PduR	PduR_FrTpStartOfReception
PduR	PduR_FrTpCopyRxData
PduR	PduR_FrTpCopyTxData
PduR	PduR_FrTpRxIndication
PduR	PduR_FrTpTxConfirmation
SchM	SCHM_ENTER_EXCLUSIVE
SchM	SCHM_EXIT_EXCLUSIVE

Table 5-10 Services used by the FrTp

5.3.2 Services that are MICROSAR extensions

n case of incompatibilities of configuration and embedded code then the function EcuM_BswErrorHook()is called in the context of FrTp_Init():

Component	API
EcuM	EcuM_BswErrorHook

5.4 Callback Functions

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

5.4.1 FrTp_TxConfirmation

Prototype	
`void` **`FrTp_TxConfirmation`** `(PduIdType FrIfTxPduId )`	
Parameter	
`FrIfTxPduId`	This parameter identifies the FrIf-Tx-Pdu assigned to the FrTp that has been transmitted successfully on the FlexRay-bus by the FrIf.
Return code	
`void`	n/a
Functional Description	
This method has to be called after the transmission of all FrIf-Tx-Pdus assigned to the FrTp.	
Particularities and Limitations	
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > After the call to this method the FrTp regards all other Tx-Pdus of the affected channel as transmitted. Of course only these Tx-Pdus are affected that have been requested by the FrTp to be transmitted by the FrIf in the current FlexRay-cycle or this. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM).	
Expected Caller Context	
> May be called from interrupt context.	

Table 5-11 FrTp_TxConfirmation

5.4.2 FrTp_RxIndication

Prototype	
`void` **`FrTp_RxIndication`** (PduIdType FrRxPduId, const PduInfoType* FrIf_PduInfoPtr)	
Parameter	
`FrRxPduId`	This parameter indicates to the FrTp which Rx-Pdu has been received successfully by the underlying FlexRay-Interface component.
`FrIf_PduInfoPtr`	This parameter points to a C-struct of type PduInfoType. The C-struct contains a pointer to the Rx-Pdu received and a value of type PduLengthType that indicates the length of the Pdus payload.
Return code	
`void`	n/a
Functional Description	
This method is a callback that is called by the FrIf in case an Rx-Pdu assigned to the FrTp has been received successfully by the FrIf. The method copies the data pointed to by the member DataPtr of the PduInfoStruct to a FrTp-internal buffer or directly to the Rx-buffer of the application. For this the member PduLength of the PduInfoStruct is used. This is done depending on the frame-type received.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM).
Expected Caller Context
> May be called from interrupt context.

Table 5-12 FrTp_RxIndication

5.4.3 FrTp_TriggerTransmit

Prototype	
`Std_ReturnType` **`FrTp_TriggerTransmit`** `(PduIdType FrIfTxPduId, PduInfoType* FrTpTxPduInfoPtr )`	
Parameter	
`FrIfTxPduId`	This parameter contains the Id of the Tx-Pdu (FrIf-L-PduId of the FrIf) whose transmission was requested and whose payload-data shall be provided by the FrTp to the FrIf.
`FrTpTxPduInfoPtr`	Pointer to PduInfoType-structure containing a pointer to memory of the FrIf (or of the Fr-driver) the FrTp shall copy the payload and containing the information about the length of the FrTp-frame.
Return code	
`Std_ReturnType`	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted, e. g. parameter check has failed or cancellation is requested.
Functional Description	
This method is called by the FrIf-Job-Execution in order to retrieve the payload-data of a FrIf-Tx-Pdu assigned to and previously transmitted by the FrTp via FrIf_Transmit(). This function copies the complete FlexRay TP-frame to a memory-area pointed to by the FrIf.	

Particularities and Limitations
> Service ID: see table 'Service IDs' > This function is synchronous. > This function is non-reentrant. > FrTp_Init() has to be called prior to any calls to this method, i.e. the FrTp has to be initialized. Otherwise an error will be indicated to the DET. > The transmission of the Tx-Pdu indicated by parameter FrIfTxPduId must have been requested by the FrTp in a preceding call to the FrIf-method FrIf_Transmit() whereas the latter call has to return E_OK in this case. > FrTp_TxConfirmation() has to be called by the FrIf in order to finalize the transmission of the affected FrIf-Tx-Pdu assigned to the FrTp. > During its runtime this method temporarily locks interrupts using the API-calls of the AUTOSAR BSW Scheduler (SchM). > This method is optional. It can be disabled by GUI switch ‘Disable Decoupled Transmit’ in case you plan to solely use FrIf-Pdus with transmission type ‘immediate’ in future.
Expected Caller Context
> May be called from interrupt context.

Table 5-13 FrTp_TriggerTransmit

6 Glossary and Abbreviations

6.1 Glossary

Term	Description
EAD	Embedded Architecture Designer; generation tool for MICROSAR components
GENy	Generation tool for CANbedded and MICROSAR components

Table 6-1 Glossary

6.2 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

Table 6-2 Abbreviations

7 Contact

Visit our website for more information on

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

www.vector.com