Document Title	Specification of FlexRay ISO Transport Layer
Document Category	Software specification (SWS)
Document UID	589
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Deleted GetVersionInfo API • Reclassifcation of Ext. Prod. Errors
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• Migration from word to Latex • Figures moved out of specification items • Adapted lower layer communication to LSduR
2023-11-23	R23-11	AUTOSAR Release Management	• Added Extended Production Errors to indicate timeouts and errors
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• Removed [SWS_FrTp_01132], [SWS_FrTp_01111] • [SWS_FrTp_01106] moved to chapter 7.4 • Structure of Chapter 7.7 changed
2019-11-28	R19-11	AUTOSAR Release Management	• Header file name changes in Chapter 8 • Changed Document Status from Final to published



△

2018-10-31	4.4.0	AUTOSAR Release Management	• Header file cleanup • Resolved inconsistent behavior of BSW modules in un-initialized state
2017-12-08	4.3.1	AUTOSAR Release Management	• Removed HIS from acronym table in section 2 and reference to HIS MISRA subset from Section 3.2 • For Rollout of Runtime errors: – DET errors FRTP_E_SEG_ERROR and FRTP_E_NO_CHANNEL are moved to new section called 'Runtime Errors' ([SWS_FrTp_01208]) – Updated requirements [SWS_FrTP_01187], [SWS_FrTP_01068],[SWS_FrTP_01185],[SWS_FrTP_01186]
2016-11-30	4.3.0	AUTOSAR Release Management	• Removed configuration parameters FrTpMaxBufferSize, FrTpMaxAs, FrTpMaxAr, FrTpMaxFrIf, FrTpTimeFrIf, FrTpTimeoutBr, FrTpTimeoutCs • Addressing in Upper Layers using MetaData • Introduced reliable TxConfirmation • Editorial changes
2015-07-31	4.2.2	AUTOSAR Release Management	• Updated the SWS requirements for DET renaming • Updated the SWS requirements [SWS_FrTp_01047] and added a note for the Tx Pdu processing
2014-10-31	4.2.1	AUTOSAR Release Management	• Added FRTP_TIME_CS in table 2, FRTP_TIMEOUT_BR and FRTP_TIMEOUT_CS in table 3 • Updated the "Use cases for NULL_PTR in CopyRxData and CopyTxData should be allowed" • Updated [SWS_FrTp_01132], [SWS_FrTp_01140], [SWS_FrTp_01146], [SWS_FrTp_01148], ▽

▽

△

			△ [SWS_FrTp_01150] for FRTP_E_PARAM_POINTER • Added FRTP_E_INIT_FAILED in thhe [SWS_FrTp_01132] (table)
2014-03-31	4.1.3	AUTOSAR Release Management	• Modified [ECUC_FrTp_00024], [SWS_FrTp_00150], [SWS_FrTp_00152], [SWS_FrTp_00153], [SWS_FrTp_01092], [SWS_FrTp_01141], [SWS_FrTp_01147], [SWS_FrTp_01148], [SWS_FrTp_01149] • Added description in the section 7.5.4 Buffer Handling • Modified chapter 8.6.2.1 name to Development Error Tracer • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Removed requirement [SWS_FrTp_01166] • Removed chapter 8.2.1, 8.2.1.1 • Removed chapter 7.5.4.2 • Modified [SWS_FrTp_01149] • Added new requirement describing the layout of BC parameter • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Corrected Retry Handling Mechanism • Clarified usage of BUFREQ_E_BUSY • Removed references to ChangeParameterConfirmation • Removed private values in NotifyResultType • Changes to support Harmonization of ECU Parameters concept • Updated scope value of configuration parameters

▽

△

2011-12-22	4.0.3	AUTOSAR Administration	• Renaming (ISO) and new UID (029 -> 589) • API Names modified
2010-09-30	3.1.5	AUTOSAR Administration	• Time_CS removed from table 2 • Add FrTp051 and Figure 24, Table 4 and Table 5 modified, renamed FrTpMaxBufReq to FrTpMaxFcWait, COUNTER_RX_BUFREQ and COUNTER_TX_BUFREQ removed • Transport Protocol supports data transfers up to 2^{16-1} Bytes payload • Remove Chapter 7.5.4.3 with FrTp1086 and FrTp1087, remove COUNTER_BS, COUNTER_CR, COUNTER_TX_RN
2010-02-02	3.1.4	AUTOSAR Administration	• FrTp according to [1] • New PduR API • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Administration	• Tables generated from UML-Models, UML-Diagrams linked to UML-Model, general improvements of requirements in preparation of CT-development. No changes in the technical contents of the specification
2007-12-21	3.0.1	AUTOSAR Administration	• Clarified the role and purpose of the functions PduR_FrTpChangeParameterConfirmation() and PduR_FrTpChannelTransmitConfirmation() with respect to the PDU Router • Document meta information extended • Small layout adaptations made
2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

This work (specification and/or software implementation) and the material contained in it, as released by AUTOSAR, is for the purpose of information only. AUTOSAR and the companies that have contributed to it shall not be liable for any use of the work.

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

This work may be utilized or reproduced without any modification, in any form or by any means, for informational purposes only. For any other purpose, no part of the work may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 10
2 Acronyms and Abbreviations 13
3 Related documentation 15
3.1 Input documents & related standards and norms . . . 15
3.2 Related specification . . . 15
4 Constraints and assumptions 16
4.1 Limitations . . . 16
4.2 Applicability to car domains . . . 16
4.3 Restriction to ISO 10681-2 . . . 16
5 Dependencies to other modules 17
5.1 FlexRay Transport Layer interactions . . . 17
5.2 PDU Router . . . 17
5.3 L-SDU Router . . . 18
5.4 ECU State Manager . . . 19
5.5 Default Error Tracing . . . 19
5.6 File structure . . . 19
5.6.1 Code file structure . . . 19
5.6.2 Header file structure . . . 19
5.6.3 Design rules . . . 19
6 Requirements Tracing 21
7 Functional specification 22
7.1 FrTp usage scenarios . . . 22
7.2 FrTp behaviour according to ISO10681-2 . . . 23
7.2.1 Protocol Data Unit (PDU) . . . 23
7.2.2 Frame Sequence charts . . . 23
7.2.2.1 Unsegmented unacknowledged data transfer with known message length . . . 23
7.2.2.2 Unsegmented acknowledged data transfer with known message length . . . 24
7.2.2.3 Segmented unacknowledged data transfer with known message length . . . 24
7.2.2.4 Segmented acknowledged data transfer with known message length . . . 25
7.2.2.5 Segmented unacknowledged data transfer with unknown message length . . . 26
7.2.2.6 Segmented acknowledged data transfer with unknown message length . . . 26
7.2.3 Limitation to ISO10681-2 . . . 27
7.3 Internal Module behaviour specification . . . 27

7.3.1 Overview . . . 27
7.3.2 Configuration data . . . 29
7.3.2.1 FrTpConnection . . . 29
7.3.2.2 FrTpTxPduPool . . . 30
7.3.2.3 FrTpConnectionControl . . . 30
7.3.3 Runtime data . . . 31
7.3.3.1 FrTpRuntime . . . 31
7.3.3.2 FrTpChannel . . . 32
7.3.3.3 FrTpConnectionControlRuntime . . . 35
7.4 Initialization and shutdown . . . 35
7.5 Data Transfer Processing . . . 36
7.5.1 Flags . . . 37
7.5.1.1 TX_SDU_AVAILABLE . . . 37
7.5.1.2 TX_SDU_UNKNOWN_MSG_LENGTH . . . 37
7.5.1.3 RX_PDU_AVAILABLE . . . 38
7.5.1.4 RX_ERROR . . . 38
7.5.2 Transmit Data . . . 38
7.5.2.1 Transmit Data via 'Immediate Buffer Access' Mode . . . 39
7.5.2.2 Transmit Data via 'Decoupled Buffer Access' Mode . . . 45
7.5.2.3 Data Transfer with unknown message length . . . 47
7.5.3 Receive Data . . . 48
7.5.3.1 Receive with unknown message length . . . 53
7.5.4 Buffer Handling . . . 53
7.5.4.1 Buffer Access Mode . . . 54
7.5.5 Dynamic Bandwidth Assignment . . . 54
7.5.6 Transmit Cancellation . . . 58
7.5.6.1 Transmit Cancellation for unsegmented data transfer . . . 58
7.5.6.2 Transmit Cancellation for segmented data transfer . . . 59
7.5.7 Change FrTp Parameter . . . 60
7.5.8 Timing parameter and timeout behaviour . . . 61
7.6 Counters . . . 65
7.7 Error Classification . . . 67
7.7.1 Development Errors . . . 67
7.7.2 Runtime Errors . . . 68
7.7.3 Production Errors . . . 68
7.8 Security Events . . . 68
8 API specification 69
8.1 Imported types . . . 69
8.2 Type definitions . . . 69
8.2.1 FrTp_ConfigType . . . 69
8.3 Function definitions . . . 70
8.3.1 Initialization and Shutdown . . . 70
8.3.1.1 FrTp_Init . . . 70

8.3.1.2 FrTp_Shutdown . . . 71
8.3.2 Normal Operation . . . 71
8.3.2.1 FrTp_Transmit . . . 71
8.3.2.2 FrTp_TriggerTransmit . . . 72
8.3.2.3 FrTp_CancelTransmit . . . 73
8.3.2.4 FrTp_TxConfirmation . . . 73
8.3.2.5 FrTp_RxIndication . . . 74
8.3.2.6 FrTp_CancelReceive . . . 75
8.3.2.7 FrTp_ChangeParameter . . . 76
8.4 Scheduled functions . . . 77
8.4.1 FrTp_MainFunction . . . 77
8.5 Expected interfaces . . . 78
8.5.1 Mandatory interfaces . . . 78
8.5.2 Optional interfaces . . . 78
8.5.3 Configurable interfaces . . . 79
8.6 Service Interfaces . . . 79
9 Sequence diagrams 80
9.1 Sending of N-Pdus . . . 80
9.2 Receiving of N-Pdus . . . 82
10 Configuration specification 84
10.1 How to read this chapter . . . 84
10.2 Containers and configuration parameters . . . 84
10.2.1 FrTp . . . 84
10.2.2 FrTpGeneral . . . 85
10.2.3 FrTpMultipleConfig . . . 89
10.2.4 FrTpConnection . . . 90
10.2.5 FrTpTxSdu . . . 94
10.2.6 FrTpRxSdu . . . 95
10.2.7 FrTpConnectionControl . . . 97
10.2.8 FrTpTxPduPool . . . 101
10.2.9 FrTpRxPduPool . . . 102
10.2.10 FrTpTxPdu . . . 103
10.2.11 FrTpRxPdu . . . 104
10.2.12 FrTpDemEventParameterRefs . . . 105
10.3 Published Information . . . 106
10.4 Configuration dependencies and recommendation . . . 106
10.4.1 Retry behaviour . . . 106
10.4.2 TP-Acknowledgement and Retry . . . 107
10.4.3 Timing and Timeout Parameters . . . 107
10.4.4 Bandwidth Control Configuration . . . 108
10.4.4.1 BandwidthControl by FlowControl Parameter in combination with Hardware FIFO buffer mechanisms . . . 108

10.4.4.2 BandwidthControl by FlowControl Parameter 109
10.4.4.3 BandwidthControl via different PDU Pools 110
10.4.5 Configuration Requirements on the FlexRay Interface 111
A Change history of AUTOSAR traceable items 112
A.1 Traceable item history of this document according to AUTOSAR Release R26-11 112
A.1.1 Added Specification Items in R26-11 112
A.1.2 Changed Specification Items in R26-11 112
A.1.3 Deleted Specification Items in R26-11 112
A.2 Traceable item history of this document according to AUTOSAR Release R25-11 114
A.2.1 Added Specification Items in R25-11 114
A.2.2 Changed Specification Items in R25-11 114
A.2.3 Deleted Specification Items in R25-11 114
B Not applicable requirements 115

1 Introduction and functional overview

This specification describes the functionality, API, and the configuration of the AUTOSAR basic software module FlexRay Transport Protocol (FrTp).

According to the AUTOSAR layered software architecture [2] (see Figure 1.1), the FlexRay Transport Protocol (FrTp) is placed between the PDU Router module and the L-SDU Router module. The main purpose of the FlexRay Transport Protocol is segmentation and reassembly of messages (I-PDUs) that do not fit in one of the assigned FlexRay L-PDUs.

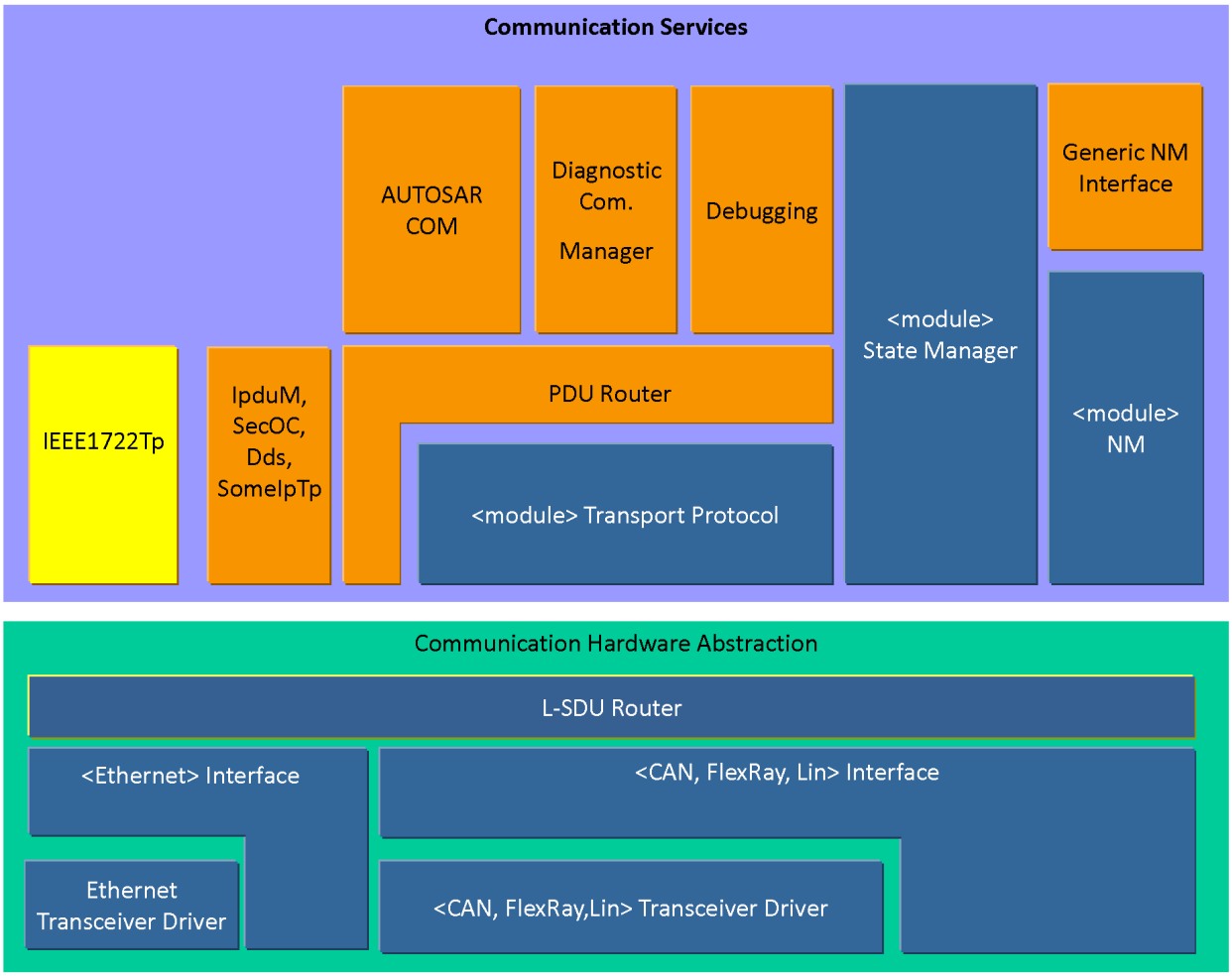


Figure 1.1: AUTOSAR FlexRay Layered Architecture

Figure 1.2 depicts the different PDU names in AUTOSAR nomenclature and the signal dependencies between the different AUTOSAR modules.

The PDU Router deploys upper Layers (e.g. COM, DCM etc) I-PDUs onto different communication protocols. The routing through a network system type (e.g. FlexRay, CAN, LIN etc.) depends on the I-PDU identifier. The PDU Router also determines (by

configuration) if a transport protocol shall be used or not. Finally, this module carries out gateway functionality, when there is no rate conversion.

Through the L-SDU Router [3] (LSduR) the FlexRay Interface [4] (FrIf) provides standardized mechanisms to access a FlexRay bus channel via a FlexRay Communication Controller regardless of its location (µC internal/external). Depending on the PDU ID, the L-SDU Router has to forward an N-PDU to FrTp or an I-PDU to PduR. The FrTp handles only transport protocol N-PDUs (i.e. SF, CF, LF and FC PDUs).

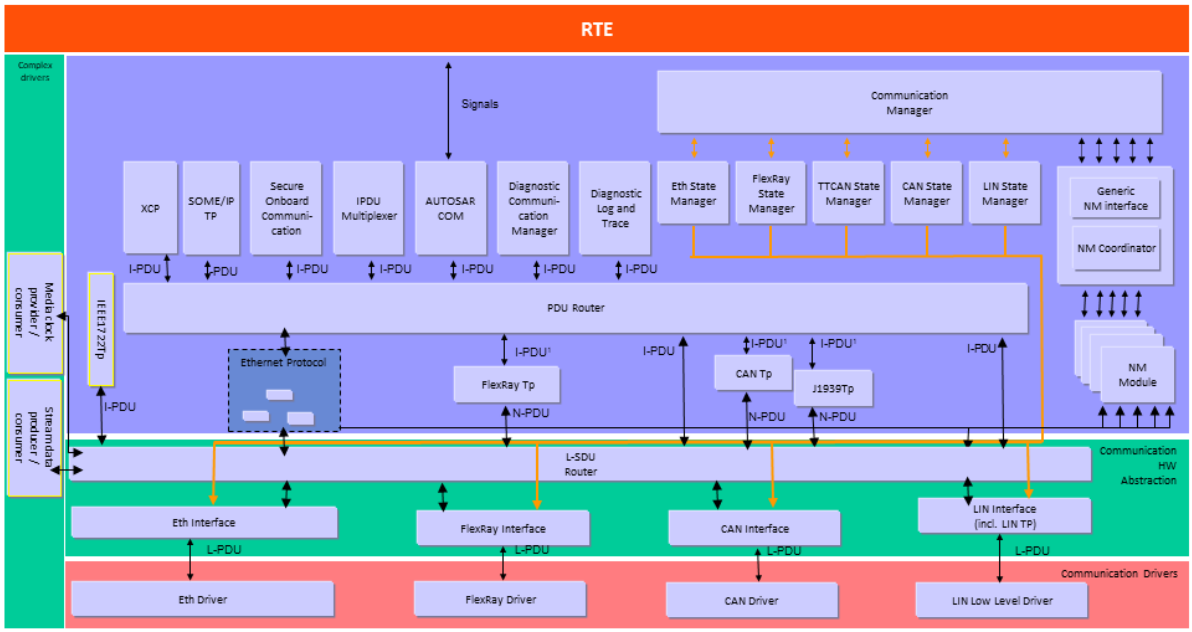


Figure 1.2: AUTOSAR Signal Nomenclature

The main purpose of FlexRay Transport Protocol is to perform a transfer of a message (I-PDU) that e.g. might or might not fit in a single FlexRay L-PDU. I-PDUs that do not fit into a single FlexRay L-PDU are segmented into multiple parts, where each can be transmitted in a FlexRay L-PDU. According to AUTOSAR basic software architecture, the FlexRay Transport Protocol provides methods for

- Segmentation of data in transmit direction
- Reassembling of data in receive direction
- Control of data flow
- Error detection in segmentation sessions
- Transmit cancellation

It is an AUTOSAR decision to base basic software module specifications on existing standards, thus this AUTOSAR FlexRay Transport Layer specification is based on the international standard [1]. This standard provides

- Transmission of a message with known message length
- Transmission of a message with unknown but finite message length

- Acknowledgement of transmission with retry mechanism

The FlexRay Transport Protocol supports 1:1 and 1:n connections.

The FlexRay Transport Protocol supports data transfers of up to 2^{16}-1 Bytes payload.

FlexRay Transport Protocol is mainly used for vehicle diagnostic systems. Nevertheless, it was developed to deal with requirements from other FlexRay based systems requiring a Transport Protocol Layer protocol (e.g. Diagnostic communication, Inter-ECU communication, XCP communication etc.).

2 Acronyms and Abbreviations

The prefix notation used in this document, is as follows:

Prefix:	Description:
I-	Relative to upper AUTOSAR Layer (e.g. COM, DCM etc.)
L-	Relative to the FlexRay Interface module.
N-	Relative to the FlexRay Transport Protocol Layer.

Table 2.1: Prefix notation used in this Document

The glossary below includes acronyms and abbreviations relevant to the FlexRay Transport Layer module that are not included in the [5, AUTOSAR glossary].

Abbreviation / Acronym:	Description:
Fr L-SDU	This is the SDU of the FlexRay Interface module. It is similar to Fr N-PDU but from the FlexRay Interface module point of view.
Fr L-SduId	This is the unique identifier of a Fr-L-SDU within the FlexRay Interface. It is used for referencing L-SDU's routing properties.
Fr N-PDU	This is the PDU of the FlexRay Transport Layer. It contains address information, protocol control information and data (the whole Fr N-SDU or a part of it).
Fr N-SDU	This is the SDU of the FlexRay Transport Layer. In the AUTOSAR architecture, it is a set of data coming from the PDU Router.
Fr N-SduId	Unique N-SDU identifier within the FlexRay Transport Layer. It is used to reference N-SDU's routing properties.
I-PDU	This is the PDU of the upper AUTOSAR Layers modules (e.g. COM, DCM etc.). If data transfer via FlexRay Transport Protocol is configured, I-PDU is similar to an FrTp N-SDU.
PDU	In layered systems, it refers to a data unit that is specified in the protocol of a given layer. This contains user data of that layer (SDU) plus possible protocol control information. Furthermore, the PDU of layer X is the SDU of its lower layer X-1 (i.e. (X)-PDU = (X-1)-SDU).
`PduInfoType`	This type refers to a structure used to store basic information to process the transmission\reception of a PDU (or a SDU), namely a pointer to its payload in RAM and the corresponding length (in bytes).
Channel	A channel is a resource of the FrTp module to handle communication links to other communication nodes from FrTp's point of view (transferring Fr N-PDUs).
Connection	A connection specifies a communication link between different communication nodes from FrTp's point of view. A connection defines the sender / receiver relation of communication nodes
PCI	Protocol Control Information
e.g.	lat. 'exempli gratia' - engl. for example
i.e.	lat. 'id est' - engl. that is
CanTp	CAN Transport Protocol
LinTp	LIN Transport Protocol
CF	Consecutive Frame Fr N-PDU
COM	AUTOSAR Communications module
DCM	Diagnostic Communication Manager module
DET	Default Error Tracer

▽

△

FC	Flow Control Fr N-PDU
Fr	FlexRay Driver module
FrIf	FlexRay Interface module
LSduR	L-SDU Router
FrTp	FlexRay Transport Protocol Layer
PduR	PDU Router
SF	Start Frame Fr N-PDU
LF	Last Frame Fr N-PDU
TP	Transport Protocol Layer
SDU	In layered systems, this refers to a set of data that is sent by a user of the services of a given layer, and is transmitted to a peer service user, whilst remaining semantically unchanged.
AUTOSAR	Automotive Open System Architecture
SWS	Software Specification
MISRA	Motor Industry Software Reliability Association
ISO	International Standard Organisation
ID	Identifier
OS	Operating System
MCAL	Microcontroller Abstraction Layer
CPU	Central Processing Unit
ROM	Read Only Memory
RAM	Random Access Memory
STF	Start Frame (please refer to [1])
AF	Acknowledge Frame (please refer to [1])
SN	Sequence Number (please refer to [1])
FrTp_As	Timer Parameter for a sender. Time for transmission of the FlexRay frame (any N_PDU) on the sender side. (please refer to [1])
FrTp_Ar	Timer Parameter for a receiver. Time for transmission of the FlexRay frame (any N_PDU) on the receiver side (please refer to [1])
FrTp_BS	Timer Parameter for a sender. Time until reception of the next FlowControl N_PDU. (please refer to [1])
FrTp_Br	Timer Parameter for a receiver. Time until transmission of the next FlowControl N_PDU. (please refer to [1])
FrTp_Cs	Timer Parameter for a sender. Time until transmission of the next ConsecutiveFrame N_PDU/LastFrame N_PDU. (please refer to [1])
FrTp_Cr	Timer Parameter for a receiver. Time until reception of the next ConsecutiveFrame N_PDU (please refer to [1])

Table 2.2: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] ISO 10681-2:2010 – Communication on FlexRay – Part 2: Communication layer services
https://www.iso.org

[2] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture

[3] Specification of Linklayer Sdu Routing Module
AUTOSAR_CP_SWS_LSduRouter

[4] Specification of FlexRay Interface
AUTOSAR_CP_SWS_FlexRayInterface

[5] Glossary
AUTOSAR_FO_TR_Glossary

[6] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[7] Specification of PDU Router
AUTOSAR_CP_SWS_PDURouter

[8] Requirements on FlexRay
AUTOSAR_CP_RS_FlexRay

[9] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [6, SWS BSW General], which is also valid for Flex Ray ISO Transport Layer.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Flex Ray ISO Transport Layer.

4 Constraints and assumptions

4.1 Limitations

The AUTOSAR architecture defines communication system specific transport protocol layers (FrTp, CanTp, LinTp etc.). Thus, the FlexRay Transport Protocol layer (FrTp) only covers FlexRay transport protocol specifics.

The FlexRay Transport Protocol has an interface to a single underlying Linklayer SDU Router and a single upper PDU Router.

All reported DTCs are set as tested (notTestedThisCycle = 0) with each start-up. This would result in a toggling DTC for permanent issues, increasing the Occurrence - and/or OperationCycle Counters (if configured) with each power cycle.

4.2 Applicability to car domains

The FlexRay module can always be used for applications if the FlexRay protocol was used.

4.3 Restriction to ISO 10681-2

The AUTOSAR FrTp module does not support [1] functionalities as listed below:

Functionality:	Description:
Status monitoring	[1] provides the functionality to monitor the status of an active data transfer. Thus, the ISO 10681-2 API provides the service primitives C_GetStatus.request and C_GetStatus.confirm.
Read Communication Parameter values	[1] provides the functionality to read out current communication parameter values. Thus, the ISO 10681-2 API provides the service primitives C_GetParameters.request and C_GetParameters.confirm.

Table 4.1: Limitation of AUTOSAR FrTp vs. ISO 10681-2

5 Dependencies to other modules

This section sets out relations between the FlexRay Transport Protocol (FrTp) and other AUTOSAR basic software modules. It contains brief descriptions of the services, which are required by the FrTp from other modules or which are required by other modules from the FrTp.

5.1 FlexRay Transport Layer interactions

The FrTp's upper interface offers the PduR module global access, to transmit and receive data (Fr N-SDU). FlexRay N-SDU identifiers (Fr N-SDU-ID) achieve this access. FlexRay N- SDU-ID refers to a constant data structure that consists of attributes describing FlexRay N-SDU. The figure below shows the interactions between FrTp, PduR and L-SDU Router modules.

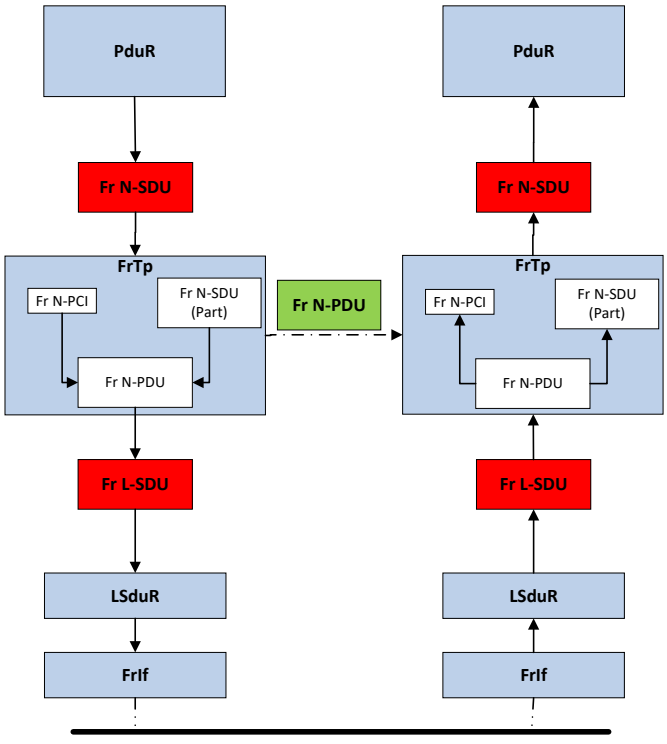


Figure 5.1: FrTp interactions

5.2 PDU Router

The FrTp module requests different services primitives provided by the PDU Router module. The requested services primitives of the PDU-Router module are listed below. For further details please refer to chapter 8.5 and specification [7]:

PduR_FrTpStartOfReception By this API service primitive, the FrTp indicates the start of reception of a FrTp-I-PDU.

PduR_FrTpCopyRxData By this API service primitive, the FrTp initiates the copy process of the received FrTp N-PDU payload data to a provided <Upper Layer> Rx buffer

PduR_FrTpRxIndication By this API service primitive, the FrTp indicates the completed (un)successful reception of an FrTp-I-PDU.

PduR_FrTpCopyTxData By this API service primitive, the FrTp initiates the copy process of the FrTp N-PDU payload data from the provided <Upper Layer> Tx buffer

PduR_FrTpTxConfirmation By this API service primitive, the FrTp module confirms the (un)successful sending of the complete Fr N-SDU to the corresponding sender module (e.g. COM, DCM etc).

The following services primitives of the FrTp module are called by the PDU-Router:

FrTp_Transmit By this API service primitive, a upper layer initiates a I-PDU data transfer via PDU-Router

FrTp_CancelTransmit By this API service primitive, sending of an Fr N-SDU is cancelled on sender site.

FrTp_ChangeParameter By this API service primitive, some communication parameters of the FrTp module could be changed.

FrTp_CancelReceive By this API service primitive, an ongoing reception could be canceled.

5.3 L-SDU Router

The following services primitives of the L-SDU Router (LSduR) module are called by the FrTp:

LSduR_FrTpTransmit By this API service primitive, the transfer of an Fr N-PDU is initiated.

The following services primitives of the FrTp module are called by the L-SDU Router module:

FrTp_RxIndication By this API service primitive, the L-SDU Router module indicates the reception of an FrTp frame (Fr N-PDU, not to be confused with a FlexRay frame) to the FrTp. The FrTp then processes this frame.

FrTp_TxConfirmation By this API service primitive, the L-SDU Router module confirms the sending of the frame containing the Fr N-PDU with result (`E_OK` if the transmission was successful, `E_NOT_OK` if the transmission failed) over the FlexRay network.

FrTp_TriggerTransmit By this API service primitive, the L-SDU Router get access to the Fr N-PDU data[1].

5.4 ECU State Manager

The following services primitives of the FrTp module are called by the ECU State Manager module ([SWS_EcuM_02859]):

FrTp_Init By this API service primitive, the FrTp module is initialized.

FrTp_Shutdown By this API service primitive, all active communication links are closed, resources are freed and the module is stopped.

5.5 Default Error Tracing

The following services primitives of the Default Error Tracing module are called by the FrTp module:

- `Det_ReportError`: By this API service primitive, the FrTp module reports development errors.

5.6 File structure

5.6.1 Code file structure

For details refer to [6] Chapter 5.1.6 *"Code file structure"*.

5.6.2 Header file structure

[SWS_FrTp_01157] Header shall contain all exported data ⌈FrTp.h shall contain all data exported from the FrTp - API declarations, callbacks, extern types and global data.⌋

5.6.3 Design rules

[SWS_FrTp_00209] Compiler and Platform independance ⌈The source code of the FrTp module shall be neither compiler (tool) nor platform (processor) dependent[2].⌋

[1] Depending on the configured buffer access mode.
[2] No compiler specifc keywords shall be used

[SWS_FrTp_01129] Support of configuration updates

Upstream requirements: SRS_Fr_05123

⌈The FlexRay Transport Layer module architecture shall support configuration modification by a dedicated update process (e.g. flash reprogramming).⌋

6 Requirements Tracing

The following tables reference the requirements specified in [6] and [8] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_FrTp_00147]
[SRS_BSW_00458]	Classification of production errors	[SWS_FrTp_00367]
[SRS_Fr_05073]	The FlexRay Transport Layer shall be configured to be compliant with the ISO 10681-2 specification	[SWS_FrTp_01005] [SWS_FrTp_01006]
[SRS_Fr_05077]	Each N-SDU shall have a unique identifier	[SWS_FrTp_01018]
[SRS_Fr_05088]	FlexRay Transport Layer’s variables shall be initialized	[SWS_FrTp_01034] [SWS_FrTp_01035]
[SRS_Fr_05093]	A cancellation service of transmission shal be provided at any time	[SWS_FrTp_01005] [SWS_FrTp_01006]
[SRS_Fr_05095]	The FlexRay Transport Layer shall support the dynamic bandwidth control mechanism	[SWS_FrTp_01005] [SWS_FrTp_01006]
[SRS_Fr_05123]	The Configuration shall be modifiable by a Flashing Process	[SWS_FrTp_01129]

Table 6.1: Requirements Tracing

7 Functional specification

This section provides a description of the FlexRay Transport Protocol Layer functionality. It explains the services provided to the upper and lower layers and the internal behaviour of the FrTp Layer module.

The main purpose of the FlexRay Transport Protocol (FrTp) Layer is transferring messages (I-PDUs) that may or may not fit in a single FlexRay frame (L-PDU). The FrTp Layer module provides services for segmentation and reassembly of upper-layer messages (Fr N-SDUs). Hence FrTp module offers services for segmentation, transmission with flow control, and reassembly of messages.

While reading this document, it is necessary to bear in mind, that the Transport Protocol functionality (e.g. frame assembly, frame handling, error handling etc.) is according to [1, ISO 10681-2].

[SWS_FrTp_01005] ISO-10681-2 protocoll shall be followed

Upstream requirements: SRS_Fr_05073, SRS_Fr_05093, SRS_Fr_05095

⌈If no explicit recommendation or requirement is defined, the FrTp module shall follow the specification [1, ISO 10681-2].⌋

Transport protocol facilities will be used to transport AUTOSAR I-PDUs from different source modules (e.g. COM, DCM etc.). Therefore, the FrTp module is able to deal with multiple connections simultaneously (i.e. multiple segmentation sessions in parallel).

The maximum number of simultaneous active connections is statically configured. This configuration has an important impact on complexity and resource consumption (CPU, ROM and RAM) of the code generated, because resources (e.g. Rx and Tx state machines, variables used to work on N-PCI data and so on) have to be reserved for each simultaneous access.

7.1 FrTp usage scenarios

As depict in Figure 7.1, the FrTp module is usable within single Electronic Control Units (ECUs) as well as in Gateways. In both cases FrTp modules are responsible to handle communication via FlexRay but for gateway purpose some additional requirements have to be taken into account.

Note: Each time a special usage scenario has to be taken into account, a foot node is given within the document.

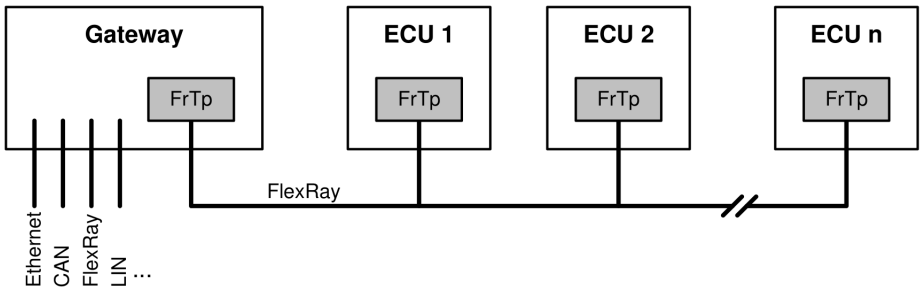


Figure 7.1: FrTp usage scenarios

7.2 FrTp behaviour according to ISO10681-2

This chapter gives a small overview about the Transport Protocol behaviour and data transmission szenarios according to [1, ISO 10681-2]. This chapter is only for a better understanding of the software solution to fullfill [1] and specifies no additional requirement.

7.2.1 Protocol Data Unit (PDU)

[SWS_FrTp_01006] Different N-PDU format support

Upstream requirements: SRS_Fr_05073, SRS_Fr_05093, SRS_Fr_05095

⌈The FrTp module shall support the Fr N-PDU formats as defined in [1, ISO 10681-2].⌋

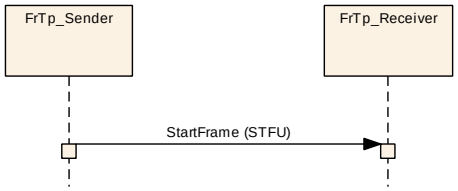
7.2.2 Frame Sequence charts

This chapter describes the data transfer modes based on Transport Protocol Layer frame (N-PDU) sequences according to [1, ISO 10681-2]. This is only for a better understanding of the FrTp internal work and shall not be an additional specification. For a final implementation only the figures in [1] are relevant.

7.2.2.1 Unsegmented unacknowledged data transfer with known message length

[SWS_FrTp_01007] Support unsegmented unacknowledged data transfer with known message length ⌈According to [1, ISO 10681-2] the FrTp module shall support an unsegmented unacknowledged data transfer with known message length as depict in [SWS_FrTp_91001]⌋

[SWS_FrTp_91001] Frame sequence of an unsegmented unacknowledged data transfer with known message length ⌈

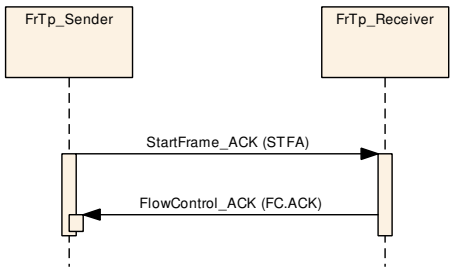


⌋

7.2.2.2 Unsegmented acknowledged data transfer with known message length

[SWS_FrTp_01008] Support unsegmented acknowledged data transfer with known message length ⌈According to [1, ISO 10681-2] the FrTp module supports an unsegmented acknowledged data transfer with known message length as depict in [SWS_FrTp_91002]⌋

[SWS_FrTp_91002] Frame sequence of an unsegmented acknowledged data transfer with known message length ⌈

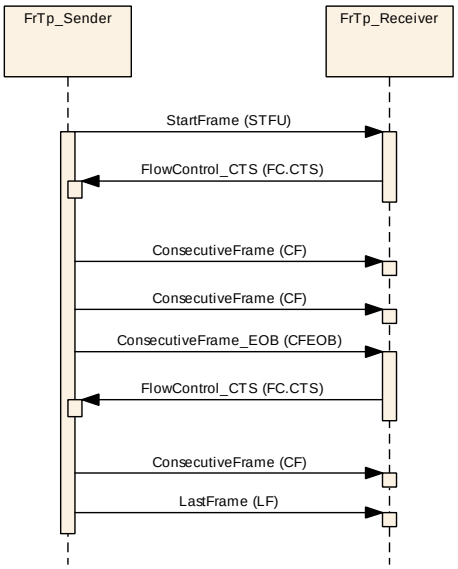


⌋

7.2.2.3 Segmented unacknowledged data transfer with known message length

[SWS_FrTp_01009] Support segmented unacknowledged data transfer with known message length ⌈According to [1, ISO 10681-2] the FrTp module shall support a segmented unacknowledged data transfer with known message length as depict in [SWS_FrTp_91003]⌋

[SWS_FrTp_91003] Frame sequence of a segmented unacknowledged data transfer with known message length ⌈

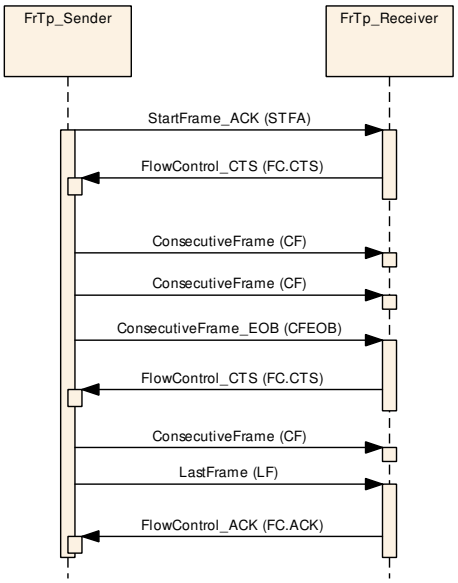


⌋

7.2.2.4 Segmented acknowledged data transfer with known message length

[SWS_FrTp_01010] Support segmented acknowledged data transfer with known message length ⌈According to [1] the FrTp module shall support a segmented acknowledged data transfer with known message length as depict in [SWS_FrTp_91004]⌋

[SWS_FrTp_91004] Frame sequence of a segmented acknowledged data transfer with known message length ⌈

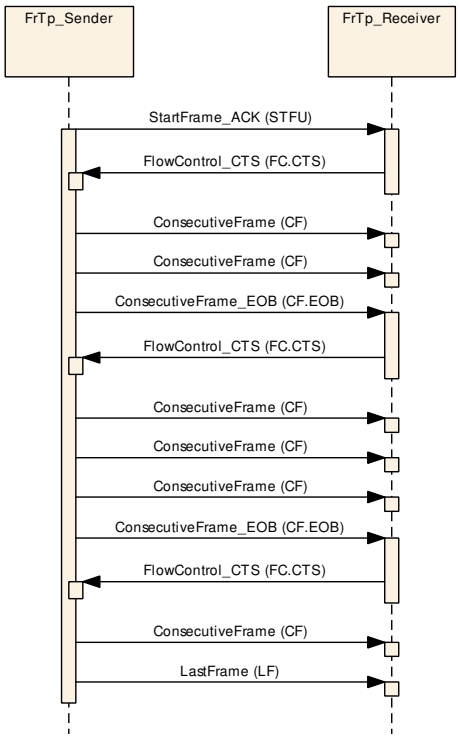


⌋

7.2.2.5 Segmented unacknowledged data transfer with unknown message length

[SWS_FrTp_01011] Support segmented unacknowledged data transfer with unknown message length ⌈According to [1] the FrTp module shall support a segmented unacknowledged data transfer with unknown message length as depict in [SWS_FrTp_91005]⌋

[SWS_FrTp_91005] Frame sequence of a segmented unacknowledged data transfer with unknown message length ⌈



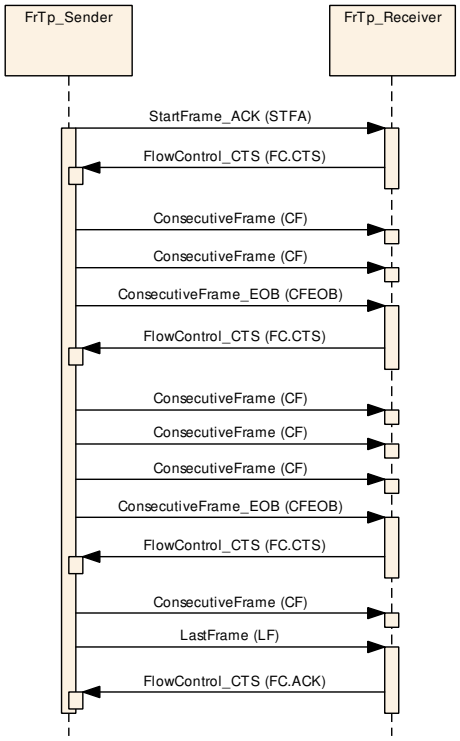
⌋

7.2.2.6 Segmented acknowledged data transfer with unknown message length

[SWS_FrTp_01012] Support segmented acknowledged data transfer with unknown message length ⌈According to [1] the FrTp module shall support a segmented

acknowledged data transfer with unknown message length as depict in [SWS_FrTp_91006]⌋

[SWS_FrTp_91006] Frame sequence of a segmented acknowledged data transfer with unknown message length ⌈



⌋

7.2.3 Limitation to ISO10681-2

The limitations to [1] are described in Table 4.1.

7.3 Internal Module behaviour specification

This chapter specifies the internal behaviour of the FlexRay Transport Layer module to fullfill the protocol behaviour according to [1].

7.3.1 Overview

Figure 7.2 depicts an abstract overview of the FrTp layer module architecture.

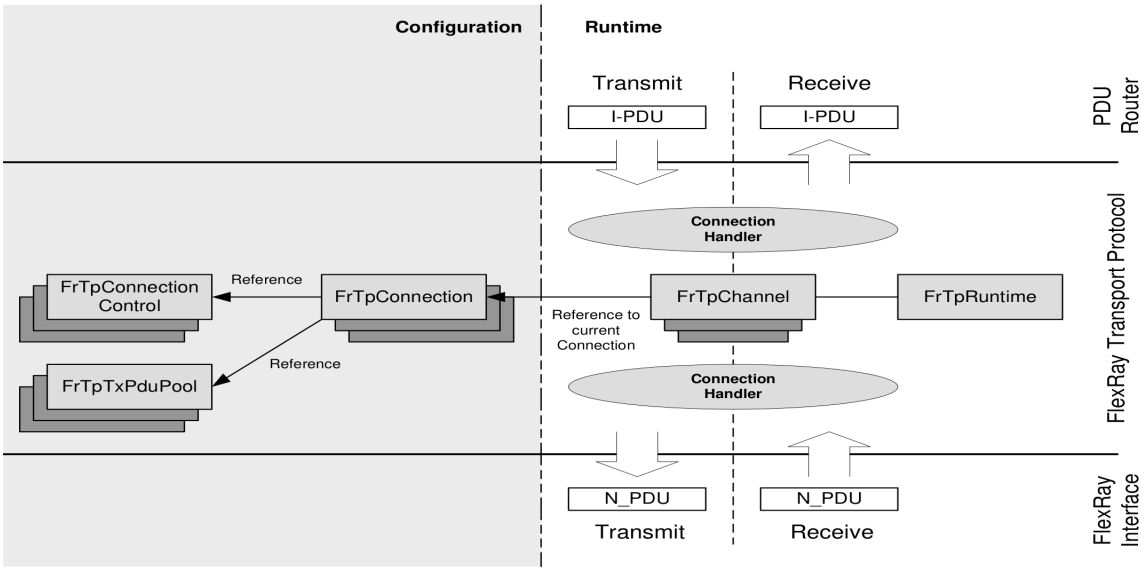


Figure 7.2: FlexRay Transport Module overview

Figure 7.2 depicts a division between configuration parts and runtime parts. After the module is initialized it is able to transmit I-PDUs from an upper layer (PDUR) or receive N-PDUs from a lower layer (FrIf). Below there is a short describes of the different parts being involved in FrTp layer handling procedure.

Term:	Description:
FrTpConnection	A connection is a configuration parameter set which includes all parameters to identify a connection link between different communication nodes uniquely. A connection has a fixed assignment between a sender node (representing by a source address), one or more receiver node(s) (representing by a target address), the upper Layer I-PDU source (representing by an I-PDU-ID). Additionally a connection has a reference to a set of N-PDUs (PDU-Pool) which are defined for sending data via FrTp. A reference to connection specific parameters (e.g. timings and timeouts etc.) is defined too.
FrTpConnectionControl	A connection configuration is a parameter set which includes configuration specific parameter (e.g. timings, timeouts, default parameters etc.). It is referenced by a connection.
FrTpTxPduPool	A Tx-N-PduPool is a set of N-PDUs which are defined for FrTp sending purpose.
FrTpChannel	A channel is a runtime resource of the FrTp module which implements all communication control mechanisms (e.g. state machine etc.) to handle a communication link via FlexRay. A channel could be allocated by the connection handler to process a required connection. Therefore a channel has a reference to the connection which is currently handled by this channel. If a data transfer has been finished the assignment between the channel and the connection is cleared and the channel could be reallocated by another connection.
FrTpRuntime	FrTpRuntime is a set of runtime parameters which is necessary to control active connections. (please refer to Chapter 7.3.3.1)

▽

△

Term:	Description:
Connection Handler	The connection handler is an abstract part of the FrTp module and is responsible for the (re-)allocation of channels and the (re-)assignment of channels and connections.

If an upper layer module (e.g. COM, DCM etc) wants to transmit data (I-PDU) the PduR module executes the corresponding FrTp layer module API call. The connection handler evaluates the I-PDU-ID (equal to N-SDU-ID from FrTp's point of view) for the corresponding connection. The connection handler allocates a free channel and set channel's connection reference to the selected connection. The channel could now initialize with the connection control parameter set which is access able via the references. The channel process the communication until all data have been transmitted. After the last N-PDU was send the connection handler will free the channel and also the reference to the connection is reset.

If an N-PDU is received via FlexRay the FrIf executes the corresponding FrTp API call. The connection handler evaluates the target address and the source address of the N-PDU which is part of the Protocol Control Information (PCI). The connection handler search for the corresponding connection, allocates a channel, set the reference to the selected connection and initialize them. Until the last N-PDU was received the connection handler reallocates the channel, skips the reference to the selected connection and delivers the I-PDU to the addressed upper layer by calling the corresponding PDUR-module API.

7.3.2 Configuration data

7.3.2.1 FrTpConnection

A connection identifies the sender and the receiver(s) of this particular communication link. An FrTp connection link is defined by

- a target address of the receiver node(s) and
- a source address of the sender node.

For the internal handling of different PDUs across the FlexRay communication stack the I-PDU-ID (N-SDU-ID) identifies the data link to upper layer's sender or receiver modules (see Figure 1.2).

Additionally a connection has a reference to a set of N-PDUs (FrTpTxPduPool) which are defined for sending data via this particular connection. A reference to connection specific parameters, e.g. timings and timeouts etc is defined too (FrTpConnection Config).

[SWS_FrTp_01013] FrTpConnection implementation ⌈An FrTpConnection container shall implement all parameters.⌋

[SWS_FrTp_01014] FrTpConnection required per connection link ⌈For each connection link the FrTp module shall handle, a new instance of FrTpConnection container shall be created.⌋

[SWS_FrTp_01015] Post-build support for connections ⌈The FrTp module shall support a post build time configurable number of connections[1].⌋

[SWS_FrTp_01017] Remote / local address pair needs to be unique ⌈Each FrTpConnection shall have a module wide unique RemoteAddress / LocalAddress pair[2].⌋

[SWS_FrTp_01018] N-SDU ID needs to be unique

Upstream requirements: SRS_Fr_05077

⌈Each FrTpConnection shall have a module wide unique FRTP_SDUID (N-SDU ID).⌋

7.3.2.2 FrTpTxPduPool

The FrTpTxPduPool contains a list of N-PDUs configured for sending FrTp N-PDUs. An FrTpTxPduPool could be referenced by different FrTpConnections but each FrTpConnection has exactly one reference to one FrTpTxPduPool. (see also Figure 7.10). The FrTpTxPduPools are necessary to support dynamic bandwidth assignment for connections.

Chapter 7.5.5 describes the dynamic bandwidth assignment in detail. At this position in specification only the term FrTpTxPduPool shall be introduced and some basic requirements to FrTpTxPduPools are specified.

[SWS_FrTp_01019] FrTpTxPduPool implementation ⌈An FrTpTxPduPool container shall implement all parameters.⌋

[SWS_FrTp_01020] Dependency FrTpTxPduPool container to FrTpConnection container ⌈A single FrTpTxPduPool can be referenced by different FrTpConnections.⌋

Note: Configuration of PDU Pools to limit bandwidth to an ECU is described in Chapter 10.4.4.

7.3.2.3 FrTpConnectionControl

An FrTpConnectionControl container contains all static (not runtime) parameters, which are necessary to control a connection e.g. initial timer values, timeout control

[1] Post-build time configurablenumber of connections is required e.g. for gateways. If new connections are defined during vehilce lifecycle only the connection's parameters set has to be updated.

[2] The AUTOSAR local address and remote address is mapped to the [1] source address and target address.

values etc. Each FrTpConnection has an exclusive link to an FrTpConnectionControl container. FrTpConnections with equal control parameters can reference the same FrTpConnectionControl[3].

[SWS_FrTp_01021] FrTpConnectionControl implementation ⌈An FrTpConnectionControl Container shall implement all parameters.⌋

[SWS_FrTp_01022] Dependency FrTpConnectionControl container to FrTpConnection container ⌈An FrTpConnectionControl container can be referenced by different FrTpConnections.⌋

7.3.3 Runtime data

As depicted in Figure 7.2 also some runtime information are required. All runtime information are encapsulated in containers. This chapter defines all the runtime containers with the corresponding variables in that scope as it necessary to understand FrTp's work.

It's recommended to place all runtime data required for implementation into the global FrTpRuntime container too.

7.3.3.1 FrTpRuntime

Module Name:	**FrTpRuntime:**
Module Description:	This container contains the runtime parameters / variables which are necessary to handle FlexRay Transport Protocol communication according to [1].

Included Containers		
Container Name:	**Multiplicity:**	**Scope / Dependency:**
FrTp_Channel	1..*	FrTp Channel: This container contains the runtime parameters / variables for an FrTpChannel.
FrTp_ConCtrlRuntime	1..*	FrTp Connection Control Runtime: This container contains the runtime parameters / variables to handle an FrTpConnection.
FrTp_PduPoolRuntime	1..*	FrTp Pdu Pool Runtime: This container contains the runtime parameters / variables to handle an FrTpPduPool.

[3]Use case: Reducing configuration control container instances

7.3.3.2 FrTpChannel

As described above a channel is a runtime resource of the FrTp. A channel could be allocated to handle a connection. This chapter describes the relevant information of a channel without implementation specific information (e.g. types etc).

Name:	**FrTpChannel:**
Description	This container contains the parameters and variables of a FlexRay channel
Container parameters and variables	

Information:	**Description:**
FrTpChannelNumber	Number of that channel
FrTpTxChannelState	Current state of the Tx channel (idle = 0 or busy = 1)
FrTpTxConState	FrTp Tx Connection State: This parameter implements the current state of the Tx connection (Tx communication state machine according to [1] protocol handling).
FrTpTxConRef	FrTp Tx Connection Reference: This is the reference (pointer to connection) to the current Tx `FrTpConnection`, the channel is currently processing.
FrTpTxConTxPduPendingCounter	FrTp Tx Connection Tx Pdu Pending Counter: This counter counts the number of currently transmitted but not confirmed Fr N-PDUs of the active TxConnection (e.g. SF, CF, LF) This counter shall be incremented each time an FrTp Tx N-PDU is send by the FrTp. This counter shall be decremented each time an transmitted FrTp Tx N-PDU is confirmed by the FrIf.
FrTpTxConTxPduPoolRuntimeRef	FrTp TxConnection Tx PDU Pool Runtime Reference: This is the reference (pointer to FrTp_TxPduPoolRuntime) to the runtime container of the corresponding PDU-Pool. **Note**: The runtime container of the PDU Pool controls the TxConfirmation signals.
FrTpTxConConfigRuntimeRef	FrTp Tx Connection Configuration Runtime Reference: This is the reference (pointer to FrTp_ConConfigRuntime) to the runtime container of the corresponding Tx connection configuration. **Note**: The runtime container of the Tx connection configuration controls the connection parameters, which are changeable during runtime.
FrTpRxChannelState	Current state of the Rx channel (idle or busy)
FrTpRxConState	FrTp Rx Connection State: This parameter implements the current state of the Rx connection (Rx communication state machine according to [1] protocol handling).
FrTpRxConRef	FrTp Rx Connection Reference: This is the reference (pointer to connection) to the current Rx `FrTpConnection`, the channel is currently processing.
FrTpRxConTxPduPendingCounter	FrTp Rx Connection Tx Pdu Pending Counter: This counter counts the number of currently transmitted but not confirmed Fr N-PDUs of the active RxConnection (FlowControl). This counter shall be incremented each time an FrTp Tx N-PDU is send by the FrTp. This counter shall be decremented each time a transmitted FrTp Tx N-PDU is confirmed by the FrIf. Therefore that FrTp Rx Connection Tx Pdu Pending Counter toggles only between 0 and 1.

▽

△

Information:	Description:
FrTpRxConTxPduPoolRuntimeRef	FrTp Rx Connection Tx PDU Pool Runtime Reference: This is the reference (pointer to FrTpTxPduPoolRuntime) to the runtime container of the corresponding PDU-Pool. **Note**: The runtime container of the PDU Pool controls the TxConfirmation signals. This reference is required for the transmission control of FlowControl N-PDU during an ongoing reception on that channel. **Note**: For a full duplex channel configuration (see chapter Chapter 7.3.3.2.1) the FrTpTxConTxPduPoolRuntimeRef and the FrTpRxConTxPduPoolRuntimeRef are equal.
FrTpRxConConfigRuntimeRef	FrTp Rx Connection Configuration Runtime Reference: This is the reference (pointer to FrTpConConfigRuntime) to the runtime container of the corresponding Rx connection configuration. **Note**: The runtime container of the Rx connection configuration controls the connection parameters, which are changeable during runtime.
No included containers	

[SWS_FrTp_00228] Support of multiple TP channels ⌈The FrTp module shall support concurrently work of multiple FrTpChannels[4].⌋

[SWS_FrTp_00088] Number of provided channels configuration ⌈The exact number of provided channels shall be configurable by the parameter `FrTpChanNum`.⌋

[SWS_FrTp_01025] RX channel state to busy when channel is allocated ⌈The runtime variable FrTpRxChannelState shall be switched from "idle" state to "busy" state if the channel is allocated for an Rx connection.⌋

[SWS_FrTp_01026] RX channel state back to idle when RX connection is closed ⌈The runtime variable FrTpRxChannelState shall be switched from "busy" state to "idle" state if the channel is free after an Rx connection is closed.⌋

[SWS_FrTp_01117] Tx channel state busy when channel is allocated ⌈The runtime variable FrTpTxChannelState shall be switched from "idle" state to "busy" state if the channel is allocated for an Tx connection.⌋

[SWS_FrTp_01118] Tx channel state back to idle when TX connection is closed ⌈The runtime variable FrTpTxChannelState shall be switched from "busy" state to "idle" state if the channel is free after an Tx connection is closed.⌋

Note: The error handling for the case if no FrTpChannel resource is available (FrTpTxChannelState != idle) for data transmission is specified in [SWS_FrTp_01041].

[4] The number of channels represents the number of connections, which could be handled concurrently for the same direction. Therefore it is an indication of te performance of the FrTp. On th e other hand a Gateway requires more channels than normal ECUs becasue a gateway handles more concurrent connections. Hence the number of supported channels shall be configurable.

7.3.3.2.1 Full Duplex and Half Duplex

Normally a Full Duplex channel supports concurrent transmission and reception of Fr N-PDUs at the same time for the same[5] connection. Figure 7.3 depicts a Full Duplex implementation.

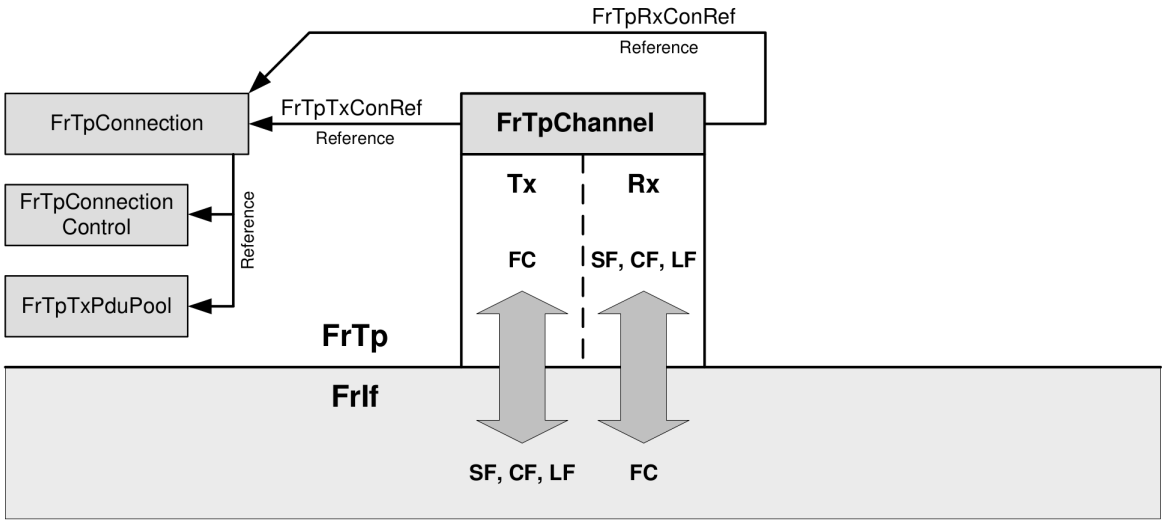


Figure 7.3: Full Duplex Overview

On the other hand a Half Duplex channel supports only a data transfer for one direction. The fact that an Rx transmission has also to send a FlowControl or a Tx transmission has to receive a FlowControl is not similar to a full duplex connection. Figure 7.4 depicts a half duplex FrTp_Channel, where either a Tx or a Rx Connection is processed.

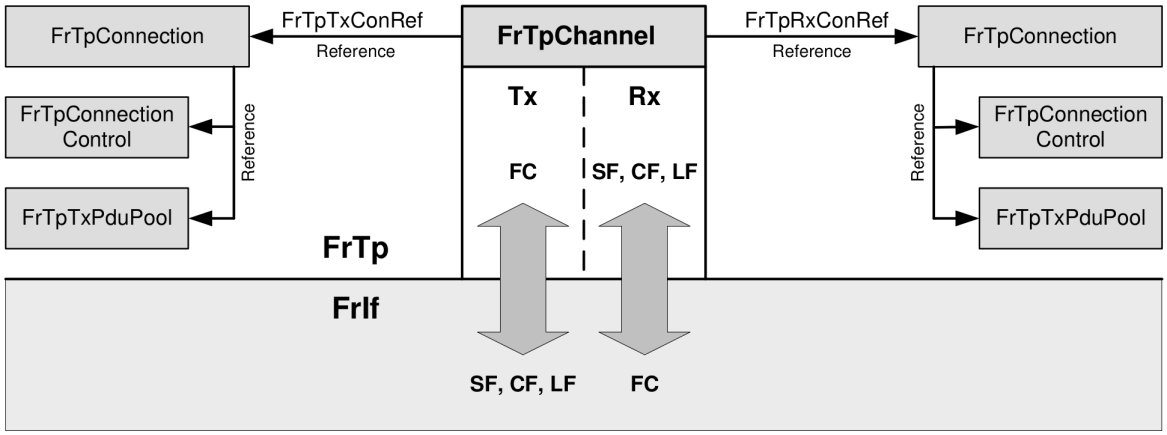


Figure 7.4: Half Duplex Overview

The final functionality of FrTpChannels depends on implementation and therefore it is not specified in this document.

[5]Theoretically it is possible that two ECUs transferring data to each other at the same time. That means that each ECU is sender and receiver concurrently. If both ECUs have only one Remote Address and noe Local Address the FrTp shall evaluate the PCI to distinguish whether a PDU for Rx-Direction (CF or LF) or a PDU for Tx-Direction (FC) was received. This is a full duplex mechanism.

7.3.3.3 FrTpConnectionControlRuntime

This chapter describes the relevant information of Connection Control without implementation specific information (e.g. types etc).

Name:	FrTpConCtrlRuntime:
Description:	FrTp Connection Control Runtime: This container contains the ConnectionControl runtime data.
Container parameters and variables	

Information:	Description:
FrTpSCexpRuntime	FRTP_SEPARATION_CYCLE_EXPONENT Runtime value of FrTpSCexp parameter. This parameter could be changed via API-Call "FrTp_ChangeParameter" and differs than from the configured default value.
FrTpMaxNbrOfNPduPerCycleRuntime	FRTP_MAX_NUMBER_OF_NPDU_PER_CYCLE Runtime value of FrTpMaxNbrOfNPduPerCycle parameter. This parameter could be changed via API-Call FrTp_ChangeParameter and differs than from the configured default value.
No included containers	

7.4 Initialization and shutdown

[SWS_FrTp_01028] FrTp module state definition ⌈The FrTp module shall have two internal states, FRTP_OFF and FRTP_ON.⌋

[SWS_FrTp_01029] FrTp state variable for initialization detection ⌈The FrTp module shall implement a static status variable FrTpState to denote whether the FrTp module is initialized or not[6].⌋

[SWS_FrTp_01106] FrTp state ⌈The FrTpState shall be checked to detect whether FrTp module is initialized or not.⌋

[SWS_FrTp_01030] Default state is OFF after power-up ⌈The FrTp module shall be in the FRTP_OFF state after power up.⌋

[SWS_FrTp_01032] Initialize state to ON after init has been called ⌈The FrTp module shall change to the internal state FRTP_ON when the FrTp has been successfully initialized by the service primitive FrTp_Init.⌋

[SWS_FrTp_01033] Processing of FrTp operations only after initialization ⌈The FrTp module shall perform normal FrTp operation tasks (e.g. segmentation, reassembly etc.) only when the FrTp module is in the FRTP_ON state[7].⌋

[6] This variable is used for development error detection

[7] This requires that FrTp_Init is called before the normal FrTp functionality is used by the COM-Stack.

[SWS_FrTp_01034] Initialization service handling

Upstream requirements: SRS_Fr_05088

⌈The service primitive FrTp_Init shall initialize all global variables of the module and sets all transport protocol connections in a sub-state of FRTP_ON, in which neither transmissions nor receptions are in progress.⌋

[SWS_FrTp_01035] Initialization service clears ongoing connections

Upstream requirements: SRS_Fr_05088

⌈If the FrTp module is in the global state FRTP_ON, a call of the service primitive FrTp_Init shall return the module to an uncritical idle state (idle state = FRTP_ON, but neither transmission nor reception are in progress) and the module shall loose all current connections.⌋

[SWS_FrTp_01036] OFF state after successful execution of shutdown primitive
⌈The FrTp module shall change to the internal state FRTP_OFF when the service primitive `FrTp_Shutdown` has been executed successfully.⌋

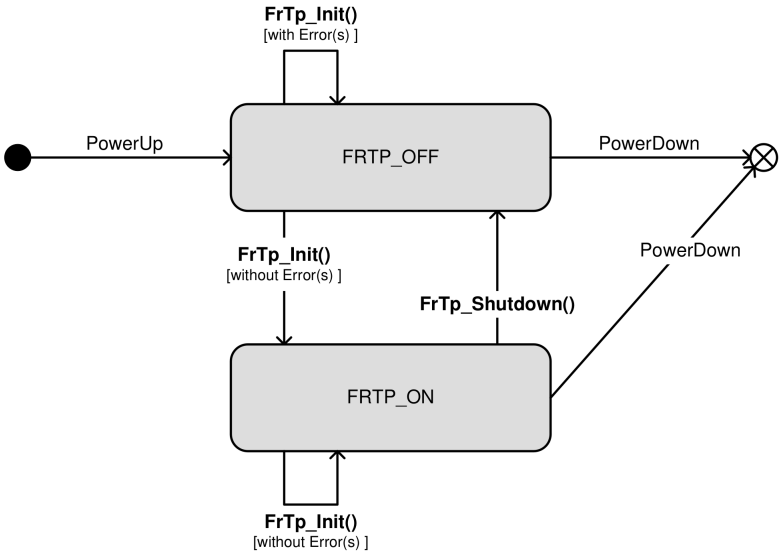


Figure 7.5: FrTp Initialization and shutdown state diagram

7.5 Data Transfer Processing

This chapter covers all topics of FrTp module data transfer processing if transmission of data (Fr N-SDU) is requested by an upper layer (e.g. COM, DCM etc.) via PduR module or if data (Fr N-PDUs) have been received via FrIf module indicated by LSduR. For a better understanding the different topics are encapsulated in several sub-clauses starting with the basic definition of data transfer and reception. Buffer handling is described within an additional chapter.

The FlexRay protocol stack supports two different buffer access modes for data transmission:

- Immediate Buffer Access Mode
- Decoupled Buffer Access Mode

Due to this fact there are two different sequences for data transfer processing.

7.5.1 Flags

The FrTp module uses several flags to signal internal states (see also sequence diagrams in Chapter 9). This chapter describes the flags required for inter-module state handling.

7.5.1.1 TX_SDU_AVAILABLE

The TX_SDU_AVAILABLE flag is set by the service primitive `FrTp_Transmit` to indicate the N-SDU transmit request.

[SWS_FrTp_00415] A TX SDU available flag per channel ⌈The TX_SDU_AVAILABLE flag shall exist for every channel.⌋

[SWS_FrTp_00416] TX SDU available flag use case ⌈The TX_SDU_AVAILABLE flag shall indicate an Fr N-SDU transmit request for a configured connection on an allocated channel.⌋

Note: For detailed information about set and reset conditions and behaviour please refer to Chapter 7.5.2 and [SWS_FrTp_01057].

7.5.1.2 TX_SDU_UNKNOWN_MSG_LENGTH

The TX_SDU_UNKNOWN_MSG_LENGTH flag is set by the service primitive `FrTp_Transmit` to indicate the N-SDU transmit request with an unknown message length. Depending on the status of that flag the FrTp module will recall the service primitive PduR_FrTpCopyTxData several times until all data are transmitted.

[SWS_FrTp_01101] TX SDU unknown message length use case ⌈The TX_SDU_UNKNOWN_MSG_LENGTH flag shall indicate an Fr N-SDU transmit request with unknown message length.⌋

[SWS_FrTp_01102] A TX SDU unknown flag per channel ⌈The TX_SDU_UNKNOWN_MSG_LENGTH flag shall exist for every channel.⌋

Note: For detailed information about set and reset conditions and behaviour please refer to Chapter 7.5.2 and [SWS_FrTp_01124].

7.5.1.3 RX_PDU_AVAILABLE

The RX_PDU_AVAILABLE flag is set by the service primitive `FrTp_RxIndication` to indicate the reception of an N-PDU.

[SWS_FrTp_00418] A RX PDU available flag per RX N-PDU ⌈The RX_PDU_AVAILABLE flag shall exist for every Fr N-PDU, which is configured to be received by the FrTp module.⌋

Note: For detailed information about set and reset conditions and behaviour please refer to Chapter 7.5.3 and [SWS_FrTp_01074].

7.5.1.4 RX_ERROR

The RX_ERROR[8] flag is required in case transmission with acknowledgement and retry is configured. During a segmented data reception an error could occur but sending FlowControl is currently not possible. In that case the information about an error has to be stored until sending a FlowControl is allowed.

[SWS_FrTp_00428] A RX error flag per every FrTp channel ⌈The RX_ERROR flag shall exist for every FrTpChannel.⌋

[SWS_FrTp_00429] RX error flag use case ⌈The RX_ERROR flag shall indicate that an error occurred during a segmented reception.⌋

[SWS_FrTp_00430] RX error flag clearing ⌈The RX_ERROR flag shall be cleared after the reaction (Retry, Negative Acknowledgement, abortion).⌋

7.5.2 Transmit Data

[SWS_FrTp_01204] Usage of meta data information for address information ⌈During transmission, the FrTp shall use addressing information provided by the upper layer via the meta data items SOURCE_ADDRESS_16 and TARGET_ADDRESS_16 as local address and remote address of the transmitted N-PDUs and to identify received flow control N-PDUs.⌋

[SWS_FrTp_01205] Usage of configured address information in transmit request ⌈If `FrTpLa` and/or `FrTpRa` are configured for a transmitted N-SDU, they are used even when the addressing information is the provided by the upper layer. If not, the address information in the N-PDUs shall be set according to the provided address information.⌋

[8] Refer [1, ISO-10681-2] chapter 7.5.7.2.3.

7.5.2.1 Transmit Data via 'Immediate Buffer Access' Mode

This chapter defines a data transfer requested by an upper layer (e.g. COM, DCM etc.) via 'Immediate Buffer Access' Mode.

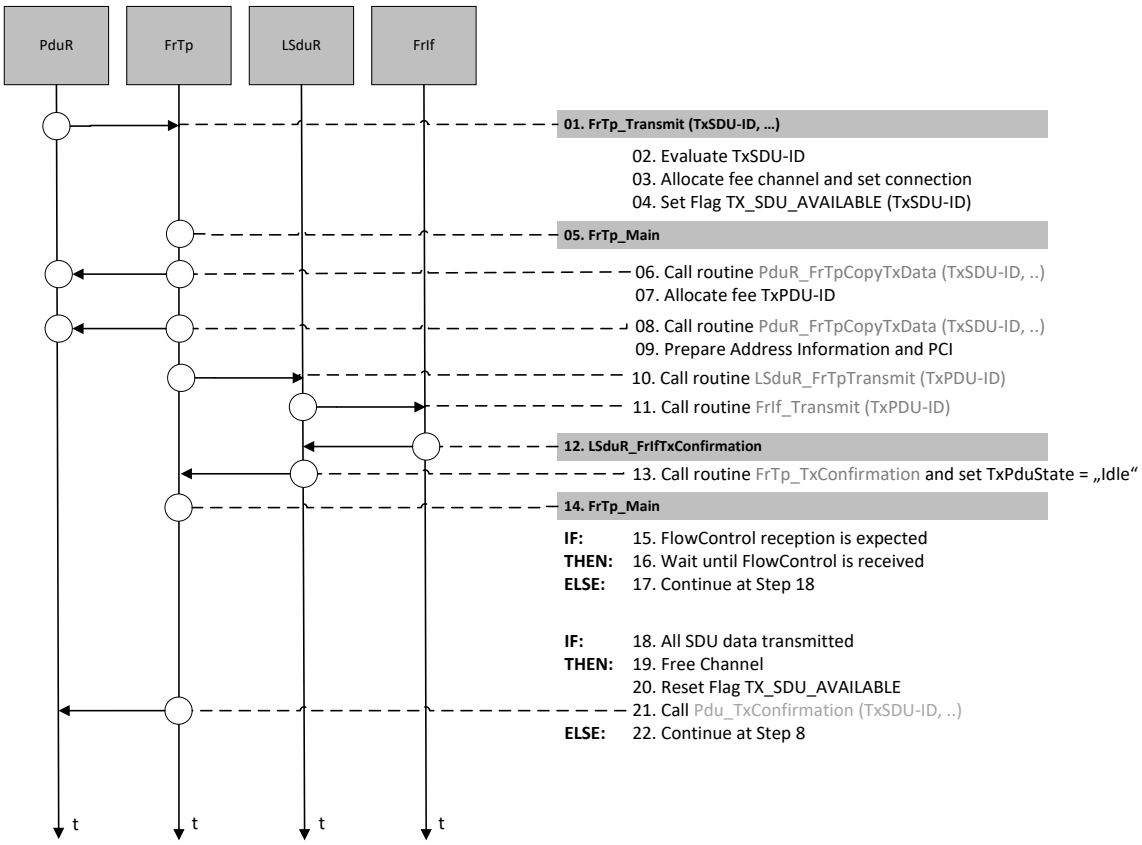


Figure 7.6: in 'Immediate Buffer Access' mode

Figure 7.6 depicts the internal processing for data transmission in principle[9] if "Immediate Buffer Access" mode is configured[10]. Below there is a description of the different steps which are necessary to transmit data via FrTp.

Step 1 - 4

[SWS_FrTp_00136] FrTp transmit initiation by upper layer ⌈Sending Fr N-SDU data shall always be initiated by the service primitive call of `FrTp_Transmit`.⌋

[SWS_FrTp_01043] Detection for transmission with unknown or known message length ⌈The FrTp module shall evaluate the value of `PduInfoType`.SduLength:

[9] Figure 7.6 depicts only the overview of data transmission for a single channel without further functionality (e.g. transmit cancellation) or internal behaviour (e.g. internal control data handling, return values, etc.). Also a schedule for data transfer in concurrent channels is not described here.

[10] Buffer access mode is configured for each N-PDU and is referenced via the PDU-Pool.

- SduLength = 0: Transmission with unknown message length is requested
- SduLength $\neq$ 0: Transmission with known message length is requested.

⌋

[SWS_FrTp_01044] Handling of TX SDU unknown message length flag ⌈If support unknown message length is configured the FrTp module shall set the flag TX_SDU_UNKNOWN_MSG_LENGTH according to the result of [SWS_FrTp_01043].⌋

[SWS_FrTp_01134] Handling of development error for unknown message length ⌈The FRTP module shall raise an development error `FRTP_E_UMSG_LENGTH_ERROR` when

1. a transmission with unknown message length is detected and
2. support of unknown message legth is not configured and
3. development error detection is enabled for the FrTp module.

⌋

The service primitive parameter TxPduId shall be used to select the correct connection. This shall be done by searching for the correct entry within the `FrTpConnection` container (`FrTpConnection`.FrTpTxSduId). If a valid parameter TxPduId is given, the FrTp module shall search for a free channel resource (FrTpTxChannelState = Idle) and allocate them to control the requested Tx data transfer.

[SWS_FrTp_01038] TX SDU available flag ⌈An ongoing data transfer shall be signalled by the flag TX_SDU_AVAILABLE.⌋

[SWS_FrTp_01039] Handling of TX SDU available flag ⌈The service primitive shall set the flag TX_SDU_AVAILABLE only, if

- the requested TxPduId is valid
- a free channel resource is available (FrTpTxChannelState = Idle)

⌋

[SWS_FrTp_01040] Reject transmission request when TX PDU ID is not valid ⌈If the current parameter TxPduId is not supported the service primitive `FrTp_Transmit`

- shall be terminated and the return value shall be set to `E_NOT_OK` and
- the FrTp module shall raise an development error `FRTP_E_INVALID_PDU_SDU_ID` when development error detection for the FrTp module is enabled.

⌋

[SWS_FrTp_01041] Reject transmission request when no free channel is available ⌈If no free channel is available (FrTpTxChannelState != Idle) the service primitive FrTp_Transmit shall be terminated and the return value shall be set to E_NOT_OK[11].⌋

[SWS_FrTp_01185] Runtime error FRTP_E_NO_CHANNEL ⌈The FrTp module shall raise a runtime error FRTP_E_NO_CHANNEL.⌋

[SWS_FrTp_01200] Reject transmission request when another transmission for the specified address information is active ⌈If the request for a message transmission has not been accepted due to another transmission for the specified address information is active; then FrTp_Transmit shall return E_NOT_OK.⌋

Step 5 - 10

If the TX_SDU_AVAILABLE flag is set, the FrTp module has to evaluate the length of the currently available FrTp N-SDU data by calling the service primitive PduR_FrTpCopyTxData()[12] a first time. With knowledge of the available data size FrTp module scans the FrTpTxPduPool and allocates the first free FrTpPdu for that data transfer. Depending on the available FrTp-N-SDU length and the FrTpTxPduPool's free FrTpPdu length the FrTp module decides whether segmentation is necessary or not for that N-SDU transfer. By calling the service primitive PduR_FrTpCopyTxData the data shall be copied to the corresponding buffer. In a next step the corresponding Address Information and PCI are prepared and the service primitive LSduR_FrTp-Transmit is called with the corresponding TxPduId.

[SWS_FrTp_01042] Start copy TX data when SDU available flag is set ⌈If the TX_SDU_AVAILABLE flag is set, the FrTp module shall call the service primitive PduR_FrTpCopyTxData to get the currently available FrTp N-SDU Length information.⌋

[SWS_FrTp_01045] Allocate first free TX PDU in a pool ⌈The FrTp module shall always allocate the first free FrTpTxPdu while scanning the corresponding FrTpTx-PduPool.⌋

[11] This scenario could occur on gateways, if communication via more connections is requested than channels resources are configured.

[12] The available data length evaluation depends on different scenarios the FrTp is used in.
– In case of a normal ECU which transfers data with unknown message lengths it is necessary to evalueate the length of the currently stored FrTp-N-SDU.
– In case an ECU transmits data with known message length the Tx data length is given by the parameter of the FrTp_Transmit service primitive. An additional evaluation by calling PduR_FrTpCopyTxData(..) is possible to have equal evaluation sequences for known and unknown message length transfers but could be skipped by runtime optimizations (implementation dependency).
– In case of a Gateway which routes N-SDUs of different bus systems it is necessary to get the currently available (received) data length in the gateway buffer.

[SWS_FrTp_01046] Usage of free TX PDU to continue transmission ⌈If a free `FrTpTxPdu` is identified, the FrTp module shall use this `FrTpTxPdu` to continue current transmission process.⌋

[SWS_FrTp_01047] Postpone processing of transmission in case no free TX PDU is available ⌈If no free `FrTpTxPdu` is identified, the FrTp module shall postpone processing for the corresponding connection till the next main function call.⌋

Note: FrTp module shall postpone the processing for the corresponding connection until either free `FrTpTxPdu` is identified according to [SWS_FrTp_01046] or a timeout occurs (see Chapter 7.5.8).

[SWS_FrTp_01048] Segmentation decision for N-SDU transfer ⌈The FrTp module shall decide whether segmentation for the requested N-SDU transfer is required or not depending on the length information of the first allocated `FrTpTxPdu` from an `FrTpTxPduPool` for the currently processed `FrTpConnection`.⌋

Note: Decision of segmentation as specified in [SWS_FrTp_01048] is shown in Figure 7.7.

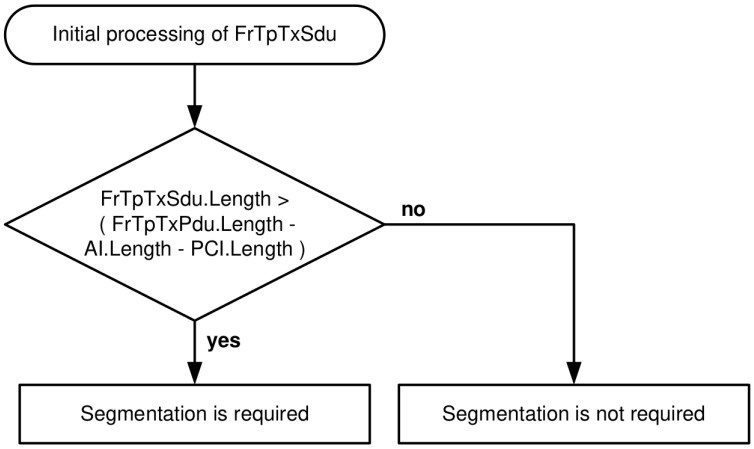


Figure 7.7: Segmentation decision

Note: The decision whether segmentation is possible or not depends also on the connection mode (1:1 or 1:n).

[SWS_FrTp_01123] Call copy TX data with according length ⌈The FrTp module shall call the service primitive PduR_FrTpCopyTxData to copy the currently available FrTp N-SDU data with a length of `FrTpTxPdu` length to the corresponding transmit buffer.⌋

[SWS_FrTp_01049] Address Information and PCI preparation ⌈The FrTp module shall prepare the Address Information and PCI according to the result of [SWS_FrTp_01048] as defined in specification [1].⌋

[SWS_FrTp_01050] Initiate transmission with according TxPduId ⌈The FrTp module shall initiate an N-PDU data transfer by calling the service primitive `LSduR_FrTpTransmit` with the TxPduId of the recently allocated `FrTpTxPdu`.⌋

[SWS_FrTp_01051] Transmit data length handling ⌈The FrTp shall set the corresponding data length referenced by the service primitive `LSduR_FrTpTransmit`'s parameter `PduInfoType` to the exact data length of the buffer[13].⌋

Step 11 - 12

If the N-PDU was successfully transmitted by the FrIf module, the FrIf module shall call the service primitive `LSduR_FrIfTxConfirmation` with result `E_OK`. The LSduR forward the transmission confirmation by calling `FrTp_TxConfirmation`. Within this service primitive the FrTp module shall reset the state of the corresponding `FrTpTxPdu`.

[SWS_FrTp_01052] TX confirmation call by lower layer ⌈The service primitive `FrTp_TxConfirmation` shall be called by the underlying layer module after a successful or failed transmission of the corresponding N-PDU.⌋

[SWS_FrTp_01053] TX confirmation call by lower layer ⌈The service primitive `FrTp_TxConfirmation` shall be called by the underlying layer module with the corresponding `FrTpTxConfirmationPduId` of the successful transmitted PDU.⌋

[SWS_FrTp_01054] N-PDU state idle after TX confirmation ⌈The service primitive `FrTp_TxConfirmation` shall reset the state of the corresponding `FrTpTxPdu` (N-PDU) to "idle".⌋

Step 13 - 16

Depending on ISO-10681-2 protocol handling in some cases a response N-PDU (Flow Control) from the receiver is expected by the sender. Hence the sender has to wait until the response N-PDU (Flow Control) is received and continue processing after reception.

[SWS_FrTp_01055] FrTp timing behaviour ⌈The FrTp module shall implement a timing and timeout behaviour as defined in [SWS_FrTp_01099].⌋

[13] FrTp transmits always the real amount of data stared in the corresponding buffer. FrTp is not responsible for fill bytes. Fill up N-SDUs to a configured frame size is done within lower layers (e.g. FrIf or FlexRay Driver). The FrTp only decides whether segmentation is necessary or not and to segment N-PDU Consecutive Frames to the maximum length of the corresponding PDU of the PduPool.

Step 17 - 21

If all N-SDU data are transmitted the FrTp module shall free all allocated resources and reset all flags which signals an ongoing data transfer for this connection. If the data transmission is pending, the FrTp module shall continue the data transfer at step 6.

[SWS_FrTp_01056] Free allocated TX Channel state after transmission finished ⌈The FrTp module shall free the allocated channel (FrTpTxChannelState = Idle) if

- all Tx N-SDU data are transmitted and
- `FrTp_TxConfirmation` was given with result `E_OK` and
- the final acknowledge is received in case acknowledge is configured.

Or

- `FrTp_TxConfirmation` was given with result `E_NOT_OK`.

⌋

[SWS_FrTp_01057] Reset handling for SDU available flag ⌈The FrTp module shall reset the flag TX_SDU_AVAILABLE, if:

- all N-SDU data are transmitted and
- `FrTp_TxConfirmation` was given with result `E_OK` and
- the final acknowledge is received in case acknowledge is configured.

Or

- `FrTp_TxConfirmation` was given with result `E_NOT_OK`.

⌋

[SWS_FrTp_01124] Reset handling for TX SDU with unknown message length flag ⌈The FrTp module shall reset the flag TX_SDU_UNKNOWN_MSG_LENGTH, if:

- all N-SDU data are transmitted and
- `FrTp_TxConfirmation` was given with result `E_OK` and
- the final acknowledge is received in case acknowledge is configured or
- if transmission was aborted or `FrTp_TxConfirmation` was given with result `E_NOT_OK`.

⌋

[SWS_FrTp_01058] TX confirmation handling ⌈The FrTp module shall always call the service primitive PduR_FrTpTxConfirmation for the corresponding `FrTpTxSdu` ID after the transmission request was accepted.

The result shall be `E_OK` if

- all N-SDU data are transmitted and
- FrTp_TxConfirmation was given with result E_OK and
- the final acknowledge is received in case acknowledge is configured.

Otherwise the result shall be E_NOT_OK.⌋

[SWS_FrTp_01198] Abort transmission in case of FC with invalid FS or FS with OVFLW or ABT ⌈If an FC frame is received with an invalid FS or with FS set to OVFLW or ABT, the FrTp module shall abort the transmission of this message and notify the upper layer by calling the callback function PduR_FrTpTxConfirmation with the result E_NOT_OK.⌋

[SWS_FrTp_01199] Abort transmission in case of FC with invalid BP ⌈If an FC frame is received with the FS set to ACK_RET, where the BP points to a position outside the buffer of the sender, then the FrTp module shall abort the transmission of this message and notify the upper layer by calling the callback function PduR_FrTpTxConfirmation with the result E_NOT_OK.⌋

7.5.2.2 Transmit Data via 'Decoupled Buffer Access' Mode

Figure 7.8 depicts the internal processing for data transmission in principle[14] if "Decoupled Buffer Access" mode is configured[15]. Below there is a description of the different steps which are necessary to transmit data via FrTp.

[14] Figure 7.8 depicts only the overview of data transmission for a single channel without further functionality (e.g. transmit cancellation) or internal behaviour (e.g. internal control data handling, return values etc.). Also a schedule for data transfer in concurrent channels is not described here.

[15] Buffer access mode is configured for each N-PDU (refer to FrIf) and is referenced via the PDU-Pool.

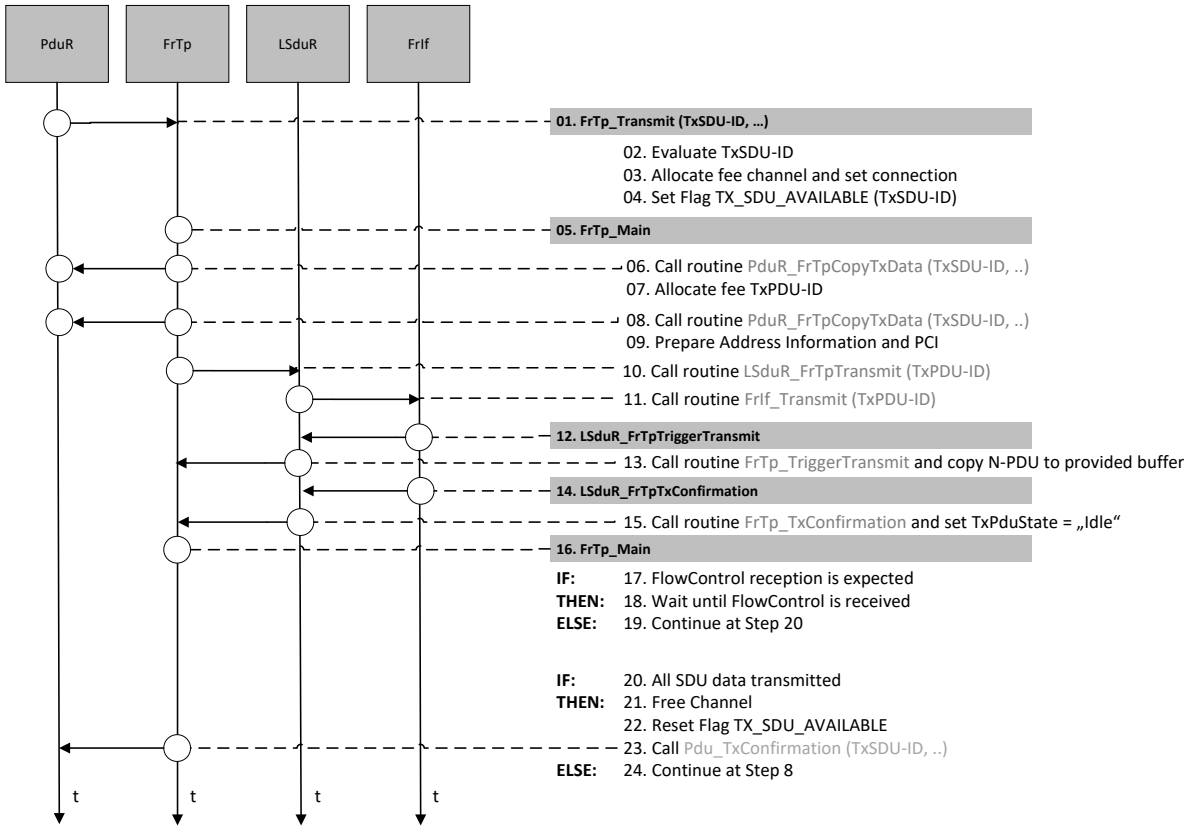


Figure 7.8: Transmit data overview in ’Decoupled Buffer Access’ mode

Step 01 - 10

Step 01 - 10 in ’decoupled access mode’ are equal to step 01 - 10 in ’immediate access mode’. Please refer to Chapter 7.5.2.1.

For step 09 it is recommended to set the address information and PCI to a local buffer because the service primitive `FrTp_TriggerTransmit` is called in interrupt mode and therefore the processing time to copy the complete N-PDU (Address Information, PCI and (part of) SDU data) shall be as short as possible.

Step 11 - 12

[SWS_FrTp_01059] Trigger transmit call by lower layer ⌈The service primitive `FrTp_TriggerTransmit` shall be called by LSduR, which was originated by the FrIf module via a call of `LSduR_FrIfTriggerTransmit` to propagate FrTp N-PDUs to the lower layers e.g. FlexRay Driver).⌋

[SWS_FrTp_01060] Handling of trigger transmit ⌈The service primitive `FrTp_TriggerTransmit` shall copy the Address Information, PCI and N-SDU data to the corresponding buffer, which is referenced by the service primitive parameter `PduInfoType`.⌋

[SWS_FrTp_01061] Data length handling for trigger transmit ⌈The service primitive `FrTp_TriggerTransmit` shall set the corresponding data length referenced by the service primitive parameter `PduInfoType` to the exact data length of the buffer.⌋

Step 13 - 23

Steps 13 - 23 in 'decoupled access mode' are equal to step 10 - 20 in 'immediate access mode'. Please refer to Chapter 7.5.2.1.

7.5.2.3 Data Transfer with unknown message length

ISO10681-2 supports the possibility to transmit data with an unknown message length.

[SWS_FrTp_01062] Configurable transmission with unknown message length ⌈The functionality to support data transmission with unknown message length shall be configurable by compiler switch.⌋

[SWS_FrTp_01063] Reject transmissions with unknown length if connection is configured for multiple receivers ⌈If a 1:n connection is configured (parameter FRTP_MULTIPLE_RECEIVER_CON is set), a Data transfer with unknown message length shall not be processed[16] and the service primitive call `FrTp_Transmit` shall be rejected with the return value `E_NOT_OK`.⌋

[SWS_FrTp_01187] Runtime error FRTP_E_SEG_ERROR handling ⌈The FrTp module shall raise a runtime error `FRTP_E_SEG_ERROR` when a 1:n connection is requested as described in [SWS_FrTp_01063].⌋

[SWS_FrTp_01064] Start of transmission with unknown length ⌈An upper layer's data transmission with unknown message length shall be initiated by an calling the service primitive `FrTp_Transmit` with the service primitive parameter PduLength = 0 ('zero').⌋

[SWS_FrTp_01065] Length parameter handling for transmissions with unknown length ⌈During an ongoing data transfer with unknown message length the service primitive parameter Length of the service primitive PduR_FrTpCopyTxData shall be set to the value of the currently stored Tx-Buffer's data bytes.⌋

[16] Unknown message length data transfer requires segmentation because at least a StartFrame and a LastFrame have to be transmitted.

[SWS_FrTp_01066] Finish condition of a transmission with unknown length ⌈An ongoing upper layer's data transmission with unknown message length shall be finished, if the parameter length within the service primitive is set to 0 ('zero').⌋

[SWS_FrTp_01067] PDU length handling ⌈The FrTp module shall add all `PduInfoType`.PduLength values to calculate the total message length which is transmitted by the LastFrame (LF).⌋

7.5.2.3.1 Segmentation condition for data transfer

[SWS_FrTp_01068] Segmentation condition for data transfer with multiple receivers ⌈If the `FrTpConnectionControl` parameter FRTP_MULTIPLE_RECEIVER_CON for the corresponding `FrTpConnection` is set, the communication handler shall not process a segmentation of an N-SDU and take the following actions:

- shall raise a runtime error `FRTP_E_SEG_ERROR` and
- shall abort the transmission of this message and notify the upper layer by calling the callback function PduR_FrTpTxConfirmation with the result `E_NOT_OK`.

⌋

7.5.3 Receive Data

This chapter defines a data reception on FrTp module via the LSduR requested by the lower layer FlexRay Interface (FrIf).

[SWS_FrTp_01206] Forward addressing information within meta data ⌈During reception, the FrTp shall forward addressing information received as local address and remote address in the N-PDU to the upper layer via the meta data items SOURCE_ADDRESS_16 and TARGET_ADDRESS_16, and shall use the same address information when transmitting flow control N-PDUs.⌋

[SWS_FrTp_01207] Addressing information handling ⌈If `FrTpLa` and/or `FrTpRa` are not configured for a received N-SDU, any received addressing information can be assigned to this N-SDU. N-SDUs with configured `FrTpLa` and/or `FrTpRa` shall be preferred during reception over those without these configuration parameters.⌋

Note: The service routine PduR_FrTpStartOfReception shall be called in either `FrTp_MainFunction` or `FrTp_RxIndication`.

Figure 7.9 depicts the internal processing for data reception in principle[17]. Below there is a description of the different steps which are necessary to receive data via FrTp.

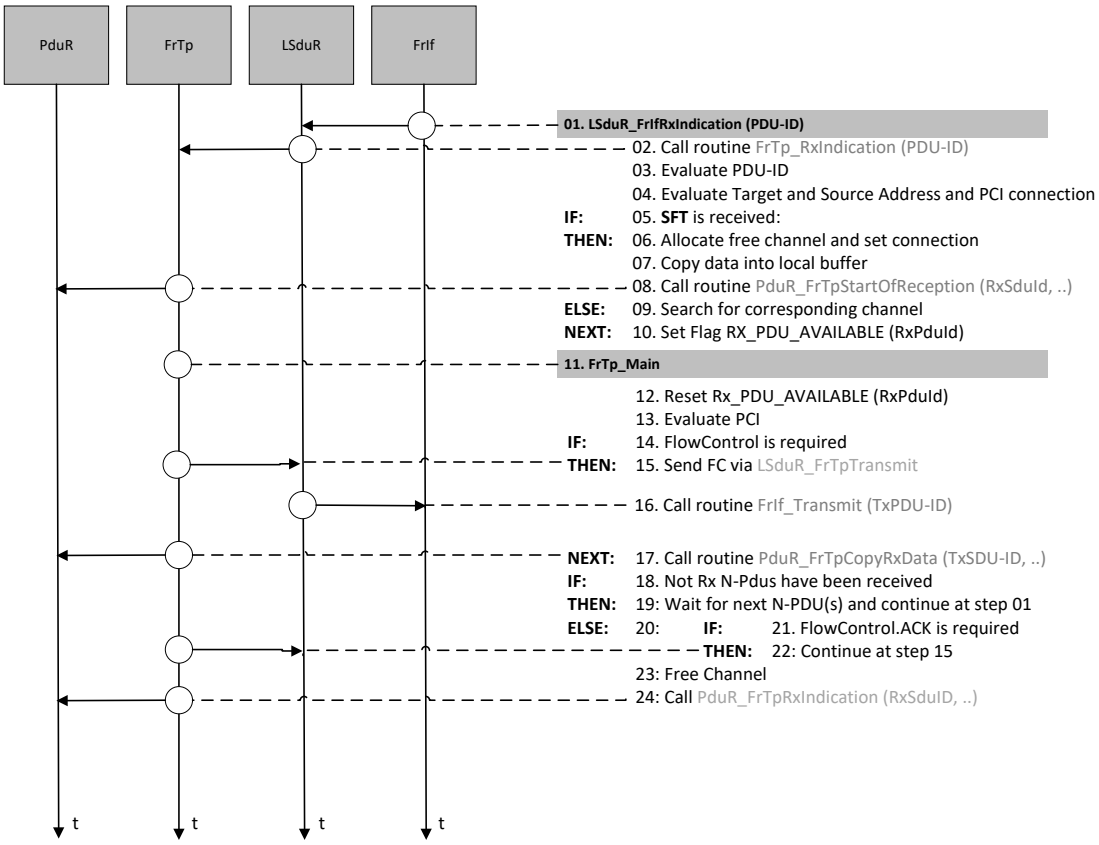


Figure 7.9: Receive data overview (`FrTp_RxIndication` shall call PduR_FrTpStartOfReception routine)

Step 1 - 9

[SWS_FrTp_00137] Reception service ⌈Receiving shall be initiated by the service primitive call `FrTp_RxIndication`.⌋

The FrTp module shall validate the RxPduId. If an invalid RxPduId is received, the service primitive `FrTp_RxIndication` is terminated. For a valid RxPduId the FrTp module evaluates the target and source address and selects the corresponding `FrTp_Connection`. If a Startframe (STF) is received a free FrTpChannel resource (FrTpRxChannelState = Idle) has to be allocated for that connection. A call of the service primitive PduR_FrTpStartOfReception signals a new data reception to the upper layer.

[17] Figure 7.9 depicts only an overview of data reception for a single channel without further functionality (e.g. transmit cancellation) or internal behaviour (e.g. internal control data handling, return values etc.). Also a schedule for data transfer in concurrent channels is also not described here.

If a consecutive frame (CF) or a last frame (LF) has been received, the corresponding channel has to be evaluated and the Rx_PDU_AVAILABLE flag shall be set.

[SWS_FrTp_01069] Processing condition for received N-PDU ⌈The FrTp module shall process a received FrTp N-PDU only if

- a valid RxPduId is received

and

- the FrTp N-PDU's address information matches to the configured FrTpConnection address information.

⌋

[SWS_FrTp_01070] Handling in case of invalid received PDU ID ⌈If an invalid (undefined) RxPduId is received the FrTp module shall

- ignore the FrTp N-PDU and
- shall raise an development error FRTP_E_INVALID_PDU_SDU_ID when development error detection for the FrTp module is enabled.

⌋

[SWS_FrTp_01071] Conditions for matching FrTpConnection for a received N-PDU ⌈A matching FrTpConnection is only identified if

- the received FrTp N-PDU's "Target Address" (see [1]) is equal to the configured FrTpConnection's Local Address (FrTpLa) **and**
- the received FrTp N-PDU's "Source Address" (see [1]) is equal to the configured FrTpConnection's Remote Address (FrTpRa).

⌋

[SWS_FrTp_01072] Ignore received N-PDU if address check fails ⌈If the address check doesn't match to any configured FrTpConnection the received FrTp N-PDU shall be ignored.⌋

[SWS_FrTp_01074] RX PDU available flag handling ⌈The service primitive shall set the flag RX_PDU_AVAILABLE only, if

- the requested RxPduId is valid and
- the address check matches to a configured FrTpConnection and
- a free channel resource is available (FrTpRxChannelState = Idle)

⌋

[SWS_FrTp_01075] Abort reception in case of invalid parameters ⌈If the current parameter RxPduId is not supported the service primitive `FrTp_RxIndication` shall be terminated without any further action[18].⌋

[SWS_FrTp_01076] Abort reception handling when no free channel is available ⌈If no free channel is available (FrTpRxChannelState **!=** Idle) the service primitive `FrTp_RxIndication` shall be terminated without any further action[19].⌋

[SWS_FrTp_01186] Runtime error FRTP_E_NO_CHANNEL ⌈The FrTp module shall raise a runtime error `FRTP_E_NO_CHANNEL`.⌋

[SWS_FrTp_01077] Store StartFrame locally ⌈Within the service primitive `FrTp_RxIndication` the FrTp module shall copy the received StartFrame PDU into a local buffer[20].⌋

[SWS_FrTp_01078] Provision of data and size during start of reception ⌈If a new connection is established, the FrTp module shall call the service primitive PduR_FrTpStartOfReception with the corresponding `FrTpRxSdu` ID and the expected data length to indicate start of data reception for an upper layer.⌋

[SWS_FrTp_01193] Provision of data and size during start of reception ⌈With the call of PduR_FrTpStartOfReception, the FrTp shall provide the data and size of STF to the upper layer via info parameter of PduR_FrTpStartOfReception.⌋

Step 10 - 14

According to [1] protocol (evaluate PCI) it is possible that a received N-PDU requires an N-PDU response (e.g. FlowControl). In that case the FrTp module shall allocate the first free N-PDU from the referenced PDU pool, prepare the response and initiate the transmission process by calling `LSduR_FrTpTransmit` with the corresponding TxPduId, which is forwarded by LSduR as a service primitive call of `FrIf_Transmit`. After transmission the FrTp module shall wait for reception of consecutive N-PDUs. If no N-PDU response is required by protocol the FrTp module shall continue reception handling.

[SWS_FrTp_01080] N-PDU response handling ⌈If transmission of an N-PDU response is required by ISO10681-2 protocol handling, the FrTp module shall send the corresponding N-PDU (e.g. FlowControl) to the initial sender node.⌋

[18] If DET is active a corresponding error shall be set

[19] It is not possible to signal that temporary resource lack to the upper layer because "PduR_FrTpStartOfReception" provides no parameter for that case.

[20] Only the received StartFrame PDU shall be stored temporary in a local buffer. This is necessary in case of a gateway has temporary no free resources for that frame. The correct protocol and timing behaviour is ensured if FrTp sends a FlowControl PDU after a free channel was allocated.

Step 15

[SWS_FrTp_01079] Handling of received N-SDUs ⌈The FrTp module shall extract the N-SDU data from the received N-PDU data according to [1].⌋

[SWS_FrTp_01138] Copy process of received N-SDUs ⌈The FrTp module shall initiate the copy process of the received N-SDU (fragment) by calling the service primitive PduR_FrTpCopyRxData[21].⌋

[SWS_FrTp_00421] RX PDU available flag cleared after processing finished ⌈The RX_PDU_AVAILABLE flag shall be cleared when finished processing the Fr N-PDU.⌋

Step 16 - 17

The FrTp module could calculate whether all N-PDUs of an N-SDU are received. If the communication is still ongoing the FrTp module shall continue data reception at step 01.

Step 18 - 22

If all FrTp Rx-PDUs of a complete N-SDU transmission have been received the FrTp module shall send an Acknowledgement if required and free the allocated channel ressource. In a next step the FrTp module shall call the service primitive PduR_FrTpRxIndication with the corresponding `FrTpRxSdu` ID to signal upper layers that an N-SDU has been received.

[SWS_FrTp_01081] Free allocated channels after successful reception ⌈The FrTp module shall free the allocated channel (FrTpRxChannelState = Idle) if

- all N-SDU data are received and
- all required response N-PDUs (FlowControl) are transmitted and TxConfirmation was given with result `E_OK`.

Or

- `FrTp_TxConfirmation` was given with result `E_NOT_OK`.

⌋

[SWS_FrTp_01083] Successful reception notification handling ⌈The FrTp module shall always call the service primitive PduR_FrTpRxIndication after PduR_FrTpStartOfReception succeeded. The result shall be `E_OK` if

- all N-SDU data are received and

[21] The procedure is also used for the "Routing-On-The-Fly" behaviour for gateways.

- all required response N-PDUs (FlowControl) are transmitted and TxConfirmation was given.

⌋

7.5.3.1 Receive with unknown message length

The FrTp according to [1] provides a method to receive data with unknown message length.

[SWS_FrTp_01184] Handling of data reception with unknown message length ⌈If a data reception with unknown message length shall be established, the FrTp shall call the API PduR_FrTpStartOfReception () with an expected data length of zero ("0").⌋

Note: If the API PduR_FrTpStartOfReception () is called with a data length of zero ("0") the upper modul shall provide the maximum buffer size that is currently available.

ECU scenario:
Upper layer, e.g. DCM, shall provide the currently available maximum buffer size.

Gateway szenario:
PduR module shall provide the currently available maximum buffer size.

7.5.4 Buffer Handling

The FrTp module handles received/transmitted data one frame at a time.

During reception it forwards data received from the FrIf via the LSduR directly to the upper layer, no buffering is involved.

During transmission in case of immediate buffer access mode it must provide a temporary buffer to the upper layer which is then directly forwarded via the LSduR to FrIf.

In case of decoupled buffer access mode a static buffer per connection has to be provided to the upper layer and be kept until TriggerTransmit occurs.

The service primitives used to request the upper layer to copy the data from/to the buffers provided by FrTp are: PduR_FrTpCopyTxData and PduR_FrTpCopyRxData.

[SWS_FrTp_01196] Retry handling when TX buffer is busy ⌈If the call for a TxBuffer (service primitive PduR_FrTpCopyTxData) does not provide a valid buffer and if service primitive notification result type value is `BUFREQ_E_BUSY`, then FrTp module shall try up to `FrTpTimeCs` times to get a valid buffer.⌋

[SWS_FrTp_01197] Abort transfer when no more TX buffer is available ⌈If the call for a TxBuffer (service primitive PduR_FrTpCopyTxData) does not provide

a valid buffer (when FrTpTimeCs expired) and if service primitive notification result type value is BUFREQ_E_NOT_OK, then the transfer shall be aborted by calling PduR_FrTpTxConfirmation with E_NOT_OK.⌋

7.5.4.1 Buffer Access Mode

[SWS_FrTp_01084] Support of immediate and decoupled buffer access ⌈For Tx direction the FlexRay Transport Protocol Layer shall support

- "Immediate Buffer Access" mode and
- "Decoupled Buffer Access" mode.

⌋

7.5.5 Dynamic Bandwidth Assignment

From FrTp's point of view physical FlexRay bandwidth is represented by N-PDUs. As depicted in Figure 7.10 there is a direct mapping between N-PDUs and L-PDUs (done within FrIf module's frame construction plan).

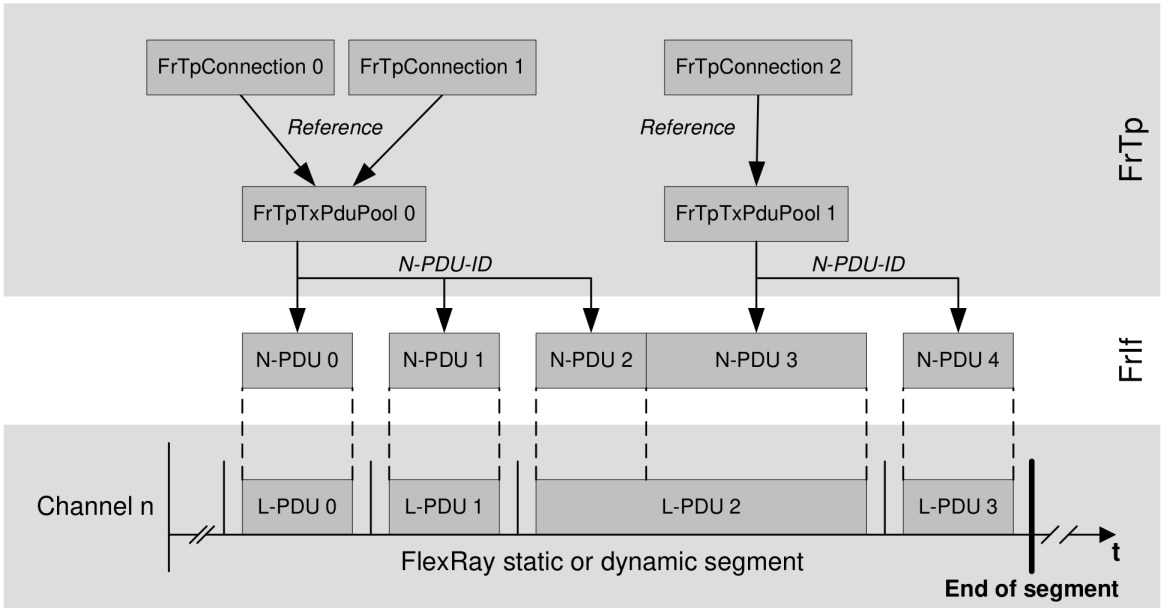


Figure 7.10: Mapping of N-PDUs to N-PDU-Pools

An FrTpTxPduPool could be referenced by different FrTpConnections (see also Chapter 7.3.2.2). Depending on the number of currently active FrTpConnections

the bandwidth (N-PDUs) is shared between them[22]. By supporting dynamic bandwidth assignment, a support of different communication scenarios is possible.

Figure 7.11 depicts two different scenarios which could be supported with only one FrTp configuration. From gateway's point of view different communication scenarios are possible:

single connection communication Complete bandwidth (slots) is assigned to one communication link (e.g. to ECU 2)

multiple connection communication Bandwidth (slots) is shared between different communication links to different ECUs (e.g. ECU 1-3).

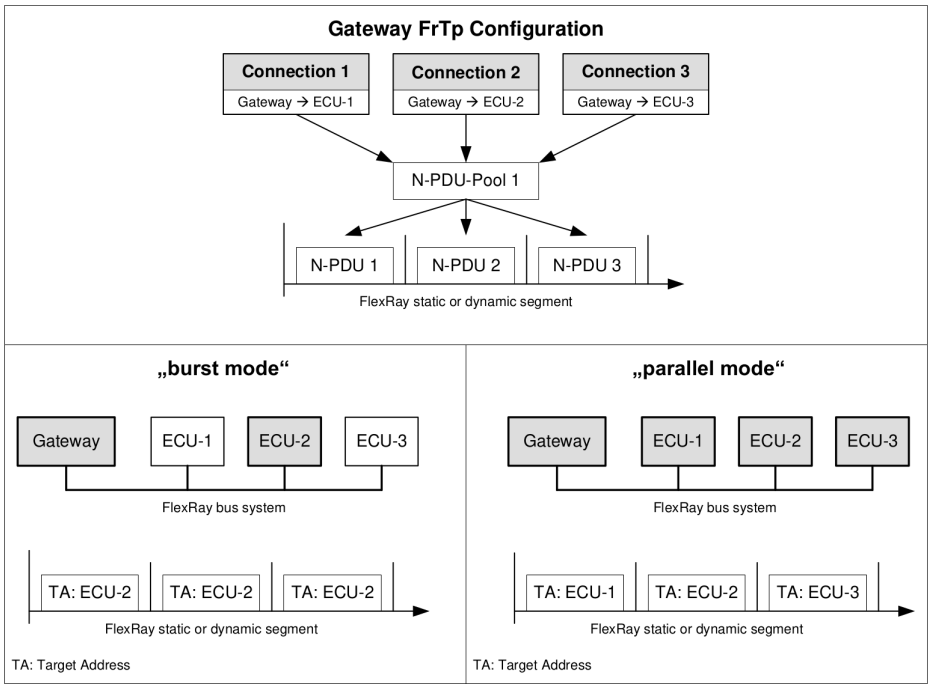


Figure 7.11: PDU-Pool sharing by different connections

The connection handler controls the partitioning of bandwidth (see Figure 7.12).

[22]Scenario: e.g. gateway communication: For diagnostic communication it is necessary to define a connection to each ECU. In some cases it is required to have a maximum communication in parrallel on the other hand it is required to have maximum bandwidth to exactly on ECU (e.g. reprogramming purpose). If dynamic bandwidth assignment is possible, both scenarios are educible with a minimum amount of FlexRay resources ("slots").

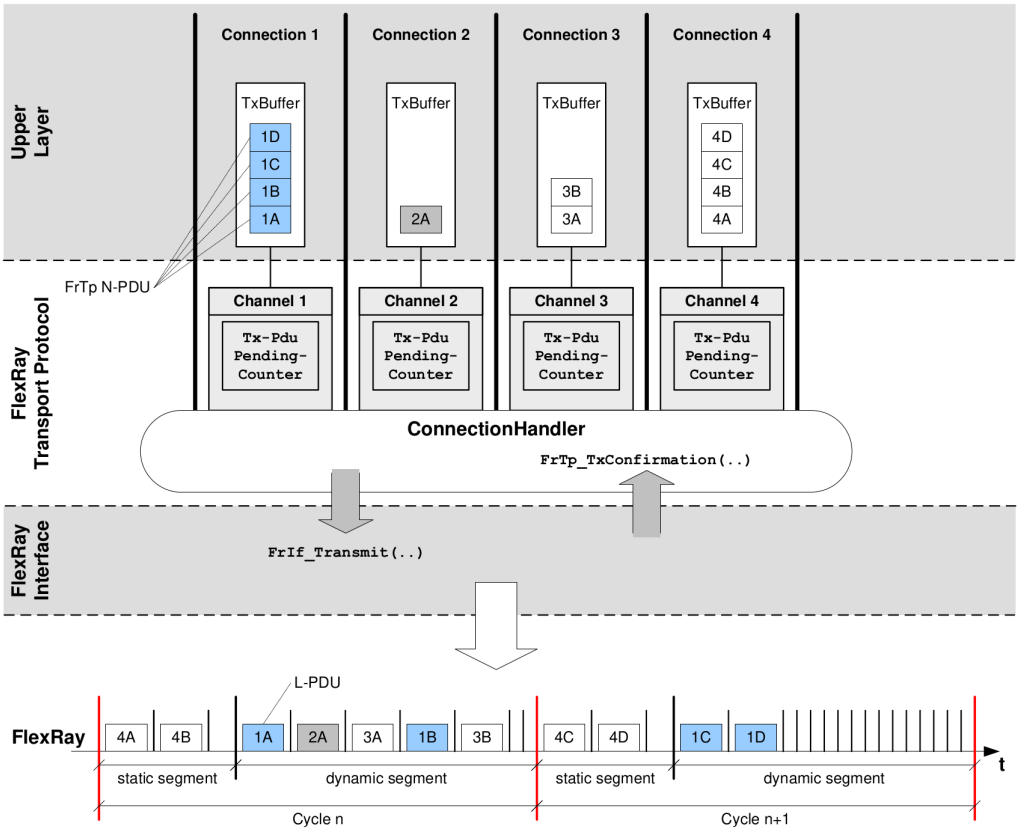


Figure 7.12: Connection Handler for different connections

The bandwidth assignment can change each communication cycle depending on the active communication links. Especially a gateway could have multiple active communication links in parallel. Hence there are some additional requirements for the FrTp module to handle concurrent connections. Especially for a segmented data transfer it is necessary to ensure that the connection handler could not swap the order of consecutive frames[23].

[SWS_FrTp_01088] Ordering of TX N-PDUs ⌈All FrTp Tx N-PDUs within an `FrTpTxPduPool` shall be listed in ascending order depending on their position within the global N-PDU network plan[24].⌋

Note: See also Chapter 7.3.2.2

[SWS_FrTp_01089] Individual Length of TX N-PDUs ⌈Each FrTp Tx N-PDU within an `FrTpTxPduPool` could have an individual length.⌋

[23] This could occure within the dynamic segment if the transfer of the last L-PDU (including consecutive frame) is skipped for the current communication cycle and within the next communication cycle other consecutive frames are sent in front of the skiped one.

[24] ECU specific N-PDU plan means that each N-PDU (uniquely identified by its N-PDU-ID) is mapped to an L-PDU. Each L-PDU is uniquely identified by its parameter set "slot-ID", "cycle counter" and "cycle offset". Hence all N-PDUs have an implicit order too.

Note: As depicted in Figure 7.10, at the end of a segemnt it could occure that only an L-PDU with less payload could be placed in the schedule. Hence the mapped FrTp N-PDU should have the corresponding length to prevent waste of bandwidth.

If more than one FrTp Tx N-PDU is used for data transmission within one connection the number of currently used FrTp Tx N-PDUs has to be controlled. Hence a counter is defined to track all initiated but currently not confirmed FrTp Tx N-PDU transmissions.

[SWS_FrTp_01090] Tx PDU pending counter support ⌈Each FrTpChannel shall implement a runtime variable TxPduPendingCounter.⌋

[SWS_FrTp_01091] Tx PDU pending counter increment handling ⌈The TxPduPendingCounter shall be incremented each time the service primitive `LSduR_FrIfTransmit` was terminated with the return value `E_OK` for the corresponding FrTp Tx N-PDU (TxPduId).⌋

[SWS_FrTp_01092] Tx PDU pending counter decrement handling ⌈The TxPduPendingCounter shall be decremented each time the service primitive `FrTp_TxConfirmation` was called with the corresponding parameter `FrTpTxConfirmationPduId`.⌋

[SWS_FrTp_01093] TxConfirmation required from lower layer ⌈A TxConfirmation shall be given for each transmitted N-PDU by the underlying layer module by calling the corresponding service primitive `FrTp_TxConfirmation` with the corresponding `FrTpTxConfirmationPduId`.⌋

The communication handler task shall process an active `FrTpConnection` (referenced by an FrTpChannel) only if the corresponding TxPduPendingCounter is zero at begin of the task. If the TxPduPendingCounter is unequal to zero an FrTp Tx N-PDU confirmation is pending and the processing for the corresponding `FrTpConnection` is skipped for the current communication handler task.

[SWS_FrTp_01094] Processing of active FrTpConnections only when no TX PDU is pending ⌈An active `FrTpConnection` (referenced by FrTpCannel) shall only processed if the TxPduPendingCouter of the corresponding FrTpChannel is zero ("0") at begin of a communication handler task.⌋

[SWS_FrTp_01095] No processing of FrTpConnection when TX PDU is pending ⌈If the TxPduPendingCouter is unequal to zero ("0") the processing for the corresponding `FrTpConnection` shall skipped for the current communication handler task.⌋

[SWS_FrTp_01096] Scheduling of active FrTpConnections ⌈A communication handler task shall process all active `FrTpConnection`s alternately[25] as long as free FrTp Tx N-PDUs are available within the referenced `FrTpTxPduPool`.⌋

[25] Alternate means that a schedule has to be implemented which processes all active `FrTpConnection`s with one N-PDU per instance. It is recommended to use a simple round-robin method but other schedules are also possible.

7.5.6 Transmit Cancellation

According to [1] the FrTp module supports "Transmit Cancellation" for an ongoing FrTp N-SDU transfer. This functionality could disable by a global compiler switch.

[SWS_FrTp_01097] Cancel transmit support depends on configuration parameter FrTpTransmitCancellation ⌈The "Transmit Cancellation" feature shall be (de)activated by static configuration of the FrTp parameter `FrTpTransmitCancellation`.⌋

[SWS_FrTp_00384] Cancel transmit request ⌈A Transmit Cancellation request shall be done by the call of the service primitive `FrTp_CancelTransmit` (see [SWS_FrTp_00150]).⌋

[SWS_FrTp_01116] Cancel Transmit handling when transmission is ongoing ⌈When a transmission is still in progress, `FrTp_CancelTransmit` shall stop the transmission and shall return `E_OK`. When a connection is not active, or when the last N-PDU of a transmission without acknowledgement has already been forwarded to the FrIf, `FrTp_CancelTransmit` shall return `E_NOT_OK`.⌋

[SWS_FrTp_00385] Cancel Transmit handling when transmission has not been started ⌈If the transmit request is pending but the transmission has not started, `FrTp_CancelTransmit` (see [SWS_FrTp_00150]) shall immediately free the connection.⌋

7.5.6.1 Transmit Cancellation for unsegmented data transfer

A Transmit Cancellation request for an unsegmented data transfer could occur on two different positions within FrTp module's processing:

Before sending the StartFrame (STF) The Transmit Cancellation Request is effective.

After sending the StartFrame (STF) The Transmit Cancellation Request is not effective because of the StartFrame was sent.

Figure 7.13 depicts the transmit cancellation behaviour of an unsegmented data transfer.

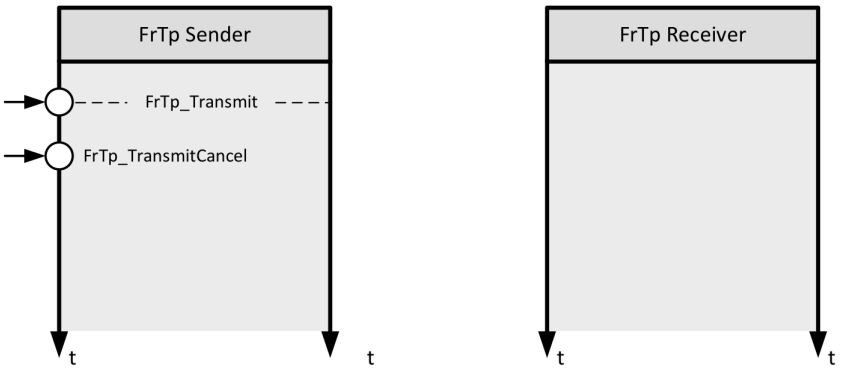


Figure 7.13: Transmit Cancellation at unsegmented data transfer

If a requested but currently not started data transfer shall be cancelled the service primitive `FrTp_CancelTransmit` is called. `FrTp_CancelTransmit` shall cancel the requested data transfer. On receiver side no data transfer is recognized.

7.5.6.2 Transmit Cancellation for segmented data transfer

A Transmit Cancellation request for a segmented data transfer could occur on three different positions within FrTp module's processing:

Before sending the StartFrame (STF) The Transmit Cancellation Request is effective.

Within an ongoing data transfer The Transmit Cancellation Request is effective.

After sending the LastFrame (LF) The Transmit Cancellation Request is not effective because of because after having transmitted the LastFrame (LF) the transmission is finished

Figure 7.14 depicts the transmit cancellation behaviour of a segmented data transfer. If a requested but currently not started data transfer shall be cancelled the service primitive `FrTp_CancelTransmit` is called. On receiver side no data transfer is recognized.

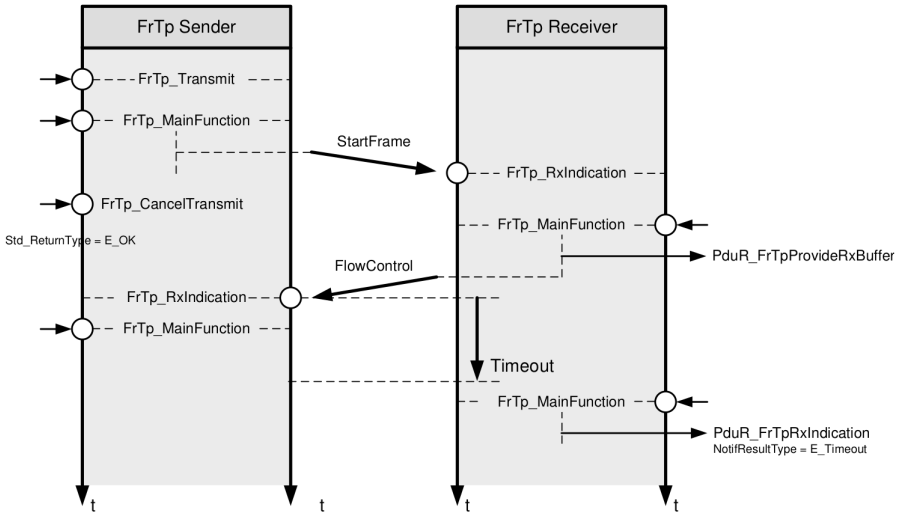


Figure 7.14: Transmit Cancellation at segmented data transfer

If an ongoing data transfer shall be cancelled, the service primitive `FrTp_CancelTransmit` is called. `FrTp_CancelTransmit` shall cancel the current data transfer process. On receiver side an initial data reception is recognized and processed (e.g. call of service primitive PduR_FrTpStartOfReception, send FlowControl N-PDU etc.). If the sender cancels data transfer a timeout occurs on receiver side.

If no retry is configured, this timeout is used to cancel the current reception by calling the service primitive PduR_FrTpRxIndication with the corresponding notification result error code.

If retry is configured, the receiver sends an additional FlowControl[26]. After a configured amount of retries the final timeout is used to cancel the current reception by calling the service primitive PduR_FrTpRxIndication with the corresponding notification result error code.

7.5.7 Change FrTp Parameter

[SWS_FrTp_00242] Bandwith control parameter BC change due to change parameter request ⌈The FrTp module shall change the ISO10681-2 FlowControl PDU parameter(s) of BandwidthControl (BC)

- `FrTpSCexp` (please refer to [1])
- `FrTpMaxNbrOfNPduPerCycle` (please refer to [1])

during runtime if the corresponding API service primitive `FrTp_ChangeParameter` is called.⌋

[SWS_FrTp_01195] BC parameter layout ⌈The layout of the BC parameter shall be identical to the layout in the FC(CTS) frame: The `FrTpMaxNbrOfNPduPerCycle` shall

[26] **Note**: On sender site the additional FlowControl is received as an unexpected N-PDU and is ignored.

be placed in bits 3..7, the FrTpSCexp in the bits 0...2. The upper byte of the parameter is not used.⌋

[SWS_FrTp_01115] Decline change parameter requests for ongoing receptions ⌈A change parameter request during an ongoing reception shall be terminated with return value of E_NOT_OK.⌋

[SWS_FrTp_01156] Usage of new parameter usage after successful change request ⌈The FrTp module shall use the new BandwidthControl parameters for the corresponding connection if the change was successfully executed.⌋

Note: Bandwidth Control is part of the runtime parameter set. For details please refer to chapter Chapter 7.3.3.3.

7.5.8 Timing parameter and timeout behaviour

The FrTp module requires different timing parameters for communication handling. This chapter defines the timing and timeout behaviour.

[SWS_FrTp_01099] FrTp timer support ⌈The FrTp module shall support the different timers and their start/stop conditions for communication handling as defined in [SWS_FrTp_01103]⌋

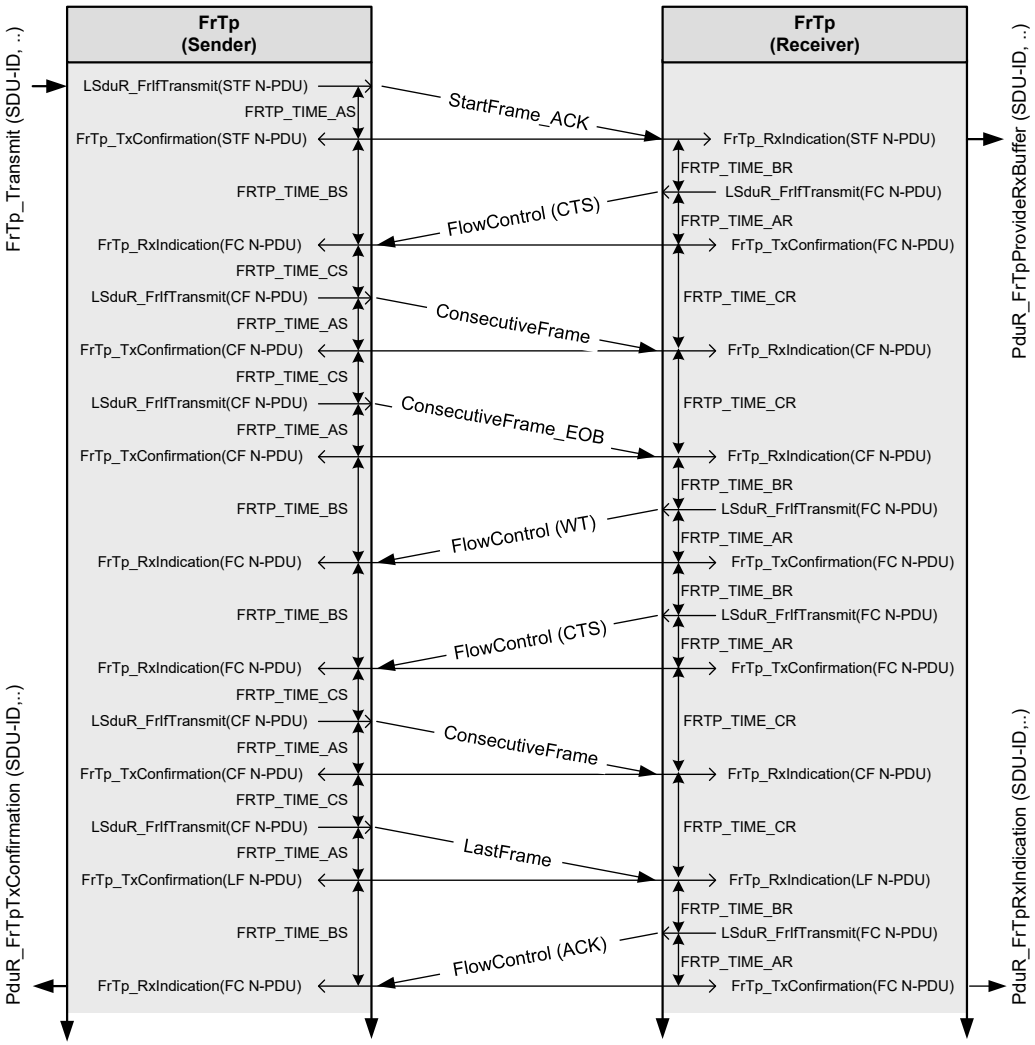


Figure 7.15: Timing parameter definition for one PDU transmission per Main-Function call

As described above it is possible to transmit more than one N-PDU per connection within on FlexRay Communication Cycle. Hence the communication handler shall be able to call `LSduR_FrTpTransmit` API several times (depending on available N-PDUs within the Tx-PDU-Pool) independent whether `FrTp_TxConfirmation` for the previously transmitted N-PDUs is given. Due to that, timing behaviour (e.g. Start FRTP_Time_AS etc.) is different too. If more than one N-PDU shall be transmitted within one Main-Function call the timing behaviour is depict in Figure 7.16.

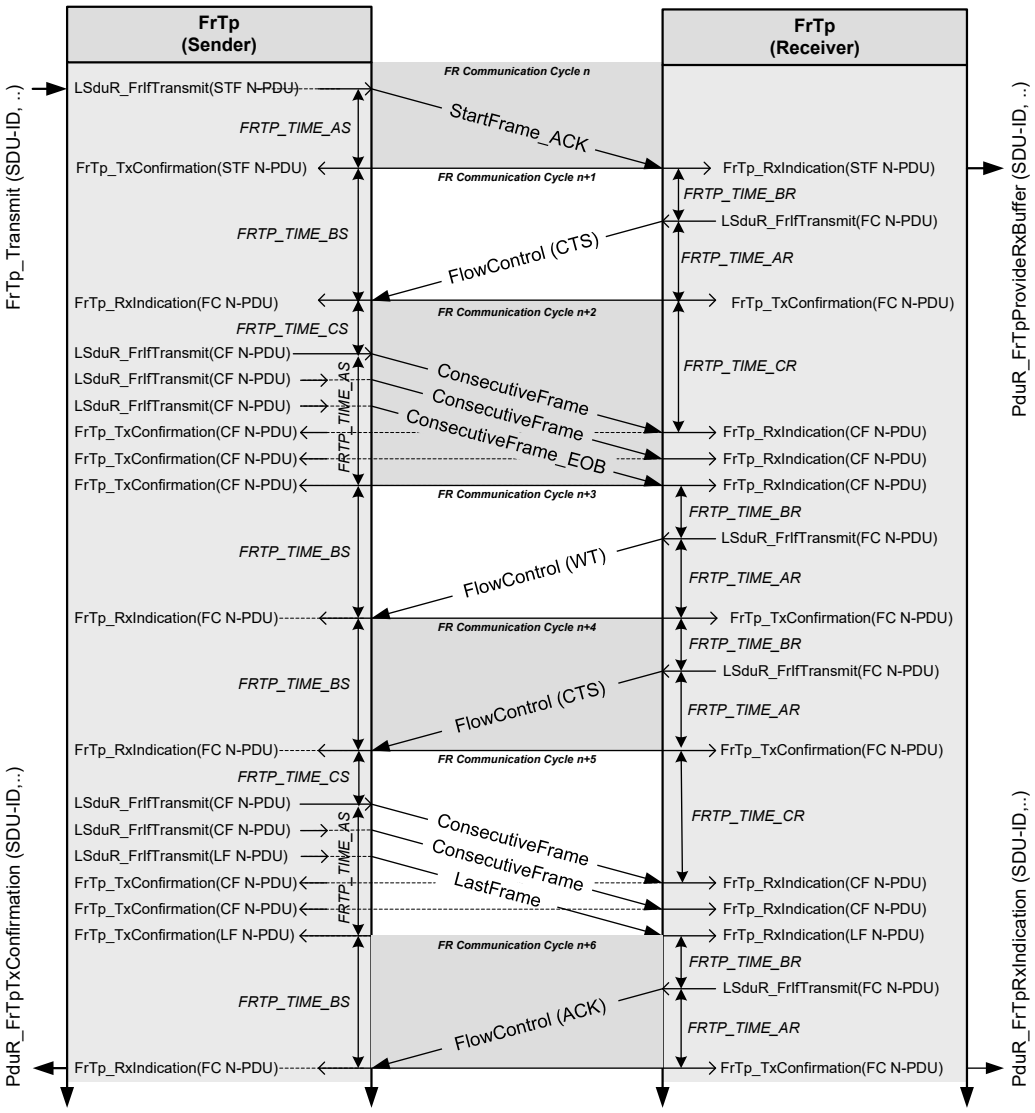


Figure 7.16: Timing parameter definition for multiple PDU transmission per Main-Function call

Note: Bandwidth Control restricts the number of N-PDUs per Flexray-Cycle. This has an impact to FrTp_Time_CS and. Hence that time depends on implementation (task schedule of `FrTp_MainFunction` and the corresponding FlowControl Parameters) For details please refer to Chapter 7.3.3.3.

[SWS_FrTp_01103] Timing parameter for FrTp module ⌈

Timing Parameter:	Description:	Timer Start Condition:	Timer Stop Condition:
FRTP_TIME_AS	Time for transmission of any FrTp N-PDU on the sender side. This is a performance requirement. The time depends on implementation and the global FlexRay schedule. The maximum time is controlled by the FRTP_TIMEOUT_AS.	LSduR_FrTp-Transmit	FrTp_Tx-Confirmation
FRTP_TIME_AR	Time for transmission of FlowControl FrTp N-PDU on the receiver side. This is a performance requirement. The time depends on implementation and the global FlexRay schedule. The maximum time is controlled by the FRTP_TIMEOUT_AR.	LSduR_FrTp-Transmit	FrTp_Tx-Confirmation
FRTP_TIME_BS	Time until reception of the next FlowControl N-PDU. The maximum time is controlled by the FRTP_TIMEOUT_BS.	FrTp_Tx-Confirmation (STF), FrTp_Rx-Indication (FC), FrTp_Tx-Confirmation (CF), FrTp_Tx-Confirmation (LF)	FrTp_Rx-Indication (FC)
FRTP_TIME_BR	Time until transmission of the next FlowControl N-PDU.	FrTp_Rx-Indication (STF), FrTp_Tx-Confirmation (FC), FrTp_Rx-Indication (CF), FrTp_Rx-Indication (LF)	LSduR_FrTp-Transmit (FC)
FRTP_TIME_CR	Time until reception of the next ConsecutiveFrame N-PDU. The maximum time is controlled by the FRTP_TIMEOUT_CR.	FrTp_Tx-Confirmation (FC), FrTp_Rx-Indication (CF)	FrTp_Rx-Indication (CF) FrTp_Rx-Indication (LF)
FRTP_TIME_CS	Time (in seconds) until transmission of the next ConsecutiveFrame N-PDU / LastFrame N-PDU.	FrTp_Tx-Confirmation (CF), FrTp_Rx-Indication (FC)	LSduR_FrTp-Transmit (CF)
S/s ... sender R/r ... receiver			

⌋

[SWS_FrTp_01100] Timeout behaviour for FrTp module ⌈

Timeout Parameter:	Cause:	Action:
FRTP_TIMEOUT_AS	Any FrTp N-PDU not transmitted in time on the sender side[27].	Abort message transmission[28]. • Call `LSduR_FrIfCancelTransmit` and free the `FrTpTxPdu`. • issue PduR_FrTpTxConfirmation with the corresponding `FrTpTxSdu` ID and `E_NOT_OK`.
FRTP_TIMEOUT_AR	Any FrTp FC N-PDU not transmitted in time on the receiver side[29].	Abort message reception and issue PduR_FrTpRxIndication with the corresponding `FrTpRxSdu` ID.
FRTP_TIMEOUT_BS	FlowControl N-PDU not received (lost, overwritten) on the sender side.	Abort message transmission and issue PduR_FrTpTxConfirmation with the corresponding `FrTpTxSdu` ID and `E_NOT_OK`.
FRTP_TIMEOUT_CR	ConsecutiveFrame or Last Frame N-PDU not received (lost, overwritten) on the receiver side[30].	Abort message reception and issue PduR_FrTpRxIndication with the corresponding `FrTpRxSdu` ID and `E_NOT_OK`.
FRTP_TIMEOUT_BR	Any FrTp FC N-PDU transmission is not initiated on the receiver side after receiving the next consecutive frame (STF or last CF of a block or LF) or after the transmit confirmation for the flow control (WT) frame.	Abort message reception and issue PduR_FrTpRxIndication with the corresponding `FrTpRxSdu` ID.
FRTP_TIMEOUT_CS	Any FrTp CF N-PDU transmission is not initiated on the sender side in time after receiving the flow control frame (CTS).	Abort message transmission and issue PduR_FrTpTxConfirmation with the corresponding `FrTpTxSdu` ID and `E_NOT_OK`.

⌋

7.6 Counters

Several counters are used to handle the different retry attempts. This chapter defines these different counters and their increasing and decreasing behaviour. Each counter is limited by a specified value. The figure below shows the different interactions between timer, counter and function calls.

[27] This could occur if an N-PDU was suspended several times within the dynamic segment

[28] **Note**: In FlexRay the transmission confirmation doesn't provide an End-To-End confirmation as on other bus protocols (e.g. CAN). This means that a transmission confirmation is provided as soon/only if the L-PDU was passed over the network. Hence if no confirmation occurs, the L-PDU is still stuck within the message buffer of the FlexRay controller, which is occupied and cannot be usesd in the meantime.

[29] This could occur if an N-PDU is suspended serveral times within the dynamic segment.

[30] This colud occur in case preceding FlowControl N-PDU not received (lost, overwritten) on overall sender side.

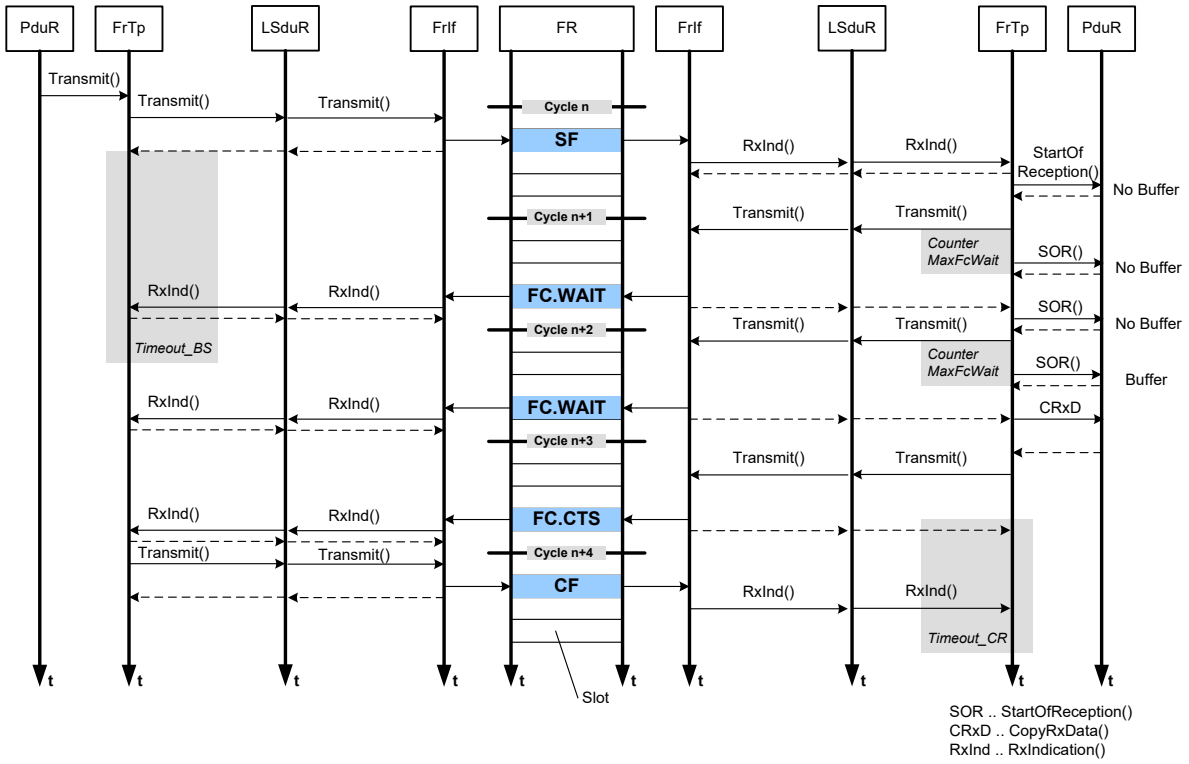


Figure 7.17: Counter, timer and function call interaction

[SWS_FrTp_01105] FrTp counter support ⌈

Counter Name:	Counter Description:	Limit:
COUNTER_FCWT	On receiver site: Counts the number of transmittes FlowControl.Wait frames[31]	FrTpMaxFCWait
Counter_RX_RN	Counts the transmission retry requests on receiver site initiated due to a frame error, e.g. bad SN in a CF.	FrTpMaxRn

The FrTp module shall support the counters and corresponding limits as defined in this table.

⌋

[SWS_FrTp_01114] FrTp counter limitation ⌈Each FrTp Counter shall be limited by a max value as defines in [SWS_FrTp_01105].⌋

[31] The limit of COUNTER_FCWT shall be in relation to the task to get an RxBuffer (service primitive call PduR_FrTpCopyRxData). The frequency of buffer request retries must be equal/higher than the FC.WAIT transmission frequency.

[SWS_FrTp_01113] Handling when FrTp counter elapsed ⌈

Counter Name:	Handling if counter has been reached:
COUNTER_FCWT	Abort transmission
COUNTER_RX_RN	**Case a) Segmented - Acknowledged transmission** • Abort reception by calling service primitive PduR_FrTpRxIndication with E_NOT_OK • Send FlowControl.ABT (if aFlowControl is possible)

If a counter of has been reached, the FrTp module shall react as defined in this table.

⌋

7.7 Error Classification

Chapter [6, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.7.1 Development Errors

[SWS_FrTp_01201] Definition of development errors in module FrTp ⌈

Type of error	*Related error code*	*Error value*
API service call without module initialization: Exception: FrTp_Init()	FRTP_E_UNINIT	0x01
NULL-Pointer on any API call	FRTP_E_PARAM_POINTER	0x02
API call with invalid SDU-ID (PduR) or PDU-ID (Fr If)	FRTP_E_INVALID_PDU_SDU_ID	0x03
API call with invalid Parameter	FRTP_E_INVALID_PARAMETER	0x04
Transmission of unknown message length is detected but not configured.	FRTP_E_UMSG_LENGTH_ERROR	0x06
FrTp initialization has been failed; e.g. selected configuration set doesn't exist.	FRTP_E_INIT_FAILED	0x08

⌋

7.7.2 Runtime Errors

[SWS_FrTp_01208] Definition of runtime errors in module FrTp ⌈

Type of error	*Related error code*	*Error value*
Segmentation is required for a 1:n connection	FRTP_E_SEG_ERROR	0x05
No free channel available	FRTP_E_NO_CHANNEL	0x07
FR TP C_As timeout is expired.	FRTP_E_FRTIMEOUTAS_TIMEOUT_OCCURRED	0x08
FR TP C_Ar timeout is expired.	FRTP_E_FRTPTIMEOUTAR_TIMEOUT_OCCURRED	0x09
FR TP C_Bs timeout is expired.	FRTP_E_FRTPTIMEOUTBS_TIMEOUT_OCCURRED	0x10
FR TP C_Br timeout is expired.	FRTP_E_FRTPTIMEBR_TIMEOUT_OCCURRED	0x11
FR TP C_Cs timeout is expired.	FRTP_E_FRTPTIMECS_TIMEOUT_OCCURRED	0x12
FR TP C_Cr timeout is expired.	FRTP_E_FRTPTIMEOUTCR_TIMEOUT_OCCURRED	0x13
FR TP receives swapped consecutive frames.	FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_RECEIVED	0x14
FR TP detects missed consecutive frames.	FRTP_E_DROPPED_CONSECUTIVE_FRAMES_DETECTED	0x15
FR TP receives FC.Ovflw frame.	FR_E_FC_OVERFLOW_RECEIVED	0x16
FR TP transmits FC.Ovflw frame.	FR_E_FC_OVERFLOW_TRANSMITTED	0x17
FR TP receives FC.Abt frame.	FR_E_FC_ABORT_RECEIVED	0x18
FR TP transmits FC.Abt frame.	FR_E_FC_ABORT_TRANSMITTED	0x19

⌋

7.7.3 Production Errors

[SWS_FrTp_00367] FR TP receives swapped consecutive frames.

Upstream requirements: SRS_BSW_00458

⌈

Diagnostic Event (Error Name)	FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_RECEIVED
Description	If FR TP receives swapped consecutive frames, FR TP shall report a FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_RECEIVED Extended Production Error to DEM.
Failed condition	FR TP receives swapped consecutive frames.
Passed condition	FrTp_Init function call.

⌋

7.8 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

This chapter lists all included types for FlexRay Transport Layer and their corresponding header files.

[SWS_FrTp_91000] Definition of imported datatypes of module FrTp ⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	BufReq_ReturnType
	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
	ComStack_Types.h	RetryInfoType
	ComStack_Types.h	TPParameterType
	ComStack_Types.h	TpDataStateType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

8.2.1 FrTp_ConfigType

The post-build-time configuration fulfills two functionalities:

- Post-build selctable, where more than one configuration is located in the ECU, and one is selected at init of the FrTp module
- Post-build loadable, where one configuration is located in the ECU. This configuration may be reprogrammed after compile-time

Basically there is no restriction to mix both selectable and loadable. Typically the post-build loadable is located in its own flash sector where it can be reprogrammed without affecting other modules/applications.

[SWS_FrTp_01194] Definition of datatype FrTp_ConfigType ⌈

Name	FrTp_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–

▽

△

Description	This is the base type for the configuration of the FlexRay Transport Protocol A pointer to an instance of this structure will be used in the initialization of the FlexRay Transport Protocol. The outline of the structure is defined in chapter 10 Configuration Specification
Available via	FrTp.h

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

8.3.1 Initialization and Shutdown

8.3.1.1 FrTp_Init

[SWS_FrTp_00147] Definition of API function FrTp_Init

Upstream requirements: SRS_BSW_00101

⌈

Service Name	FrTp_Init	

Syntax	`void FrTp_Init (`
	`  const FrTp_ConfigType* configPtr`
	`)`	

Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	configPtr	Pointer to FlexRay Transport Protocol configuration.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes all global variables of a FlexRay Transport Layer instance and set it in the idle state. It has no return value because software errors in initialisation data shall be detected during configuration time (e.g. by configuration tool).	
Available via	FrTp.h	

⌋

8.3.1.2 FrTp_Shutdown

[SWS_FrTp_00148] Definition of API function FrTp_Shutdown ⌈

Service Name	FrTp_Shutdown
Syntax	`void FrTp_Shutdown ( void )`
Service ID [hex]	0x01
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	This service closes all pending transport protocol connections by simply stopping operation, frees all resources and stops the FrTp Module
Available via	FrTp.h

⌋

8.3.2 Normal Operation

8.3.2.1 FrTp_Transmit

[SWS_FrTp_00149] Definition of API function FrTp_Transmit ⌈

Service Name	FrTp_Transmit	
Syntax	`Std_ReturnType FrTp_Transmit ( PduIdType TxPduId, const PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x49	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identifier of the PDU to be transmitted
	PduInfoPtr	Length of and pointer to the PDU data and pointer to MetaData.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Transmit request has been accepted. `E_NOT_OK`: Transmit request has not been accepted.
Description	Requests transmission of a PDU.	
Available via	FrTp.h	

⌋

Note: The service primitive `FrTp_Transmit` sets the flag TX_SDU_AVAILABLE if new data are available for transmission.

[SWS_FrTp_01139] Raise development error when transmit is called with invalid PDU ID ⌈If development error detection for the `FrTp_Transmit` is enabled:

the function FrTp_Transmit shall check the parameter TxPduId for being valid. If the check for TxPduId fails, the function FrTp_Transmit shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.⌋

[SWS_FrTp_01140] Raise development error when transmit is called with invalid PDU info pointer ⌈If development error detection for the FrTp_Transmit is enabled: the function FrTp_Transmit shall check the parameter PduInfoPtr for being valid. If the check for PduInfoPtr fails, the function FrTp_Transmit shall raise the development error FRTP_E_PARAM_POINTER and return E_NOT_OK.⌋

8.3.2.2 FrTp_TriggerTransmit

[SWS_FrTp_00154] Definition of callback function FrTp_TriggerTransmit ⌈

Service Name	FrTp_TriggerTransmit	
Syntax	`Std_ReturnType FrTp_TriggerTransmit ( PduIdType TxPduId, PduInfoType* PduInfoPtr )`	
Service ID [hex]	0x41	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the SDU that is requested to be transmitted.
Parameters (inout)	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied, and the available buffer size in SduLengh. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU data has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description	Within this API, the upper layer module (called 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 in PduInfoPtr->SduLength. If not, it returns E_NOT_OK without changing PduInfoPtr.	
Available via	FrTp.h	

⌋

[SWS_FrTp_01145] Raise development error when trigger transmit is called with invalid PDU ID ⌈If development error detection for the FrTp_TriggerTransmit is enabled: the function FrTp_TriggerTransmit shall check the parameter TxPduId for being valid. If the check for TxPduId fails, the function FrTp_TriggerTransmit shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.⌋

[SWS_FrTp_01146] Raise development error when trigger transmit is called with invalid PDU info pointer ⌈If development error detection for the FrTp_TriggerTransmit is enabled: the function FrTp_TriggerTransmit shall check the parameter PduInfoPtr for being valid. If the check for PduInfoPtr fails, the function FrTp_TriggerTransmit shall raise the development error FRTP_E_PARAM_POINTER and return E_NOT_OK.⌋

8.3.2.3 FrTp_CancelTransmit

[SWS_FrTp_00150] Definition of API function FrTp_CancelTransmit ⌈

Service Name	FrTp_CancelTransmit	
Syntax	`Std_ReturnType FrTp_CancelTransmit ( PduIdType TxPduId )`	
Service ID [hex]	0x4a	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	
Available via	FrTp.h	

⌋

[SWS_FrTp_01141] Raise development error when cancel transmit is called with invalid PDU ID ⌈If development error detection for the FrTp_CancelTransmit is enabled: the function FrTp_CancelTransmit shall check the parameter TxPduId for being valid. If the check for TxPduId fails, the function FrTp_CancelTransmit shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.⌋

8.3.2.4 FrTp_TxConfirmation

[SWS_FrTp_00153] Definition of callback function FrTp_TxConfirmation ⌈

Service Name	FrTp_TxConfirmation
Syntax	`void FrTp_TxConfirmation ( PduIdType TxPduId, Std_ReturnType result )`
Service ID [hex]	0x40
Sync/Async	Synchronous

▽

△

Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted successfully. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	
Available via	FrTp.h	

⌋

[SWS_FrTp_01149] Raise development error when TX confirmation is called with invalid PDU ID ⌈If development error detection for the FrTp_TxConfirmation is enabled: the function FrTp_TxConfirmation shall check the parameter TxPduId for being valid. If the check for TxPduId fails, the function FrTp_TxConfirmation shall raise the development error FRTP_E_INVALID_PDU_SDU_ID.⌋

[SWS_FrTp_01203] Raise development error when TX confirmation is called with invalid parameter result ⌈If development error detection is enabled: The function FrTp_TxConfirmation shall check the parameter result for being valid. If the check for result fails, the function FrTp_TxConfirmation shall raise the development error FRTP_E_INVALID_PARAMETER.⌋

8.3.2.5 FrTp_RxIndication

[SWS_FrTp_00152] Definition of callback function FrTp_RxIndication ⌈

Service Name	FrTp_RxIndication	

Syntax	`void FrTp_RxIndication (`
	`  PduIdType RxPduId,`
	`  const PduInfoType* PduInfoPtr`
	`)`	

Service ID [hex]	0x42	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in)	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Indication of a received PDU from a lower layer communication interface module.	
Available via	FrTp.h	

⌋

[SWS_FrTp_01147] Raise development error when RX indication is called with invalid PDU ID ⌈If development error detection for the `FrTp_RxIndication` is enabled: the function `FrTp_RxIndication` shall check the parameter RxPduId for being valid. If the check for RxPduId fails, the function `FrTp_RxIndication` shall raise the development error `FRTP_E_INVALID_PDU_SDU_ID`.⌋

[SWS_FrTp_01148] Raise development error when RX indication is called with invalid PDU info pointer ⌈If development error detection for the `FrTp_RxIndication` is enabled: the function `FrTp_RxIndication` shall check the parameter PduInfoPtr for being valid. If the check for PduInfoPtr fails, the function `FrTp_RxIndication` shall raise the development error `FRTP_E_PARAM_POINTER`.⌋

8.3.2.6 FrTp_CancelReceive

[SWS_FrTp_01172] Definition of API function FrTp_CancelReceive ⌈

Service Name	FrTp_CancelReceive	
Syntax	`Std_ReturnType FrTp_CancelReceive ( PduIdType RxPduId )`	
Service ID [hex]	0x4c	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	RxPduId	Identification of the PDU to be cancelled.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Cancellation was executed successfully by the destination module. `E_NOT_OK`: Cancellation was rejected by the destination module.
Description	Requests cancellation of an ongoing reception of a PDU in a lower layer transport protocol module.	
Available via	FrTp.h	

⌋

[SWS_FrTp_01180] Raise development error when cancel receive is called with invalid RX PDU ID ⌈If development error detection is enabled:

The function `FrTp_CancelReceive` shall check the parameter RxPduId for being valid. If the check for RxPduId fails, the function shall raise the development error `FRTP_E_INVALID_PDU_SDU_ID` and return `E_NOT_OK`.⌋

[SWS_FrTp_01181] Abort reception when cancel receive provides a valid RX PDU ID ⌈The FrTp shall abort the reception of the current N-PDU if the service `FrTp_CancelReceive` provides a valid RxPduId.⌋

[SWS_FrTp_01182] Reject cancel receive for unsgemented reception or when processing last frame ⌈The FrTp shall reject the request for receive cancellation in case of an

- unsegmented reception or
- in case the FrTp is in the process of receiving the LastFrame of the N-SDU

and shall return E_NOT_OK.⌋

[SWS_FrTp_01183] Return handling for cancel receive after success ⌈If the FrTp_CancelReceive service has been successfully Executed, FrTp_CancelReceive shall return with result E_OK.⌋

8.3.2.7 FrTp_ChangeParameter

[SWS_FrTp_00151] Definition of API function FrTp_ChangeParameter ⌈

Service Name	FrTp_ChangeParameter	
Syntax	`Std_ReturnType FrTp_ChangeParameter ( PduIdType id, TPParameterType parameter, uint16 value )`	
Service ID [hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	id	Identification of the PDU which the parameter change shall affect.
	parameter	ID of the parameter that shall be changed.
	value	The new value of the parameter.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The parameter was changed successfully. E_NOT_OK: The parameter change was rejected.
Description	Request to change a specific transport protocol parameter (e.g. block size).	
Available via	FrTp.h	

⌋

Note: The service FrTp_ChangeParameter is used to change the transport protocol parameter Bandwidth Control (BC).

[SWS_FrTp_01143] Raise development error when change parameter is called with invalid parameter ID ⌈If development error detection for the FrTp_ChangeParameter is enabled: the function FrTp_ChangeParameter shall check the parameter Id for being valid. If the check for Id fails, the function FrTp_ChangeParameter shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.⌋

[SWS_FrTp_01144] Raise development error when change parameter is called with invalid parameter ⌈If development error detection for the `FrTp_ChangeParameter` is enabled: the function `FrTp_ChangeParameter` shall check the parameter for being valid. If the check for parameter fails, the function `FrTp_ChangeParameter` shall raise the development error `FRTP_E_INVALID_PARAMETER` and return `E_NOT_OK`.⌋

8.4 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.4.1 FrTp_MainFunction

[SWS_FrTp_00203] Main Function schedule ⌈The `FrTp_MainFunction` shall be used to schedule the FrTp module.⌋

[SWS_FrTp_00580] Main function processing ⌈The `FrTp_MainFunction` shall be the entry point for FrTp processing tasks.⌋

[SWS_FrTp_01152] Main function call at least required once per FR cycle ⌈The `FrTp_MainFunction` shall be called at least one time per FlexRay cycle[1].⌋

The `FrTp_MainFunction` shall follow the service primitive definition as described below:

[SWS_FrTp_00162] Definition of scheduled function FrTp_MainFunction ⌈

Service Name	FrTp_MainFunction
Syntax	`void FrTp_MainFunction ( void )`
Service ID [hex]	0x10
Description	Schedules the FlexRay TP. (Entry point for scheduling)
Available via	SchM_FrTp.h

⌋

[1] The number of MainFunction calls depends on the global FlexRay communication cycle length, the available receive buffers of the FlexRay driver and the implementation (which functionality of transmission and reception could be implemented in interrupt mode). At least one call is necessary to reconfigure the buffers for the corresponding cycle.
If more than one call is necessary it is recommended to call `FrTp_MainFunction` at the start of the static segment and at the start of the dynamic segment within the communication cycle. If the length of that segments are asymetric the different segment lengths have to be considered.

8.5 Expected interfaces

In this chapter all interfaces required from other modules are listed.

8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[SWS_FrTp_00577] Definition of mandatory interfaces required by module FrTp
⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
LSduR_FrTpTransmit (draft)	LSduR_<module>.h	Requests transmission of a PDU.
PduR_FrTpCopyRxData	PduR_FrTp.h	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.
PduR_FrTpCopyTxData	PduR_FrTp.h	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
PduR_FrTpRxIndication	PduR_FrTp.h	Called after an I-PDU has been received via the TP API, the result indicates whether the reception was successful or not.
PduR_FrTpStartOfReception	PduR_FrTp.h	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSdu Length equal to 0.
PduR_FrTpTxConfirmation	PduR_FrTp.h	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

[SWS_FrTp_00579] Definition of optional interfaces requested by module FrTp ⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
LSduR_FrTpCancelTransmit (draft)	–	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.

⌋

8.5.3 Configurable interfaces

No interfaces are defined.

8.6 Service Interfaces

No interfaces are defined.

9 Sequence diagrams

Although the following sequence diagrams are quite detailed, they do not depict every detail. Thus they should be seen as an addendum to this specification.

9.1 Sending of N-Pdus

The flow chart below depicts the sending process' API calls and flag handling. The flow chart shows segmented/unsegmented transfer, Transfer with and without acknowledgement and immediate / decoubled buffer access.

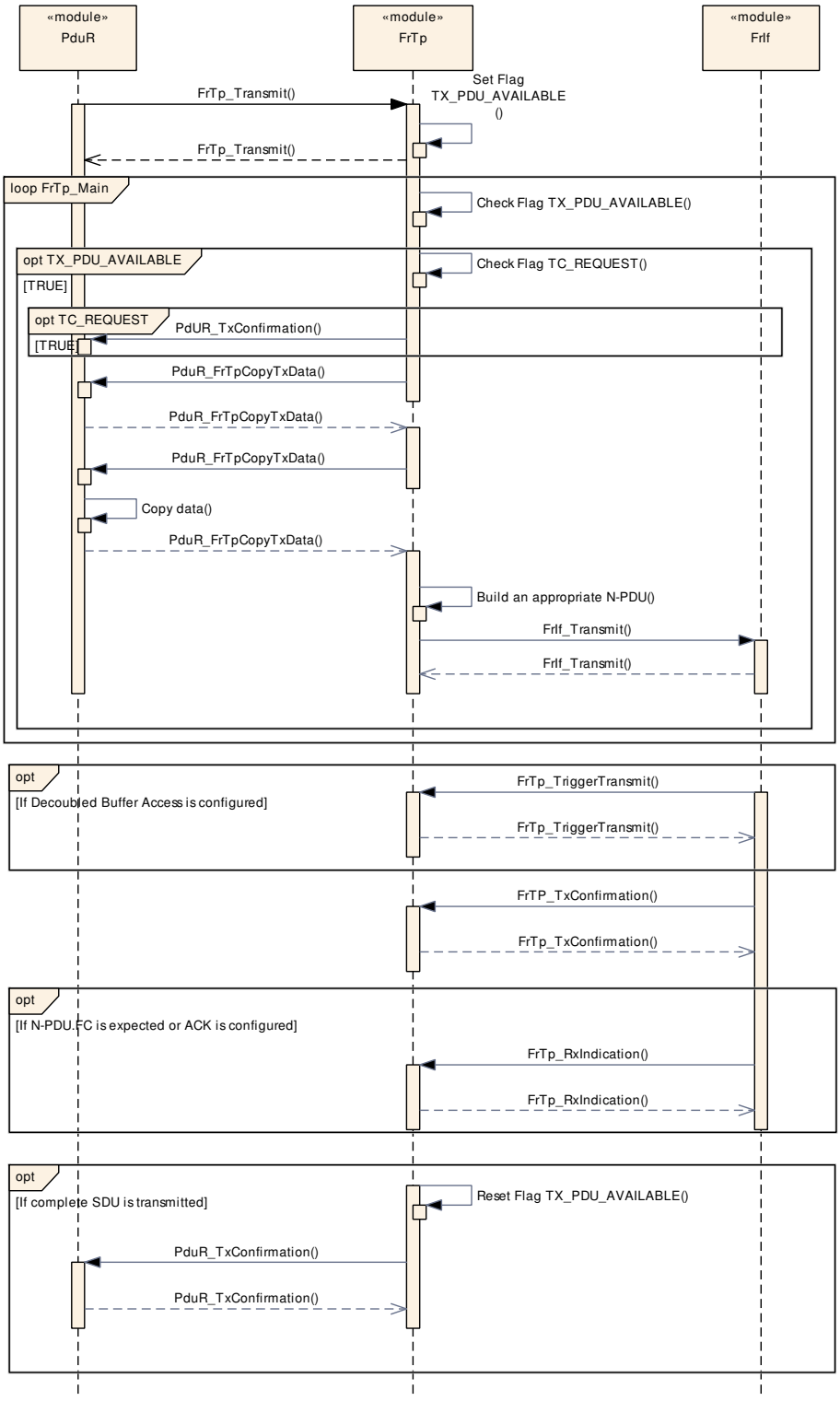


Figure 9.1: Sending of N-Pdus

9.2 Receiving of N-Pdus

The flow chart below depicts the receiving process' API calls and flag handling. The flow chart shows segmented/unsegmented transfer, Transfer with and without acknowledgement and immediate / decoubled buffer access.

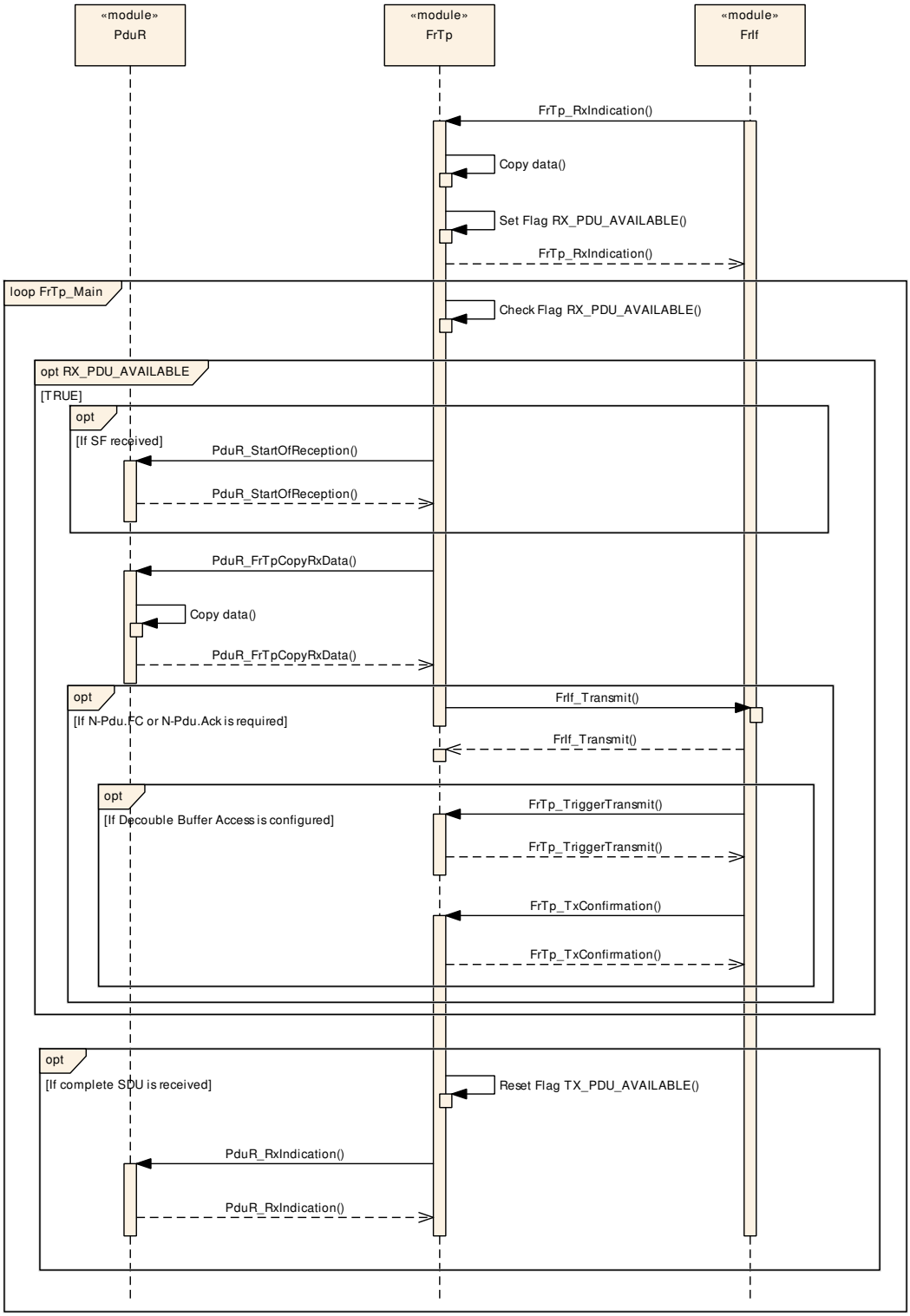


Figure 9.2: Receiving of N-Pdus (`FrTp_MainFunction` shall call PduR_FrTpStartOfReception routine)

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module FlexRay Transport Layer module.

Chapter 10.3 specifies published information of the module FlexRay Transport Layer module.

[SWS_FrTp_00569] Configuration Tool ⌈The configuration tool should extract all information to configure the FlexRay Transport Protocol.⌋

10.1 How to read this chapter

For details refer to [6] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

10.2.1 FrTp

[ECUC_FrTp_00001] Definition of EcucModuleDef FrTp ⌈

Module Name	FrTp
Description	Configuration of the FlexRay Transport Protocol module according to ISO 10681-2.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	**Multiplicity**	**Description**
FrTpDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
FrTpGeneral	1	This container contains the general configuration parameters of the FlexRay Transport Protocol module.
FrTpMultipleConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR FrTp module.

⌋

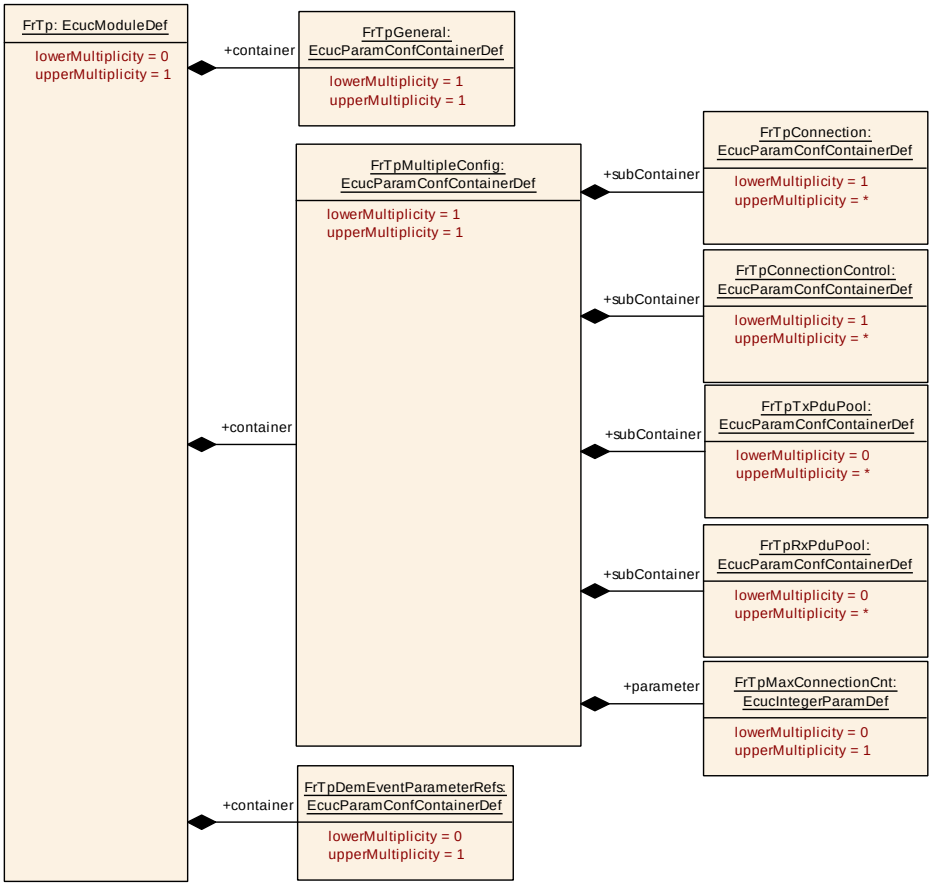


Figure 10.1: FrTp Module Configuration

10.2.2 FrTpGeneral

[ECUC_FrTp_00009] Definition of EcucParamConfContainerDef FrTpGeneral ⌈

Container Name	FrTpGeneral
Parent Container	FrTp
Upstream mapping defined	–
Description	This container contains the general configuration parameters of the FlexRay Transport Protocol module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpAckRt	1	[ECUC_FrTp_00002]
FrTpChangeParamApi	1	[ECUC_FrTp_00052]
FrTpChanNum	1	[ECUC_FrTp_00004]
FrTpDevErrorDetect	1	[ECUC_FrTp_00008]
FrTpFullDuplexEnable	1	[ECUC_FrTp_00051]
FrTpMainFuncCycle	1	[ECUC_FrTp_00011]
FrTpTransmitCancellation	1	[ECUC_FrTp_00036]
FrTpUnknownMsgLength	1	[ECUC_FrTp_00044]

No Included Containers

⌋

[ECUC_FrTp_00002] Definition of EcucBooleanParamDef FrTpAckRt ⌈

Parameter Name	FrTpAckRt
Parent Container	FrTpGeneral
Upstream mapping defined	–
Description	Preprocessor switch for enabling the Acknowledgement and retry mechanisms. True: Acknowledge and Retry is enabled False: Acknowledge and Retry is disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_FrTp_00052] Definition of EcucBooleanParamDef FrTpChangeParamApi ⌈

Parameter Name	FrTpChangeParamApi
Parent Container	FrTpGeneral
Upstream mapping defined	–
Description	Preprocessor switch for enabling the API to change FrTp communication parameters. True: ChangeParameter API is enabled False: ChangeParameter API is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_FrTp_00004] Definition of EcucIntegerParamDef FrTpChanNum ⌈

Parameter Name	FrTpChanNum
Parent Container	FrTpGeneral
Upstream mapping defined	–

▽

△

Description	Preprocessor switch for defining the number of concurrent channels the module supports. Up to 32 channels shall be definable here.
Multiplicity	1
Type	EcucIntegerParamDef
Range	1 .. 32
Default value	–
Pre-Compile only	true

⌋

[ECUC_FrTp_00008] Definition of EcucBooleanParamDef FrTpDevErrorDetect ⌈

Parameter Name	FrTpDevErrorDetect
Parent Container	FrTpGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FrTp_00051] Definition of EcucBooleanParamDef FrTpFullDuplexEnable ⌈

Parameter Name	FrTpFullDuplexEnable
Parent Container	FrTpGeneral
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Preprocessor switch for enabling full duplex mechanisms for all channels. True: Full duplex is enabled False: Fullduplex is disabled (Half duplex is enabled)
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FrTp_00011] Definition of EcucFloatParamDef FrTpMainFuncCycle ⌈

Parameter Name	FrTpMainFuncCycle
Parent Container	FrTpGeneral
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This parameter contains the calling period of the TPs Main Function. The parameter is specified in seconds.
Multiplicity	1
Type	EcucFloatParamDef

▽

△

Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FrTp_00036] Definition of EcucBooleanParamDef FrTpTransmitCancellation ⌈

Parameter Name	FrTpTransmitCancellation
Parent Container	FrTpGeneral
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	Preprocessor switch for enabling Transmit Cancellation and Receive Cancellation. True: Transmit/Receive Cancellation is enabled False: Transmit/Receive Cancellation is disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FrTp_00044] Definition of EcucBooleanParamDef FrTpUnknownMsgLength ⌈

Parameter Name	FrTpUnknownMsgLength
Parent Container	FrTpGeneral
Upstream mapping defined	–
Description	Preprocessor switch to support data transfer with unknown message length. True: Transmission with unknown message length is enabled False: Transmission with unknown message length is disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

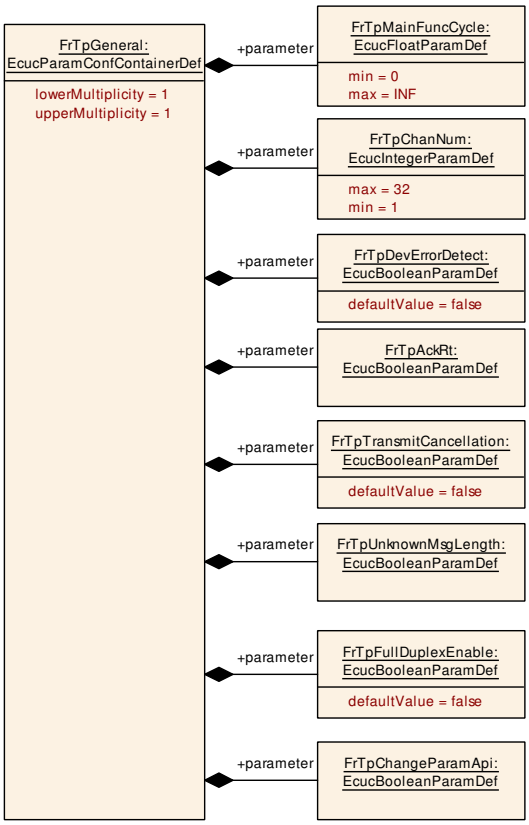


Figure 10.2: FrTpGeneral

10.2.3 FrTpMultipleConfig

[ECUC_FrTp_00018] Definition of EcucParamConfContainerDef FrTpMultipleConfig ⌈

Container Name	FrTpMultipleConfig
Parent Container	FrTp
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR FrTp module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpMaxConnectionCnt	0..1	[ECUC_FrTp_00054]

Included Containers		
Container Name	**Multiplicity**	**Description**
FrTpConnection	1..*	This container contains the connection specific parameters to transfer N-PDUs via FlexRay TP.
FrTpConnectionControl	1..*	This container contains the configuration parameters to control a FlexRay TP connection.
FrTpRxPduPool	0..*	This container contains all Pdus that are assigned to that Pdu Pool.
FrTpTxPduPool	0..*	This container contains all Pdus that are assigned to that Pdu Pool.

⌋

[ECUC_FrTp_00054] Definition of EcucIntegerParamDef FrTpMaxConnectionCnt

⌈

Parameter Name	FrTpMaxConnectionCnt	
Parent Container	FrTpMultipleConfig	
Upstream mapping defined	–	
Description	Maximum number of TP connections. This parameter is needed only in case of post-build loadable implementation using static memory allocation.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 18446744073709551615	
Default value	–	
Pre-Compile only	true	

⌋

10.2.4 FrTpConnection

[ECUC_FrTp_00006] Definition of EcucParamConfContainerDef FrTpConnection

⌈

Container Name	FrTpConnection	
Parent Container	FrTpMultipleConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This container contains the connection specific parameters to transfer N-PDUs via Flex Ray TP.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpBandwidthLimitation	1	[ECUC_FrTp_00050]
FrTpLa	0..1	[ECUC_FrTp_00010]
FrTpMultipleReceiverCon	1	[ECUC_FrTp_00019]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpRa	0..1	[ECUC_FrTp_00021]
FrTpConCtrlRef	1	[ECUC_FrTp_00005]
FrTpRxPduPoolRef	0..1	[ECUC_FrTp_00025]
FrTpTxPduPoolRef	0..1	[ECUC_FrTp_00039]

Included Containers		
Container Name	**Multiplicity**	**Description**
FrTpRxSdu	0..1	This parameter defines the Rx Service Data Unit Identifier (Sdu Id) which uniquely identifies a data transfer (inter-module communication) between FrTp and PDUR. This N-SDU can produce meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.
FrTpTxSdu	0..1	This parameter defines the Tx Service Data Unit Identifier (Sdu Id) which uniquely identifies a data transfer (inter-module communication) between FrTp and PDUR. This N-SDU can consume meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.

⌋

[ECUC_FrTp_00050] Definition of EcucBooleanParamDef FrTpBandwidthLimitation ⌈

Parameter Name	FrTpBandwidthLimitation	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter indicates whether the connection requires a bandwidth limitation or not. If FrTpBandwidthLimitation=True the sender shall send a StartFrame always on the first PDU of a PDU-Pool.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00010] Definition of EcucIntegerParamDef FrTpLa ⌈

Parameter Name	FrTpLa	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the Local Address for the respective connection. When the local instance is the sender, this is the Source Address within the TP frame. When the local instance is the receiver, this is the Target Address within the TP frame. If this parameter is not configured, all related Rx N-SDUs must be configured to use the meta data item TARGET_ADDRESS_16, and all related Tx-N-SDUs must be configured to use the meta data item SOURCE_ADDRESS_16.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	

▽

△

Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00019] Definition of EcucBooleanParamDef FrTpMultipleReceiverCon ⌈

Parameter Name	FrTpMultipleReceiverCon	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines, whether this connection is an 1:1 ('false') or an 1:n ('true') connection. If data segmentation is required this parameter is used to check whether segmentation is possible or not. If the connection is 1:n segmentation is not possible and an error will occur.	
Multiplicity	1	
Type	EcucBooleanParamDef	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00021] Definition of EcucIntegerParamDef FrTpRa ⌈

Parameter Name	FrTpRa	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the Remote Address for the respective connection. When the local instance is the sender, this is the Target Address within the TP frame. When the local instance is the receiver, this is the Source Address within the TP frame. If this parameter is not configured, all related Rx N-SDUs must be configured to use the meta data item SOURCE_ADDRESS_16, and all related Tx-N-SDUs must be configured to use the meta data item TARGET_ADDRESS_16.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00005] Definition of EcucReferenceDef FrTpConCtrlRef ⌈

Parameter Name	FrTpConCtrlRef	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	FrTpConnectionControlReference: This parameter defines a reference to a connection control container.	
Multiplicity	1	
Type	Reference to FrTpConnectionControl	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00025] Definition of EcucReferenceDef FrTpRxPduPoolRef ⌈

Parameter Name	FrTpRxPduPoolRef	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines a reference to a RxPduPool.	
Multiplicity	0..1	
Type	Reference to FrTpRxPduPool	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00039] Definition of EcucReferenceDef FrTpTxPduPoolRef ⌈

Parameter Name	FrTpTxPduPoolRef	
Parent Container	FrTpConnection	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines a reference to a TxPduPool.	
Multiplicity	0..1	
Type	Reference to FrTpTxPduPool	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

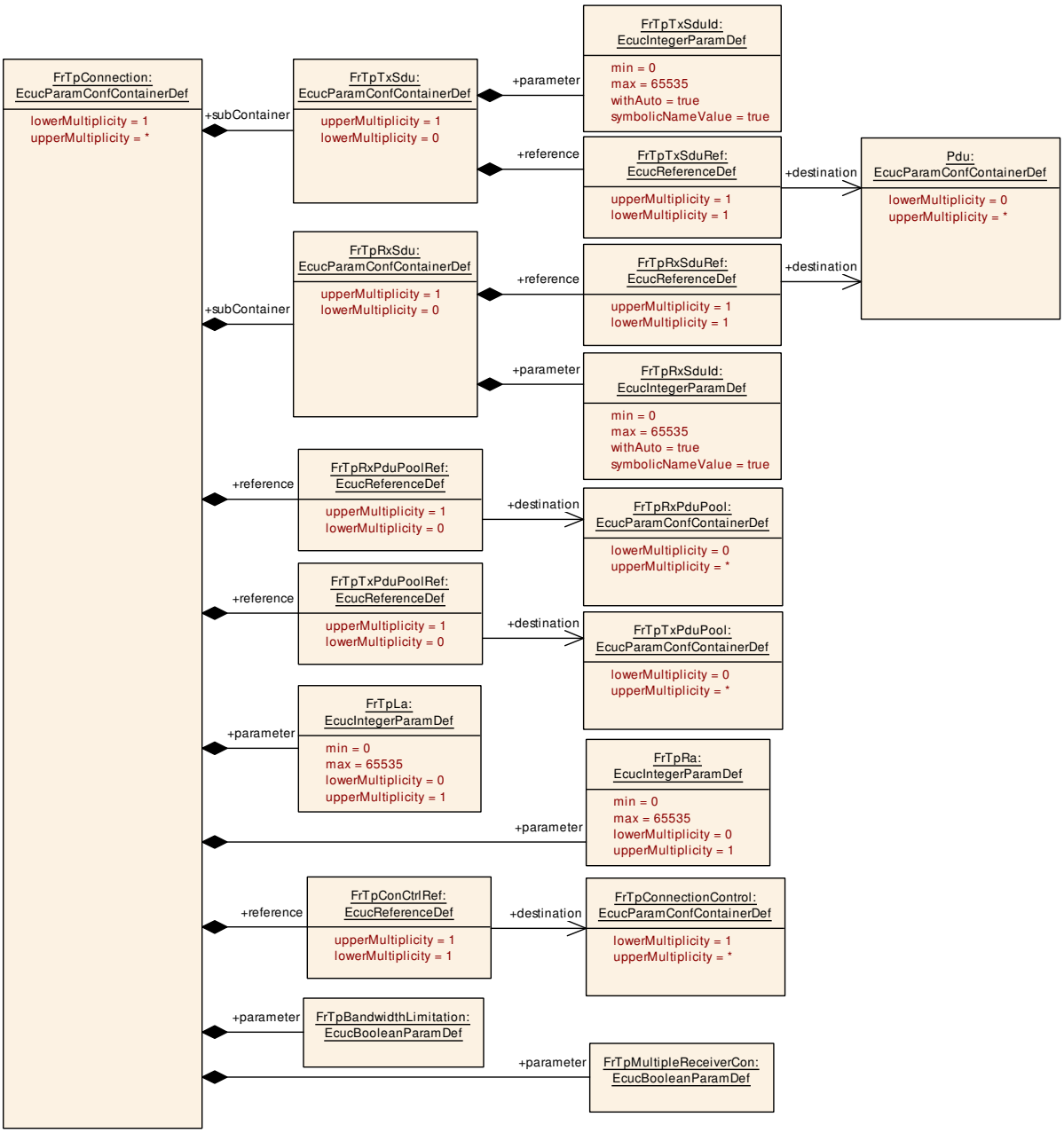


Figure 10.3: FrTpConnection

10.2.5 FrTpTxSdu

[ECUC_FrTp_00041] Definition of EcucParamConfContainerDef FrTpTxSdu ⌈

Container Name	FrTpTxSdu
Parent Container	FrTpConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	This parameter defines the Tx Service Data Unit Identifier (Sdu Id) which uniquely identifies a data transfer (inter-module communication) between FrTp and PDUR. This N-SDU can consume meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpTxSduId	1	[ECUC_FrTp_00042]
FrTpTxSduRef	1	[ECUC_FrTp_00043]

No Included Containers

⌋

[ECUC_FrTp_00042] Definition of EcucIntegerParamDef FrTpTxSduId ⌈

Parameter Name	FrTpTxSduId	
Parent Container	FrTpTxSdu	
Upstream mapping defined	–	
Description	This is a unique identifier for a to be transmitted message from the PduR to the FrTp. ImplementationType: PduIdType	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_FrTp_00043] Definition of EcucReferenceDef FrTpTxSduRef ⌈

Parameter Name	FrTpTxSduRef	
Parent Container	FrTpTxSdu	
Upstream mapping defined	–	
Description	Reference to a PDU in the global PDU structure.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.6 FrTpRxSdu

[ECUC_FrTp_00027] Definition of EcucParamConfContainerDef FrTpRxSdu ⌈

Container Name	FrTpRxSdu
Parent Container	FrTpConnection
Upstream mapping defined	[9, CP_TPS_SystemTemplate]
Description	This parameter defines the Rx Service Data Unit Identifier (Sdu Id) which uniquely identifies a data transfer (inter-module communication) between FrTp and PDUR. This N-SDU can produce meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpRxSduId	1	[ECUC_FrTp_00053]
FrTpRxSduRef	1	[ECUC_FrTp_00028]

No Included Containers

⌋

[ECUC_FrTp_00053] Definition of EcucIntegerParamDef FrTpRxSduId ⌈

Parameter Name	FrTpRxSduId	
Parent Container	FrTpRxSdu	
Upstream mapping defined	–	
Description	This unique identifier is used for change parameter request or receive cancellation from PduR to FrTp. ImplementationType: PduIdType	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_FrTp_00028] Definition of EcucReferenceDef FrTpRxSduRef ⌈

Parameter Name	FrTpRxSduRef	
Parent Container	FrTpRxSdu	
Upstream mapping defined	–	
Description	Reference to a PDU in the global PDU structure.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.7 FrTpConnectionControl

[ECUC_FrTp_00007] Definition of EcucParamConfContainerDef FrTpConnection Control ⌈

Container Name	FrTpConnectionControl	
Parent Container	FrTpMultipleConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This container contains the configuration parameters to control a FlexRay TP connection.	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpAckType	1	[ECUC_FrTp_00003]
FrTpMaxFCWait	1	[ECUC_FrTp_00014]
FrTpMaxNbrOfNPduPerCycle	1	[ECUC_FrTp_00029]
FrTpMaxRn	1	[ECUC_FrTp_00017]
FrTpSCexp	1	[ECUC_FrTp_00020]
FrTpTimeBr	1	[ECUC_FrTp_00047]
FrTpTimeCs	1	[ECUC_FrTp_00056]
FrTpTimeoutAr	1	[ECUC_FrTp_00032]
FrTpTimeoutAs	1	[ECUC_FrTp_00033]
FrTpTimeoutBs	1	[ECUC_FrTp_00034]
FrTpTimeoutCr	1	[ECUC_FrTp_00035]

No Included Containers

⌋

[ECUC_FrTp_00003] Definition of EcucEnumerationParamDef FrTpAckType ⌈

Parameter Name	FrTpAckType	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the type of acknowledgement which is used for the specific channel.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FRTP_ACK_WITH_RT	Acknowledgement with retry
	FRTP_NO	No acknowledgement
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00014] Definition of EcucIntegerParamDef FrTpMaxFCWait ⌈

Parameter Name	FrTpMaxFCWait	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the maximum number of FlowControl N-PDUs with FlowState "WAIT"	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00029] Definition of EcucIntegerParamDef FrTpMaxNbrOfNPduPer Cycle ⌈

Parameter Name	FrTpMaxNbrOfNPduPerCycle	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter is part of the ISO 10681-2 protocol's FlowControl parameter "Bandwidth Control (BC)". It limits the number of N-Pdus the sender is allowed to transmit within a FlexRay cycle.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 31	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00017] Definition of EcucIntegerParamDef FrTpMaxRn ⌈

Parameter Name	FrTpMaxRn	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the maximum number of retries (if retry is configured).	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00020] Definition of EcucIntegerParamDef FrTpSCexp ⌈

Parameter Name	FrTpSCexp
Parent Container	FrTpConnectionControl
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	This parameter is part of the ISO 10681-2 protocol's FlowControl parameter "Bandwidth Control (BC)". It represents the exponent to calculate the minimum number of "Separation Cycles" the sender has to wait for the next transmission of an FrTp N-Pdu.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 7	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00047] Definition of EcucFloatParamDef FrTpTimeBr ⌈

Parameter Name	FrTpTimeBr	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the time in seconds the FrTp requires to transmit a corresponding FlowControl Frame. According to ISO 10681-2 this parameter is a performance requirement.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 0.255]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00056] Definition of EcucFloatParamDef FrTpTimeCs ⌈

Parameter Name	FrTpTimeCs	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the time in seconds between the sending of two CFs or between the sending of a CF and LF or between the reception of a FC and sending of the next CF.	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00032] Definition of EcucFloatParamDef FrTpTimeoutAr ⌈

Parameter Name	FrTpTimeoutAr
Parent Container	FrTpConnectionControl
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	This parameter states the timeout in seconds between the PDU transmit request of the Transport Layer to the FlexRay Interface and the corresponding confirmation of the FlexRay Interface on the receiver side (for FC or AF). Dependency: It is obvious that FRTP_TIME_BR + FRTP_TIMEOUT_AR < FRTP_TIMEOUT_BS must hold (because the transmission duration on the bus has also to be considered).	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00033] Definition of EcucFloatParamDef FrTpTimeoutAs ⌈

Parameter Name	FrTpTimeoutAs	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter specifies the timeout in seconds the FrIf shall confirm a transmitted Pdu to the FrTp via the LSduR. Dependency: It is obvious that FRTP_TIME_CS + FRTP_TIMEOUT_AS < FRTP_TIMEOUT_CR must hold (because the transmission duration on the bus has also to be considered).	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00034] Definition of EcucFloatParamDef FrTpTimeoutBs ⌈

Parameter Name	FrTpTimeoutBs	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the timeout in seconds for waiting for an FC or AF on the sender side in a 1:1 connection. Dependency: It is obvious that FRTP_TIME_BR + FRTP_TIMEOUT_AR < FRTP_TIMEOUT_BS must hold (because the transmission duration on the bus has also to be considered).	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

[ECUC_FrTp_00035] Definition of EcucFloatParamDef FrTpTimeoutCr ⌈

Parameter Name	FrTpTimeoutCr	
Parent Container	FrTpConnectionControl	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This parameter defines the timeout value in seconds a receiver is waiting for a CF or a LF. Dependency: It is obvious that FRTP_TIME_CS + FRTP_TIMEOUT_AS < FRTP_TIMEOUT_CR must hold (because the transmission duration on the bus has also to be considered).	
Multiplicity	1	
Type	EcucFloatParamDef	
Range	[0 .. 65.535]	
Default value	–	
Post-Build Value	Loadable	Selectable

⌋

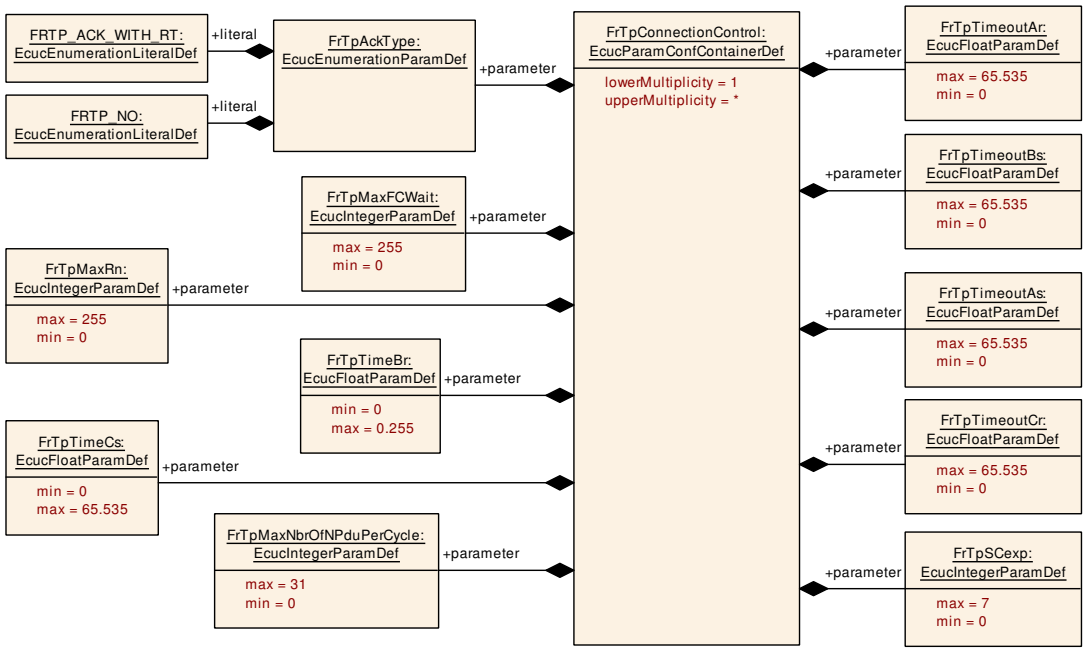


Figure 10.4: FrTpConnectionControl

10.2.8 FrTpTxPduPool

[ECUC_FrTp_00038] Definition of EcucParamConfContainerDef FrTpTxPduPool ⌈

Container Name	FrTpTxPduPool
Parent Container	FrTpMultipleConfig
Upstream mapping defined	[9, CP_TPS_SystemTemplate]

▽

△

Description	This container contains all Pdus that are assigned to that Pdu Pool.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
FrTpTxPdu	1..*	Container to hold the PDU parameters. ImplementationType: PduInfoType

⌋

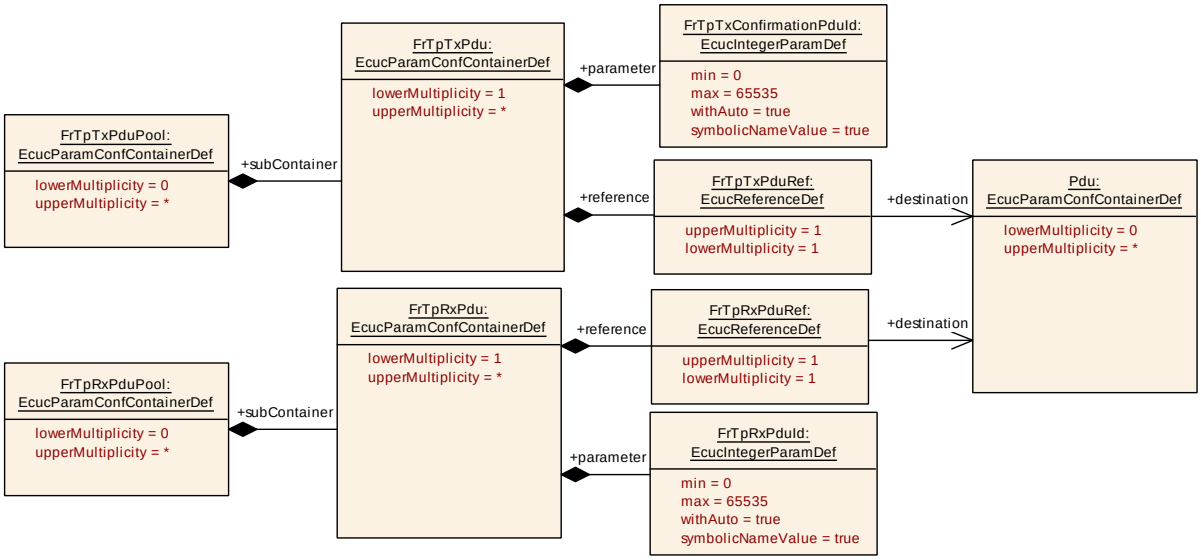


Figure 10.5: FrTpPduPool

10.2.9 FrTpRxPduPool

[ECUC_FrTp_00024] Definition of EcucParamConfContainerDef FrTpRxPduPool

⌈

Container Name	FrTpRxPduPool	
Parent Container	FrTpMultipleConfig	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	This container contains all Pdus that are assigned to that Pdu Pool.	
Multiplicity	0..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

No Included Parameters

Included Containers		
Container Name	**Multiplicity**	**Description**
FrTpRxPdu	1..*	Container to hold the PDU parameters. ImplementationType: PduInfoType

⌋

10.2.10 FrTpTxPdu

[ECUC_FrTp_00037] Definition of EcucParamConfContainerDef FrTpTxPdu ⌈

Container Name	FrTpTxPdu	
Parent Container	FrTpTxPduPool	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpTxConfirmationPduId	1	[ECUC_FrTp_00049]
FrTpTxPduRef	1	[ECUC_FrTp_00040]

No Included Containers

⌋

[ECUC_FrTp_00049] Definition of EcucIntegerParamDef FrTpTxConfirmationPduId ⌈

Parameter Name	FrTpTxConfirmationPduId	
Parent Container	FrTpTxPdu	
Upstream mapping defined	–	
Description	Handle Id to be used by the FrIf to confirm the transmission of the FrTpTxPdu via the LSduR to the FrTp module (FrTp_TxConfirmation) and for requesting TriggerTransmit via LSduR at the FrTp module (FrTp_TriggerTransmit).	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_FrTp_00040] Definition of EcucReferenceDef FrTpTxPduRef ⌈

Parameter Name	FrTpTxPduRef	
Parent Container	FrTpTxPdu	
Upstream mapping defined	–	
Description	Reference to a PDU in the global PDU structure.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.11 FrTpRxPdu

[ECUC_FrTp_00022] Definition of EcucParamConfContainerDef FrTpRxPdu ⌈

Container Name	FrTpRxPdu	
Parent Container	FrTpRxPduPool	
Upstream mapping defined	[9, CP_TPS_SystemTemplate]	
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType	
Multiplicity	1..*	
Post-Build Multiplicity	Loadable	Selectable
Configuration Parameters		

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTpRxPduId	1	[ECUC_FrTp_00023]
FrTpRxPduRef	1	[ECUC_FrTp_00026]

No Included Containers

⌋

[ECUC_FrTp_00023] Definition of EcucIntegerParamDef FrTpRxPduId ⌈

Parameter Name	FrTpRxPduId	
Parent Container	FrTpRxPdu	
Upstream mapping defined	–	
Description	This is a unique identifier for a received message which is forwarded from the FrIf via the LSduR to the FrTp. ImplementationType: PduIdType	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65535	
Default value	–	
With Auto	true	

▽

△

Pre-Compile only	true

⌋

[ECUC_FrTp_00026] Definition of EcucReferenceDef FrTpRxPduRef ⌈

Parameter Name	FrTpRxPduRef	
Parent Container	FrTpRxPdu	
Upstream mapping defined	–	
Description	Reference to a PDU in the global PDU structure.	
Multiplicity	1	
Type	Reference to Pdu	
Post-Build Value	Loadable	Selectable

⌋

10.2.12 FrTpDemEventParameterRefs

[ECUC_FrTp_00057] Definition of EcucParamConfContainerDef FrTpDemEvent ParameterRefs ⌈

Container Name	FrTpDemEventParameterRefs
Parent Container	FrTp
Upstream mapping defined	–
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_-RECEIVED	0..1	[ECUC_FrTp_00065]

No Included Containers

⌋

[ECUC_FrTp_00065] Definition of EcucReferenceDef FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_RECEIVED ⌈

Parameter Name	FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_RECEIVED
Parent Container	FrTpDemEventParameterRefs
Upstream mapping defined	–
Description	A Reference to DemEventParameter element which shall be invoked using the API Dem_SetEventStatus in case FRTP_E_SWAPPED_CONSECUTIVE_FRAMES_RECEIVED error occurs. The EventId is taken from the referenced DemEvent Parameter's DemEventId symbolic value.
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

10.3 Published Information

For details refer to [6] Chapter 10.3 *"Published Information"*.

10.4 Configuration dependencies and recommendation

The FrTp module functionality is based on several configuration parameters. To guarantee a well working software module this chapter gives some recommendation to the configuration parameter set. These rules shall be part of consistency checks of configuration tools.

10.4.1 Retry behaviour

The term of retry is used several times within this document but always with different focus. As depict in Figure 10.6 the FrTp module has basically two different retry behaviours:

1. Multiple API calls in case of an error or a busy system
2. Retry of PDU transfer depending on transport protocol conditions

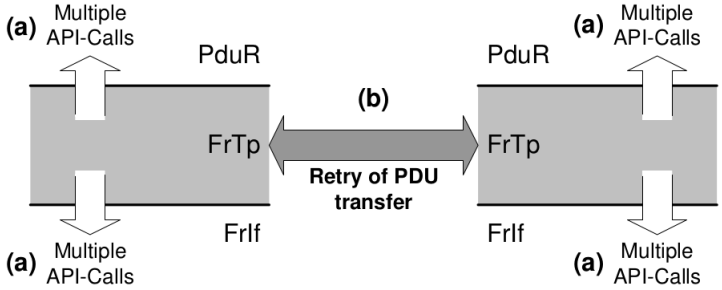


Figure 10.6: FrTp Retry Scenarios

Only for case (2) Retry of PDU transfer a global switch for enabling and disabling is defined (FrTpAckType). All other cases depend on the configuration of the different counters for the API calls: FRTP_MAX_FCWAIT, FRTP_MAX_AR.

10.4.2 TP-Acknowledgement and Retry

Acknowledgement and retry is only possible on 1:1 connections because the communication nodes have to deal with the FlowControl N-PDU parameters. FlowControl is not allowed for 1:n connections.

[SWS_FrTp_00598] Configuration parameter FrTpAckType not relevant when FrTpMultipleReceiverCon is defined ⌈If configuration parameter FrTpMultipleReceiverCon is set for a connection, no Acknowledgement and retry shall be supported irrespective whether the configuration parameter FrTpAckType is set or not.⌋

10.4.3 Timing and Timeout Parameters

Timing and timeout behaviour depends on the global FlexRay schedule.

To guarantee a stable system some timing and timeout relations shall be taken into account. The timeout behaviour is defined in chapter 7.5.8.

[SWS_FrTp_01154] Configuration dependency AS timeout to AS time ⌈For timeout As configuration it shall be considered that FRTP_TIMEOUT_AS > FRTP_TIME_AS⌋

[SWS_FrTp_01155] Configuration dependency AR timeout to AR time ⌈For timeout Ar configuration it shall be considered that FRTP_TIMEOUT_AR > FRTP_TIME_AR⌋

[SWS_FrTp_00599] Configuration dependency BS timeout to BR and AR time ⌈For timeout Bs configuration it shall be considered that FRTP_TIMEOUT_BS > FRTP_TIME_BR + FRTP_TIME_AR⌋

[SWS_FrTp_01153] Configuration dependency CR timeout to CS and AS time ⌈For timeout Cr configuration it shall be considered that FRTP_TIMEOUT_CR > FRTP_TIME_CS + FRTP_TIME_AS⌋

Note: FRTP_TIME_AR, FRTP_TIME_AS, FRTP_TIME_BR and FRTP_TIME_CS are performance timing values and depend on the global FlexRay schedule. To calculate that values please refer to the formulas at [1].

[SWS_FrTp_00180] Configuration dependency cycle time and CR timeout ⌈The FrTp configuration shall ensure that $(2^{SeparationCycleExponent} - 1) * t_{CycleTime} \leq FRTP_TIMEOUT_CR$ [1].⌋

[1] This is to prevent a timeout. Please refer to [1].

10.4.4 Bandwidth Control Configuration

It could occur that an ECU is not able to receive as much PDUs as defined within the PDU-Pool referenced by a connection[2]. In that case the bandwidth has to be limited by the receiver. There are three possibilities to limit the bandwidth for a connection link:

- Limit Bandwidth by Parameter FlowControl.BandwidthControl.MaxPduPerCycle in combination with Hardware FIFO buffer mechanisms[3].
- Limit Bandwidth by Parameter FlowControl.BandwidthControl.MaxPduPerCycle to reduce the number of allowed PDUs of the currently selected PDU-Pool.
- Limit Bandwidth by using a dedicated PDU-Pool for the connection to the affected Ecu.

10.4.4.1 BandwidthControl by FlowControl Parameter in combination with Hardware FIFO buffer mechanisms

If BandwidthControl by FlowControl Parameter in combination with Hardware FIFO buffer mechanisms is used no additional configuration restictions occur. This szenario implements "pure" [1] behaviour with focus on FlexRay 3.0 Hardware FIFO buffer mechanisms. The figure below shows the dependencies.

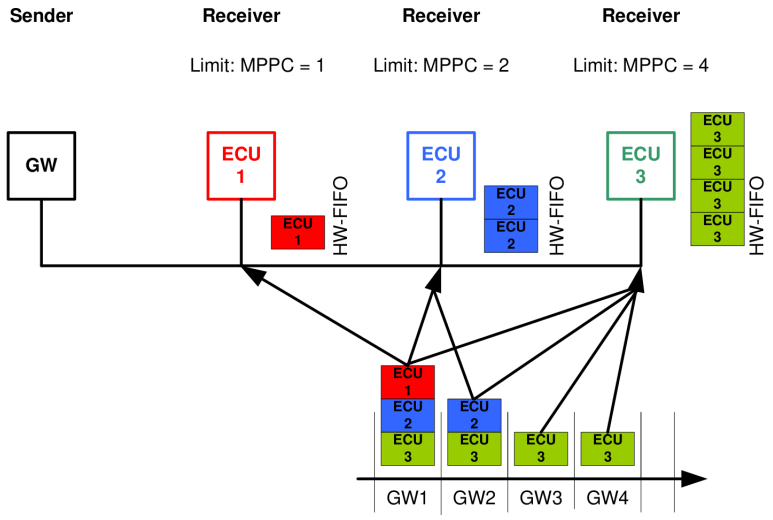


Figure 10.7: BandwidthControl by FlowControl Parameters in combination with HW FIFO buffer

[2]This is possible if a FlexRay Communication Controller only supports less Rx buffers and buffer reconfiguration at the end of a cycle is not possible or desired.

[3]This is an essential mechanism of FlexRay 3.0 protocol.

10.4.4.2 BandwidthControl by FlowControl Parameter

If BandwidthControl by FlowControl Parameter is used some configuration restictions have to be taken into account. The figure below shows the dependencies.

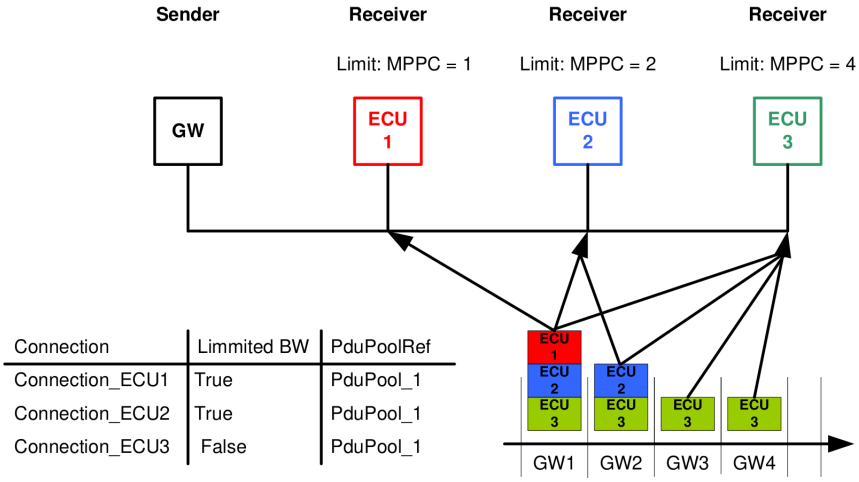


Figure 10.8: BandwidthControl by FlowControl Parameters

In a network four FlexRay nodes are connected. 4 PDUs are defined for the sender node. Two of the receivers have bandwidth limitations (ECU1 and ECU2).

The configuration restrictions are:

[SWS_FrTp_01173] Bandwidth limitation needs to be enabled when flow control parameter BC is used ⌈If bandwidth limitation in a connection to a certain ECU is realized by FlowControl parameter BC (without HW-FIFO mechanism), the attribute "FrTpBandwidthLimitation"within the corresponding connection shall be set to "TRUE".⌋

[SWS_FrTp_01174] Start frame handling in case of bandwith limitation ⌈If bandwidth limitation is realized by FlowControl parameter BC and if the attribute "FrTpBandwidthLimitation" is True, a Start Frame to initiate a communication link shall always be send in the first PDU of the referenced PDU-Pool. This is valid for both 1:1 and 1:n connections.

Note: The reason for using the first frame of the pool in case of bandwidth limitation is historical.

Bandwidth limitation is only required for FlexRay controllers with a limited number of buffers, which are then assigned to the first slots of the pool (the ones with the lowest numbers).⌋

[SWS_FrTp_01175] Flow control handling according to bandwith control parameter ⌈If an ECU responds with a FlowControl-Parameter BandwidthControl. MPPC $!=$ 0 ("zero") the sender shall use only the number of BC.MPPC PDUs of the PDU-Pool in ascending order to transmit data within that connection.⌋

[SWS_FrTp_01177] Flow control handling in case of limited bandwith ⌈If the attribute "FrTpBandwidthLimitation" is set to "TRUE", a Rx-connection shall use the first Pdu of the referenced Tx-Pdu-Pool for sending the required FlowControl frame to continue a communication link.⌋

10.4.4.3 BandwidthControl via different PDU Pools

If BandwidthControl is realized by different PDU Pools two different scenarios could occur.

10.4.4.3.1 BandwidthControl via non-overlapping Tx-Pdu-Pools

In case an ECU is not capable of receiving all Tx-Pdus sent for TP-communication in the FlexRay-cluster then non-overlapping Tx-Pdu-Pools can be configured as shown in the figure below:

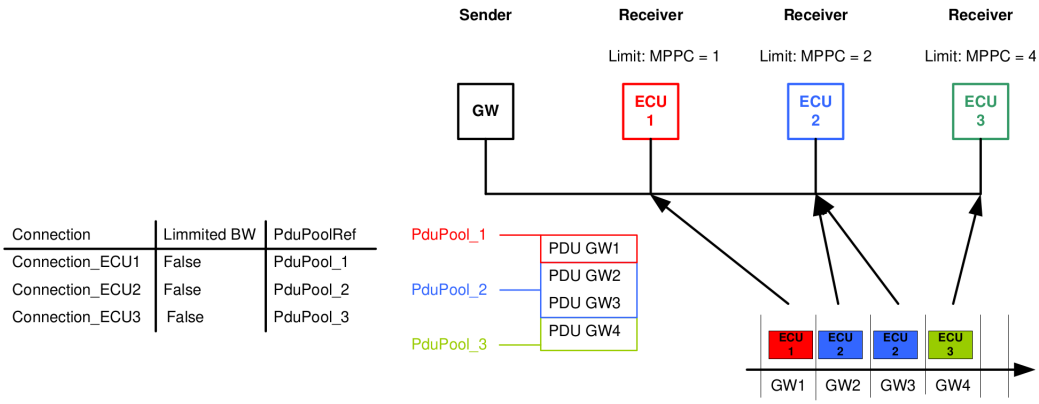


Figure 10.9: BandwidthControl by non-overlapping PDU Pools

10.4.4.3.2 BandwidthControl via overlapping Tx-Pdu-Pools

In case an Ecu is not capable of receiving all Tx-Pdus sent for TP- communication in the FlexRay-cluster then dedicated overlapping Tx-Pdu-Pools can be configured as shown in the figure below:

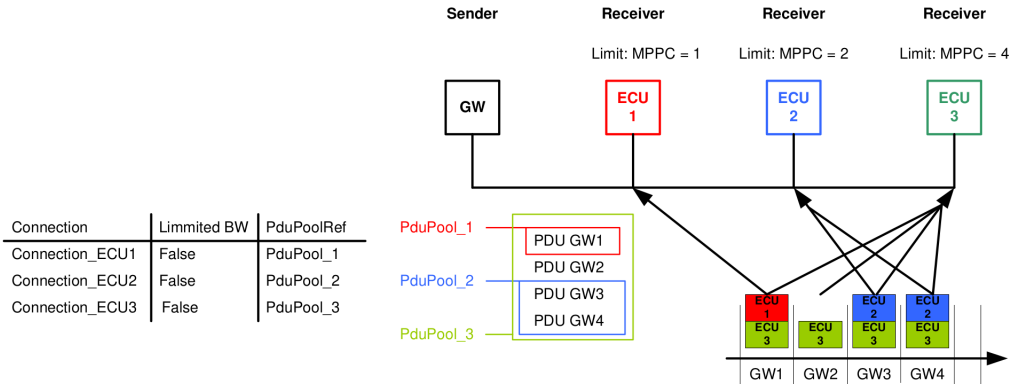


Figure 10.10: BandwidthControl by multiple PDU Pools

In the figure above ECU1 and ECU2 are not capable of receiving all Pdus GW1-GW4 shown above in their Rx-buffers ("Weak Ecu") and dedicated overlapping Pdu-Pools are configured and used. One Tx-Pdu can belong to more than one Tx-PDU-Pool at the same time[4].

[SWS_FrTp_01176] Support overlappig PDU-Pools ⌈It shall be possible to have overlapping PDU-Pools⌋

10.4.5 Configuration Requirements on the FlexRay Interface

If more than one Fr N-PDU is used for one Fr N-SDU within a connection, the FrIf shall guarantee that the Fr N-PDUs (Fr L-SDUs) are scheduled (sent over the bus) in the same order the FlexRay Transport Protocol Layer uses them, i.e. in ascending order regarding the Fr N-PDU IDs used in the FlexRay Transport Protocol Layer. Furthermore these PDUs shall be scheduled with the same frequency and within one Job (concerning the Joblist) in the FlexRay Interface module (since the reading of the PDU-Available Information for all PDUs of a connection has to be atomic.) This is necessary to avoid CFs coming out of order in a segmented transfer.

For every FrTp L-SDU the PDU-Update/Valid Information of the FrIf shall be activated.

For each transmitted N-Pdu a TransmitConfirmations shall be given by the FrIf module via the LSduR.

For every FrTp L-SDU no FrIf Trigger Transmit counter shall be utilized, i.e. the limit of the respective counter shall be 1. This is necessary in order to avoid multiple service primitive calls of `FrTp_TriggerTransmit` for the same Fr N-PDU in case e. g. a retry is necessary due to an timeout of the AS / AR timer.

[4]Reduced pools have to be taken into account for configuring the FlexRay-Driver of "weak Ecus".

A Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

A.1 Traceable item history of this document according to AUTOSAR Release R26-11

A.1.1 Added Specification Items in R26-11

Number	Heading
[SWS_FrTp_01103]	Timing parameter for FrTp module

Table A.1: Added Specification Items in R26-11

A.1.2 Changed Specification Items in R26-11

Number	Heading
[SWS_FrTp_00577]	Definition of mandatory interfaces required by module FrTp
[SWS_FrTp_01099]	FrTp timer support
[SWS_FrTp_01201]	Definition of development errors in module FrTp
[SWS_FrTp_91000]	Definition of imported datatypes of module FrTp

Table A.2: Changed Specification Items in R26-11

A.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_FrTp_00045]	Definition of EcucBooleanParamDef FrTpVersionInfoApi
[ECUC_FrTp_00058]	Definition of EcucReferenceDef FRTP_E_FRTPNAS_TIMEOUT_OCCURRED
[ECUC_FrTp_00059]	Definition of EcucReferenceDef FRTP_E_FRTPNAR_TIMEOUT_OCCURRED
[ECUC_FrTp_00060]	Definition of EcucReferenceDef FRTP_E_FRTPNBS_TIMEOUT_OCCURRED
[ECUC_FrTp_00062]	Definition of EcucReferenceDef FRTP_E_FRTPNCS_TIMEOUT_OCCURRED

▽

△

Number	Heading
[ECUC_FrTp_00063]	Definition of EcucReferenceDef FRTP_E_FRTPNCR_TIMEOUT_OCCURRED
[ECUC_FrTp_00064]	Definition of EcucReferenceDef FRTP_E_FRTPNBR_TIMEOUT_OCCURRED
[ECUC_FrTp_00066]	Definition of EcucReferenceDef FRTP_E_DROPPED_CONSECUTIVE_FRAMES_DETECTED
[ECUC_FrTp_00067]	Definition of EcucReferenceDef FRTP_E_FC_OVERFLOW_RECEIVED
[ECUC_FrTp_00068]	Definition of EcucReferenceDef FRTP_E_FC_OVERFLOW_TRANSMITTED
[ECUC_FrTp_00069]	Definition of EcucReferenceDef FRTP_E_FC_ABORT_RECEIVED
[ECUC_FrTp_00070]	Definition of EcucReferenceDef FRTP_E_FC_ABORT_TRANSMITTED
[SWS_FrTp_00141]	Import of standard return type
[SWS_FrTp_00215]	Definition of API function FrTp_GetVersionInfo
[SWS_FrTp_00361]	FR TP detects a C_As timeout.
[SWS_FrTp_00362]	FR TP detects a C_Ar timeout.
[SWS_FrTp_00363]	FR TP detects a C_Bs timeout.
[SWS_FrTp_00364]	FR TP detects a C_Br timeout.
[SWS_FrTp_00365]	FR TP detects a C_Cs timeout.
[SWS_FrTp_00366]	FR TP detects a C_Cr timeout.
[SWS_FrTp_00368]	FR TP detects missing consecutive frames.
[SWS_FrTp_00369]	FR TP receives FC.Ovflw frame.
[SWS_FrTp_00370]	FR TP transmits FC.Ovflw frame.
[SWS_FrTp_00371]	FR TP receives FC.Abt frame.
[SWS_FrTp_00372]	FR TP transmits FC.Abt frame.
[SWS_FrTp_00498]	Version info functionality depends on configuration parameter FrTpVersionInfoApi
[SWS_FrTp_01150]	Raise development error when get version info is called with invalid version info pointer
[SWS_FrTp_01164]	Import of version info type
[SWS_FrTp_01165]	Import of buffer request return type
[SWS_FrTp_01167]	Import of PDU ID type
[SWS_FrTp_01168]	Import of PDU info type
[SWS_FrTp_01169]	Import of PDU length type
[SWS_FrTp_01170]	Import of retry info type
[SWS_FrTp_01178]	Import of TP parameter type

Table A.3: Deleted Specification Items in R26-11

A.2 Traceable item history of this document according to AUTOSAR Release R25-11

A.2.1 Added Specification Items in R25-11

none

A.2.2 Changed Specification Items in R25-11

none

A.2.3 Deleted Specification Items in R25-11

none

B Not applicable requirements

[SWS_FrTp_NA_09999] Not applicable requirements

Upstream requirements: SRS_BSW_00004, SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00323, SRS_BSW_00345, SRS_BSW_00369, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00386, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00405, SRS_BSW_00409, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491

⌈These requirements are not applicable to this specification.⌋