Document Title	Specification of FlexRay ISO Transport Layer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	589
Document Classification	Standard
Document Version	5.2.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.03.2014	5.2.0	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.
31.10.2013	5.1.0	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

30.01.2013	5.0.0	AUTOSAR Administration	• Corrected Retry Handling Mechanism • Clarified usage of BUFREQ_E_BUSY • Removed references to ChangeParameterConfirmation • Removed private values in NotifResultType • Changes to support Harmonization of ECU Parameters concept • Updated scope value of configuration parameters
30.11.2011	4.0.0	AUTOSAR Administration	• Renaming (ISO) and new UID (029=>589) • API Names modified
29.10.2010	3.1.0	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 of up to 2^16-1 Bytes payload • Remove Chapter 7.5.4.3 with FrTp-1086 and FrTp-1087, remove COUNTER_BS, COUNTER_CR, Counter_TX_RN
30.11.2009	3.0.0	AUTOSAR Administration	• FrTp according to ISO 10681-2 • New PduR API • Legal disclaimer revised
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
15.05.2008	2.2.0	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.
17.12.2007	2.1.0	AUTOSAR Administration	• Clarified the role and purpose of the functions PduR_FrTpChangeParameterConfirmation() and PduR_FrTpCancelTransmitConfirmation() with respect to the PDU Router. • Document meta information extended • Small layout adaptations made
25.04.2006	2.0.0	AUTOSAR Administration	Document structure adapted to common Release 2.0 SWS Template.
19.09.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR Specification Documents may contain exemplary items (exemplary reference models, "use cases", and/or references to exemplary technical solutions, devices, processes or software).

Any such exemplary items are contained in the Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, nor any later documentation of AUTOSAR conformance of products actually implementing such exemplary items, imply that intellectual property rights covering such exemplary items are licensed under the same rules as applicable to the AUTOSAR Standard.

Table of Contents

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), the FlexRay Transport Protocol (FrTp) is placed between the PDU Router module and the FlexRay Interface 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.

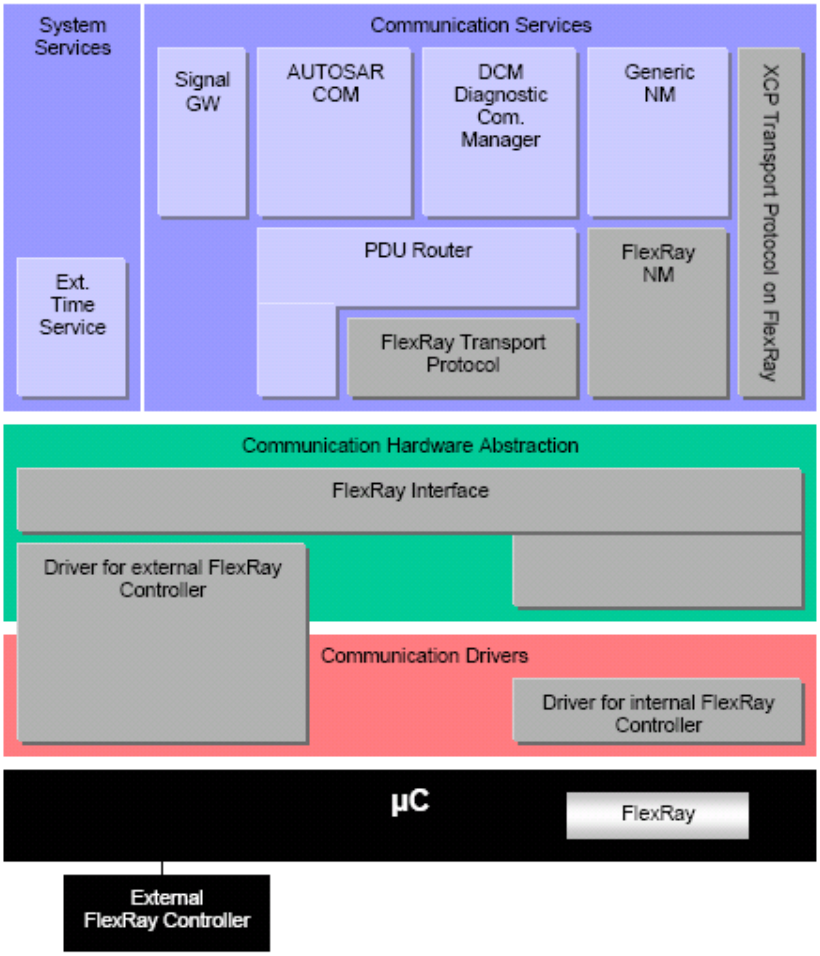


Figure 1: AUTOSAR FlexRay Layered Architecture

Figure 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.

FlexRay Interface (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 FlexRay Interface 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).

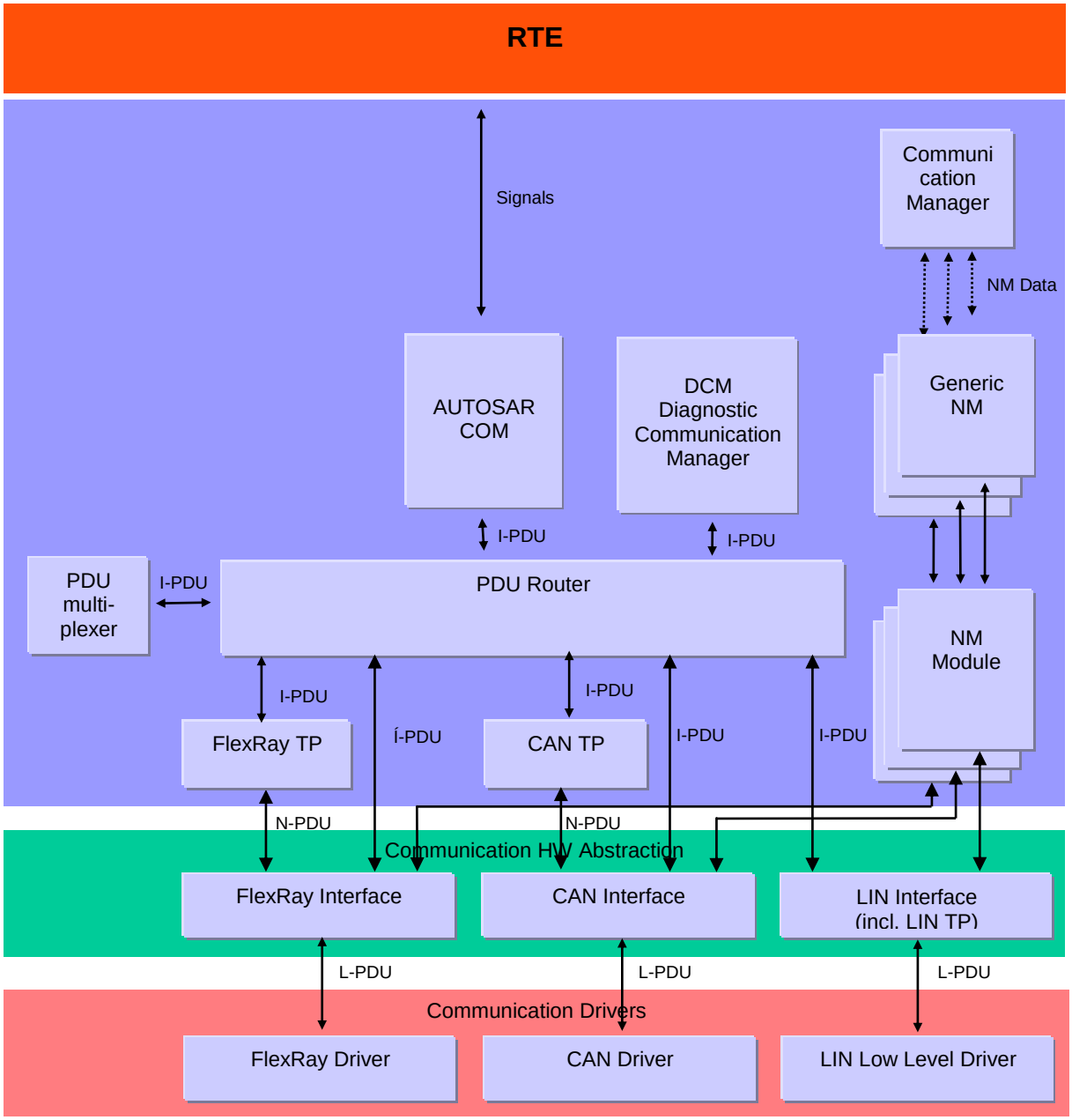


Figure 2 : AUTOSAR Signal Nomenclature

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 ISO 10681-2 [16]. 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.

All acronyms and abbreviations, which are specific to the FlexRay Transport Layer and are therefore not contained in the AUTOSAR glossary are described in the following:

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	Development Error Tracer
FC	Flow Control Fr N-PDU
Fr	FlexRay Driver module
FrIf	FlexRay Interface module
FrTp	FlexRay Transport Protocol Layer
PduR	PDU Router
SF	Start Frame Fr N-PDU
LF	Last Frame Fr N-PDU
TP	Transport Protocol Layer

Acronym:	*Description:*
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
HIS	Hersteller Initiative Software
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 ISO 10681-2)
AF	Acknowledge Frame (please refer to ISO 10681-2)
SN	Sequence Number (please refer to ISO 10681-2)
FrTp_As	Timer Parameter for a sender. Time for transmission of the FlexRay frame (any N_PDU) on the sender side. (please refer to ISO 10681-2)
FrTp_Ar	Timer Parameter for a receiver. Time for transmission of the FlexRay frame (any N_PDU) on the receiver side (please refer to ISO 10681-2)
FrTp_BS	Timer Parameter for a sender. Time until reception of the next FlowControl N_PDU. (please refer to ISO 10681-2)
FrTp_Br	Timer Parameter for a receiver. Time until transmission of the next FlowControl N_PDU. (please refer to ISO 10681-2)
FrTp_Cs	Timer Parameter for a sender. Time until transmission of the next ConsecutiveFrame N_PDU/LastFrame N_PDU. (please refer to ISO 10681-2)
FrTp_Cr	Timer Parameter for a receiver. Time until reception of the next ConsecutiveFrame N_PDU (please refer to ISO 10681-2)

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf.pdf

[3] General Requirements of Basic Software Modules
AUTOSAR_SRS_BSWGeneral.pdf

[4] Requirements on FlexRay
AUTOSAR_SRS_FlexRay.pdf

[5] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface.pdf

[6] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[7] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[8] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[9] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[10] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[11] Specification of ECU Configuration,
AUTOSAR_TPS_ECUConfiguration.pdf

[12] Specification of Diagnostic Event Manager,
AUTOSAR_SWS_DiagnosticEventManager.pdf

[13] Specification of Development Error Tracer,
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[14] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

[15] FlexRay Communications System Protocol Specification Version 2.1

[16] ISO10681-2, Road vehicles – Communication on FlexRay – Part 2: Communication Layer services

[17] HIS MISRA SubSet V2.0
www.automotive-his.de/download/HIS_SubSet_MISRA_C_2.0.pdf

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [14] (SWS BSW General), which is also valid for FlexRay ISO Trnasport Layer.

Thus, the specification SWS BSW General shall be considered as additional and required specification for FlexRay ISO Trnasport 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 FlexRay Interface Layer and a single upper PDU Router module.

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 ISO 10681-2 functionalities as listed below:

Functionality	Description
Status monitoring	ISO 10681-2 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	ISO 10681-2 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 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 FrIf modules.

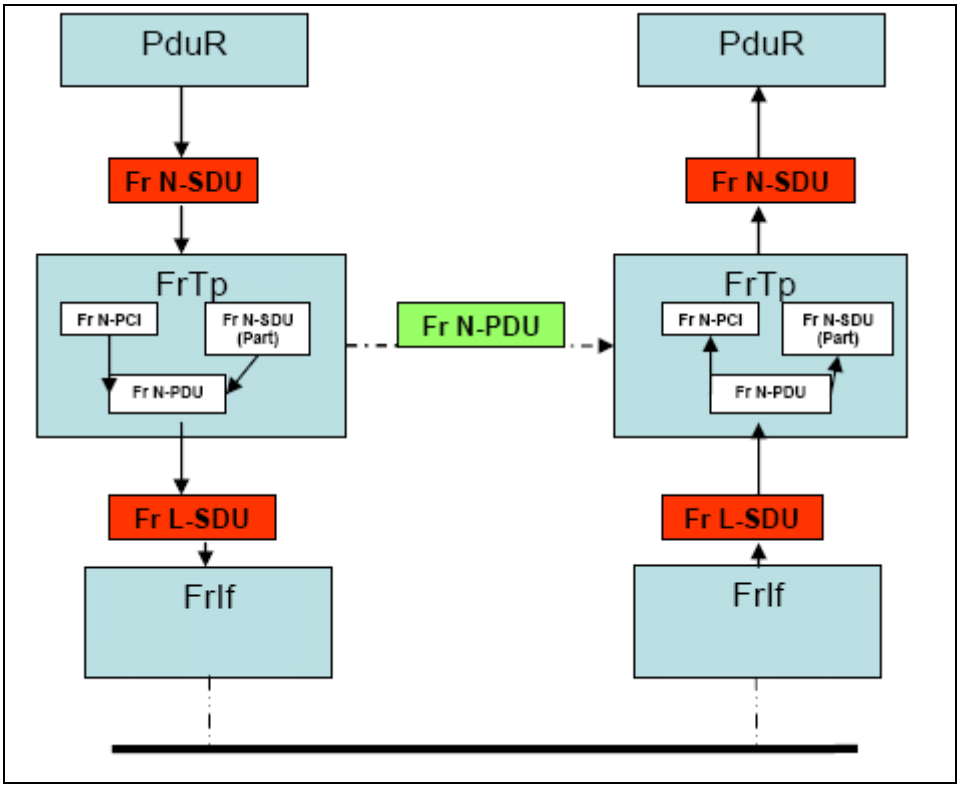


Figure 3: FrTp interactions

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 8.6 and specification [6]:

- ***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 FlexRay Interface

The following services primitives of the FlexRay Interface (FrIf) module are called by the FrTp:

- ***FrIf_Transmit***
  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 FlexRay Interface module:

- ***FrTp_RxIndication***
  By this API service primitive, the FlexRay Interface 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 FlexRay Interface module confirms the sending of the frame containing the Fr N-PDU over the FlexRay network.
- ***FrTp_TriggerTransmit***
  By this API service primitive, the FlexRay Interface 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 Development Error Tracing

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

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

[1] Depending on the configured buffer access mode.

5.6 File structure

5.6.1 Code file structure

For details refer to the chapter 5.1.6 “Code file structure” in *SWS_BSWGeneral.*

5.6.2 Header file structure

[SWS_FrTp_01157] ⌈FrTp.h shall contain all data exported from the FrTp – API declarations (except callbacks), extern types and global data.⌋()

[SWS_FrTp_01004] ⌈The header file structure of the FrTp module shall include the following files:

- FrIf.h – header file of FrIf,
- FrTp_MemMap.h – header file for Memory Mapping,
- Det.h – header file of Det,
- SchM_FrTp.h – header file of SchM declarations,
- PduR_FrTp.h – header file of PduR,
- Std_Types.h – header file for standard types
- ComStack_Types.h – header file for ComStack types
- FrTp_Types.h – header file for FrTp specific types
- FrTp_Cfg.h – header file for configuration parameters

⌋()

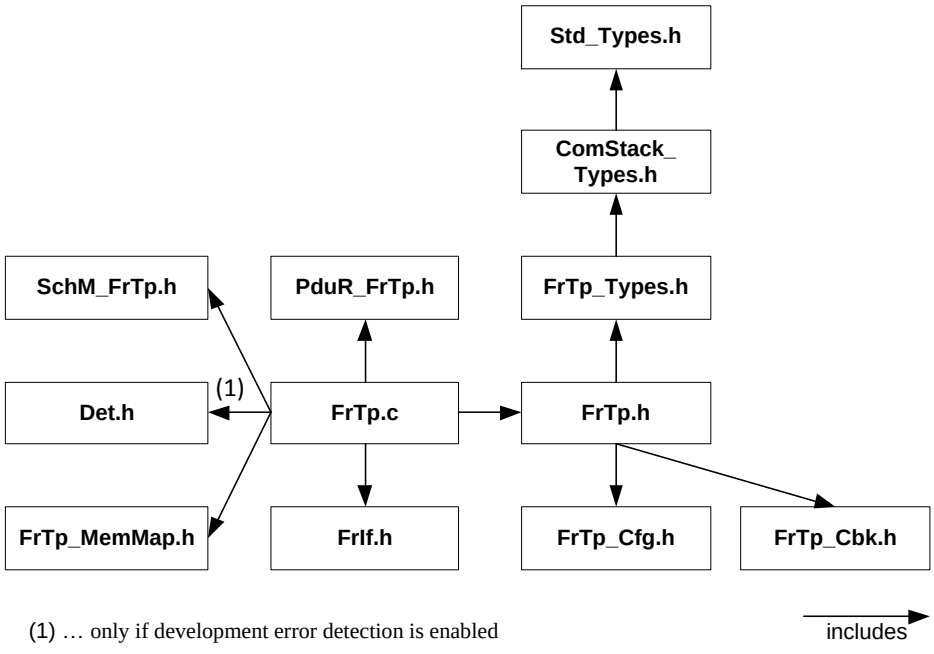


Diagram 1: File Structure

The structure as depict in Diagram 1 allows the separation between platform, compiler and implementation specific definitions and declarations from general definitions as well as the separation of source code and configuration.

5.6.3 Module Files Consistency

[SWS_FrTp_00200] ⌈Each code (*.c) file of the FrTp module shall provide data of version identification as defined in chapter 10.3.⌋(SRS_BSW_00004)

[SWS_FrTp_01158] ⌈Each header (*.h) file of the FrTp module shall provide data of version identification as defined in chapter 10.3.⌋()

5.6.4 Design rules

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

[SWS_FrTp_01129] ⌈The FlexRay Transport Layer module architecture shall support configuration modification by a dedicated update process (e.g. flash reprogramming)⌋(SRS_Fr_05123)

[2] No compiler specific keywords shall be used.

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_FrTp_00088
-	-	SWS_FrTp_00136
-	-	SWS_FrTp_00137
-	-	SWS_FrTp_00141
-	-	SWS_FrTp_00148
-	-	SWS_FrTp_00149
-	-	SWS_FrTp_00150
-	-	SWS_FrTp_00151
-	-	SWS_FrTp_00152
-	-	SWS_FrTp_00153
-	-	SWS_FrTp_00154
-	-	SWS_FrTp_00162
-	-	SWS_FrTp_00180
-	-	SWS_FrTp_00202
-	-	SWS_FrTp_00203
-	-	SWS_FrTp_00209
-	-	SWS_FrTp_00228
-	-	SWS_FrTp_00242
-	-	SWS_FrTp_00384
-	-	SWS_FrTp_00385
-	-	SWS_FrTp_00415
-	-	SWS_FrTp_00416
-	-	SWS_FrTp_00418
-	-	SWS_FrTp_00421
-	-	SWS_FrTp_00428
-	-	SWS_FrTp_00429
-	-	SWS_FrTp_00430
-	-	SWS_FrTp_00498
-	-	SWS_FrTp_00569
-	-	SWS_FrTp_00577
-	-	SWS_FrTp_00578
-	-	SWS_FrTp_00579
-	-	SWS_FrTp_00580
-	-	SWS_FrTp_00598
-	-	SWS_FrTp_00599
-	-	SWS_FrTp_01004

-	-	SWS_FrTp_01007
-	-	SWS_FrTp_01008
-	-	SWS_FrTp_01009
-	-	SWS_FrTp_01010
-	-	SWS_FrTp_01011
-	-	SWS_FrTp_01012
-	-	SWS_FrTp_01013
-	-	SWS_FrTp_01014
-	-	SWS_FrTp_01015
-	-	SWS_FrTp_01017
-	-	SWS_FrTp_01019
-	-	SWS_FrTp_01020
-	-	SWS_FrTp_01021
-	-	SWS_FrTp_01022
-	-	SWS_FrTp_01025
-	-	SWS_FrTp_01026
-	-	SWS_FrTp_01028
-	-	SWS_FrTp_01029
-	-	SWS_FrTp_01030
-	-	SWS_FrTp_01032
-	-	SWS_FrTp_01033
-	-	SWS_FrTp_01036
-	-	SWS_FrTp_01038
-	-	SWS_FrTp_01039
-	-	SWS_FrTp_01040
-	-	SWS_FrTp_01041
-	-	SWS_FrTp_01042
-	-	SWS_FrTp_01043
-	-	SWS_FrTp_01044
-	-	SWS_FrTp_01045
-	-	SWS_FrTp_01046
-	-	SWS_FrTp_01047
-	-	SWS_FrTp_01048
-	-	SWS_FrTp_01049
-	-	SWS_FrTp_01050
-	-	SWS_FrTp_01051
-	-	SWS_FrTp_01052
-	-	SWS_FrTp_01053
-	-	SWS_FrTp_01054

-	-	SWS_FrTp_01055
-	-	SWS_FrTp_01056
-	-	SWS_FrTp_01057
-	-	SWS_FrTp_01058
-	-	SWS_FrTp_01059
-	-	SWS_FrTp_01060
-	-	SWS_FrTp_01061
-	-	SWS_FrTp_01062
-	-	SWS_FrTp_01063
-	-	SWS_FrTp_01064
-	-	SWS_FrTp_01065
-	-	SWS_FrTp_01066
-	-	SWS_FrTp_01067
-	-	SWS_FrTp_01068
-	-	SWS_FrTp_01069
-	-	SWS_FrTp_01070
-	-	SWS_FrTp_01071
-	-	SWS_FrTp_01072
-	-	SWS_FrTp_01074
-	-	SWS_FrTp_01075
-	-	SWS_FrTp_01076
-	-	SWS_FrTp_01077
-	-	SWS_FrTp_01078
-	-	SWS_FrTp_01079
-	-	SWS_FrTp_01080
-	-	SWS_FrTp_01081
-	-	SWS_FrTp_01083
-	-	SWS_FrTp_01084
-	-	SWS_FrTp_01088
-	-	SWS_FrTp_01089
-	-	SWS_FrTp_01090
-	-	SWS_FrTp_01091
-	-	SWS_FrTp_01092
-	-	SWS_FrTp_01093
-	-	SWS_FrTp_01094
-	-	SWS_FrTp_01095
-	-	SWS_FrTp_01096
-	-	SWS_FrTp_01097
-	-	SWS_FrTp_01099

-	-	SWS_FrTp_01100
-	-	SWS_FrTp_01101
-	-	SWS_FrTp_01102
-	-	SWS_FrTp_01105
-	-	SWS_FrTp_01106
-	-	SWS_FrTp_01108
-	-	SWS_FrTp_01109
-	-	SWS_FrTp_01111
-	-	SWS_FrTp_01113
-	-	SWS_FrTp_01114
-	-	SWS_FrTp_01115
-	-	SWS_FrTp_01116
-	-	SWS_FrTp_01117
-	-	SWS_FrTp_01118
-	-	SWS_FrTp_01123
-	-	SWS_FrTp_01124
-	-	SWS_FrTp_01131
-	-	SWS_FrTp_01132
-	-	SWS_FrTp_01134
-	-	SWS_FrTp_01137
-	-	SWS_FrTp_01138
-	-	SWS_FrTp_01139
-	-	SWS_FrTp_01140
-	-	SWS_FrTp_01141
-	-	SWS_FrTp_01143
-	-	SWS_FrTp_01144
-	-	SWS_FrTp_01145
-	-	SWS_FrTp_01146
-	-	SWS_FrTp_01147
-	-	SWS_FrTp_01148
-	-	SWS_FrTp_01149
-	-	SWS_FrTp_01150
-	-	SWS_FrTp_01151
-	-	SWS_FrTp_01152
-	-	SWS_FrTp_01153
-	-	SWS_FrTp_01154
-	-	SWS_FrTp_01155
-	-	SWS_FrTp_01156
-	-	SWS_FrTp_01157

-	-	SWS_FrTp_01158
-	-	SWS_FrTp_01164
-	-	SWS_FrTp_01165
-	-	SWS_FrTp_01167
-	-	SWS_FrTp_01168
-	-	SWS_FrTp_01169
-	-	SWS_FrTp_01170
-	-	SWS_FrTp_01172
-	-	SWS_FrTp_01173
-	-	SWS_FrTp_01174
-	-	SWS_FrTp_01175
-	-	SWS_FrTp_01176
-	-	SWS_FrTp_01177
-	-	SWS_FrTp_01178
-	-	SWS_FrTp_01180
-	-	SWS_FrTp_01181
-	-	SWS_FrTp_01182
-	-	SWS_FrTp_01183
-	-	SWS_FrTp_01184
-	-	SWS_FrTp_01185
-	-	SWS_FrTp_01186
-	-	SWS_FrTp_01187
-	-	SWS_FrTp_01193
-	-	SWS_FrTp_01194
-	-	SWS_FrTp_01195
BSW00431	-	SWS_FrTp_09999
BSW00434	-	SWS_FrTp_09999
BSW05082	-	SWS_FrTp_09999
BSW05083	-	SWS_FrTp_01005, SWS_FrTp_01006
BSW05084	-	SWS_FrTp_01005, SWS_FrTp_01006
SRS_BSW_00004	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	SWS_FrTp_00200
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_FrTp_09999
SRS_BSW_00006	The source code of software modules above the æC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_FrTp_09999
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_FrTp_09999
SRS_BSW_00010	The memory consumption of all Basic SW Modules	SWS_FrTp_09999

	shall be documented for a defined configuration for all supported platforms.	
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_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_FrTp_09999
SRS_BSW_00160	Configuration files of AUTOSAR Basic SW module shall be readable for human beings	SWS_FrTp_09999
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_FrTp_09999
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_FrTp_09999
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_FrTp_09999
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_FrTp_09999
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_FrTp_09999
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_FrTp_09999
SRS_BSW_00305	Data types naming convention	SWS_FrTp_01133
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_FrTp_09999
SRS_BSW_00312	Shared code shall be reentrant	SWS_FrTp_09999
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_FrTp_09999
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_FrTp_09999
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_FrTp_09999
SRS_BSW_00326	-	SWS_FrTp_09999
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_FrTp_09999
SRS_BSW_00329	-	SWS_FrTp_09999
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_FrTp_09999
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_FrTp_09999
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_FrTp_09999
SRS_BSW_00335	Status values naming convention	SWS_FrTp_09999
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_FrTp_09999

SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_FrTp_09999
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_FrTp_09999
SRS_BSW_00347	A Naming seperation of different instances of BSW drivers shall be in place	SWS_FrTp_09999
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall apply a specific naming rule for enabling/disabling the detection and reporting of development errors	SWS_FrTp_09999
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_FrTp_09999
SRS_BSW_00370	-	SWS_FrTp_09999
SRS_BSW_00371	The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules	SWS_FrTp_09999
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_FrTp_09999
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_FrTp_09999
SRS_BSW_00376	-	SWS_FrTp_09999
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_FrTp_09999
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_FrTp_09999
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_FrTp_09999
SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_FrTp_09999
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_FrTp_09999
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_FrTp_00215
SRS_BSW_00409	All production code error ID symbols are defined by the Dem module and shall be retrieved by the other BSW modules from Dem configuration	SWS_FrTp_09999
SRS_BSW_00410	Compiler switches shall have defined values	SWS_FrTp_09999
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_FrTp_09999
SRS_BSW_00414	The init function may have parameters	SWS_FrTp_09999
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_FrTp_09999
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_FrTp_09999
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_FrTp_09999
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_FrTp_09999
SRS_BSW_00425	The BSW module description template shall provide	SWS_FrTp_09999

	means to model the defined trigger conditions of schedulable objects	
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_FrTp_09999
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_FrTp_09999
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_FrTp_09999
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_FrTp_09999
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_FrTp_09999
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_FrTp_09999
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_05089	The FlexRay Transport Layer services shall not be operational before initializing the module.	SWS_FrTp_01037
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

6.1 General Requirements on Basic Software Modules

Requirement		***Satisfied by***
SRS_BSW_00003	Version identification	
SRS_BSW_00305	Self-defined data types naming convention	**[SWS_FrTp_01133**
SRS_BSW_00306	Avoid direct use of compiler and platform specific keywords	n/a
SRS_BSW_00312	Shared code shall be reentrant	n/a
SRS_BSW_00314	Separation of interrupt frames and service routines	n/a
SRS_BSW_00318	Format of module version numbers	For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*
SRS_BSW_00321	Enumeration of module version numbers	For details refer to the chapter 10.3 “Published Information” in

		SWS_BSWGeneral.
SRS_BSW_00323	API parameter checking	n/a
SRS_BSW_00325	Runtime of interrupt service routines	n/a
SRS_BSW_00326	Transition from ISRs to OS tasks	n/a
SRS_BSW_00328	Avoid duplication of code	n/a
SRS_BSW_00329	Avoidance of generic interfaces	n/a
SRS_BSW_00330	Usage of macros / inline functions instead of functions	n/a
SRS_BSW_00331	Separation of error and status values	n/a
SRS_BSW_00333	Documentation of callback function context	n/a
SRS_BSW_00335	Status values naming convention	n/a
SRS_BSW_00336	Shutdown interface	Chapter 7.4
SRS_BSW_00337	Classification of errors	Chapter 7.7.3
SRS_BSW_00338	Detection and Reporting of development errors	Chapter 7.7.3
SRS_BSW_00339	Reporting of production relevant error status	Chapter 7.7.3
SRS_BSW_00341	Microcontroller compatibility documentation	n/a
SRS_BSW_00343	Specification and configuration of time	n/a
SRS_BSW_00345	Pre-compile-time configuration	n/a
SRS_BSW_00346	Basic set of module files	Chapter 5.6
SRS_BSW_00347	Naming separation of different instances of BSW drivers	n/a
SRS_BSW_00348	Standard type header	Chapter 5.6
SRS_BSW_00350	Development error detection keyword	n/a
SRS_BSW_00353	Platform specific type header	Chapter 5.6
SRS_BSW_00357	Standard API return type	Chapter 8.3
SRS_BSW_00358	Return type of init() functions	n/a
SRS_BSW_00359	Return type of callback functions	Chapter 8.2.1
SRS_BSW_00360	Parameters of callback functions	Chapter 8.2.1
SRS_BSW_00361	Compiler specific language extension header	Chapter 5.6
SRS_BSW_00369	Do not return development error codes via API	Chapter 7.7.3
SRS_BSW_00370	Separation of callback interface from API	n/a
SRS_BSW_00371	Do not pass function pointers via API	n/a
SRS_BSW_00373	Main processing function naming convention	n/a
SRS_BSW_00374	Module vendor identification	For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*
SRS_BSW_00375	Notification of wake-up reason	n/a
SRS_BSW_00376	Return type and parameters of main processing functions	n/a
SRS_BSW_00377	Module specific API return types	n/a
SRS_BSW_00379	Module identification	For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*
SRS_BSW_00383	List dependencies of configuration files	Chapter 5
SRS_BSW_00384	List dependencies to other modules	Chapter 5
SRS_BSW_00385	List possible error notifications	Chapter 8.2.1
SRS_BSW_00386	Configuration for detecting an error	n/a
SRS_BSW_00387	Specify the configuration class of callback function	n/a
SRS_BSW_00388	Introduce containers	Chapter 10.2

SRS_BSW_00389	Containers shall have names	Chapter 10.2
SRS_BSW_00390	Parameter content shall be unique within the module	Chapter 10.2
SRS_BSW_00391	Parameter shall have unique names	Chapter 10.2
SRS_BSW_00392	Parameters shall have a type	Chapter 10.2
SRS_BSW_00393	Parameters shall have a range	Chapter 10.2
SRS_BSW_00394	Specify the scope of the parameters	Chapter 10.2
SRS_BSW_00395	List the required parameters (per parameter)	Chapter 10.2
SRS_BSW_00396	Configuration classes	Chapter 10.2
SRS_BSW_00397	Pre-compile-time parameters	Chapter 10.2
SRS_BSW_00398	Link-time parameters	Chapter 10.2
SRS_BSW_00399	Loadable Post-build time parameters	Chapter 10.2
SRS_BSW_00004	Version check	**[SWS_FrTp_00200**, FRTP201
SRS_BSW_00400	Selectable Post-build time parameters	Chapter 10.2
SRS_BSW_00401	Documentation of multiple instances of configuration parameters	n/a
SRS_BSW_00402	Published information	Chapter 10.3
SRS_BSW_00405	Reference to multiple configuration sets	n/a
SRS_BSW_00406	Check module initialization	Chapter 7.4
SRS_BSW_00407	Function to read out published parameters	**[SWS_FrTp_00215**
SRS_BSW_00409	Header files for production code error IDs	n/a
SRS_BSW_00410	Compiler switches shall have defined values	n/a
SRS_BSW_00413	Accessing instances of BSW modules	n/a
SRS_BSW_00414	Parameter of init function	n/a
SRS_BSW_00415	User dependent include files	n/a
SRS_BSW_00416	Sequence of Initialization	Chapter 7.4
SRS_BSW_00417	Reporting of Error Events by Non-Basic Software	n/a
SRS_BSW_00423	Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	n/a
SRS_BSW_00424	BSW main processing function task allocation	n/a
SRS_BSW_00425	Trigger conditions for schedulable objects	n/a
SRS_BSW_00426	Exclusive areas in BSW modules	n/a
SRS_BSW_00427	ISR description for BSW modules	n/a
SRS_BSW_00428	Execution order dependencies of main processing functions	n/a
SRS_BSW_00429	Restricted BSW OS functionality access	n/a
BSW00431	The BSW Scheduler module implements task bodies	n/a
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	n/a
SRS_BSW_00433	Calling of main processing functions	n/a
BSW00434	The Schedule Module shall provide an API for exclusive areas	n/a
SRS_BSW_00435	Header File Structure for the Basic Software Scheduler	Chapter 5.6.2
SRS_BSW_00436	Module Header File Structure for the Memory Mapping	Chapter 5.6.2
SRS_BSW_00005	No hard coded horizontal interfaces within MCAL	n/a
SRS_BSW_00006	Platform independency	n/a
SRS_BSW_00009	Module User Documentation	n/a
SRS_BSW_00010	Memory resource documentation	n/a
SRS_BSW_00101	Initialization interface	**[SWS_FrTp_00147**
SRS_BSW_00158	Separation of configuration from implementation	Chapter 5.6.1
SRS_BSW_00159	Tool-based configuration	n/a
SRS_BSW_00160	Human-readable configuration data	n/a
SRS_BSW_00161	Microcontroller abstraction	n/a

SRS_BSW_00162	ECU layout abstraction	n/a
SRS_BSW_00164	Implementation of interrupt service routines	n/a
SRS_BSW_00167	Static configuration checking	n/a
SRS_BSW_00168	Diagnostic Interface of SW components	n/a
SRS_BSW_00171	Configurability of optional functionality	Chapter 10.4
SRS_BSW_00172	Compatibility and documentation of scheduling strategy	n/a

6.2 Requirements on FlexRay

Requirement		*Satisfied by*
SRS_Fr_05073	Compliance with ISO 10681-2	**[SWS_FrTp_01005]**, **[SWS_FrTp_01006]**
SRS_Fr_05074	Location of the FlexRay Transport Layer software module.	Chapter 1
SRS_Fr_05075	Independence of the Network Configuration.	Chapter 1
SRS_Fr_05076	Support of at least 32 logical FlexRay Transport Layer Channels being used concurrently.	**ECUC_FrTp_00004**
SRS_Fr_05077	Identification of each N-SDU with a unique identifier.	**[SWS_FrTp_01018]**
SRS_Fr_05079	Individual properties of each N-SDU.	Chapter 10.2
BSW05082	Support acknowledgement without retry.	n/a
BSW05083	Support acknowledgement with retry.	**[SWS_FrTp_01005]**, **[SWS_FrTp_01006]**
BSW05084	Maximum length of PDUs	**[SWS_FrTp_01005]**, **[SWS_FrTp_01006]**
SRS_Fr_05088	Interface for initialization.	**[SWS_FrTp_01034]**, **[SWS_FrTp_01035]**
SRS_Fr_05089	The FlexRay Transport Layer services shall not be operational before initializing the module.	**[SWS_FrTp_01037]**
SRS_Fr_05090	Support of the Change Parameter Service according to ISO 10681-2.	Chapter 7.5.7
SRS_Fr_05093	Provide a transmit cancellation service to the sender and the receiver.	**[SWS_FrTp_01005]**, **[SWS_FrTp_01006]**, Chapter 7.5.6
SRS_Fr_05095	Dynamic control of used bandwidth.	**[SWS_FrTp_01005]**, **[SWS_FrTp_01006]**
SRS_Fr_05123	Modification of configuration data by a flashing process	**[SWS_FrTp_01129]**

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 behavior 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 ISO10681-2, Road vehicles – Communication on FlexRay – Part 2: Communication Layer services [16].

[SWS_FrTp_01005] 「If no explicit recommendation or requirement is defined, the FrTp module shall follow the specification ISO10681-2, Road vehicles – Communication on FlexRay – Part 2: Communication Layer services [16].」(SRS_Fr_05073, BSW05083, BSW05084, SRS_Fr_05093, SRS_Fr_05095)

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 4, 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.

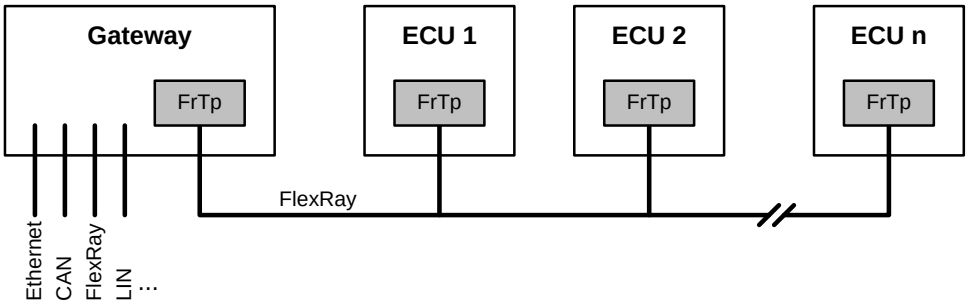


Figure 4: FrTp usage scenarios

7.2 FrTp behavior according to ISO10681-2

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

7.2.1 Protocol Data Unit (PDU)

[SWS_FrTp_01006] 「The FrTp module shall support the Fr N-PDU formats as defined in [16] ISO10681-2, Road vehicles – Communication on FlexRay – Part 2: Communication Layer services」(SRS_Fr_05073, BSW05083, BSW05084, SRS_Fr_05093, SRS_Fr_05095)

7.2.2 Frame Sequence charts

This chapter describes the data transfer modes based on Transport Protocol Layer frame (N-PDU) sequences according to ISO10681-2. 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 [16] ISO10681-2, Road vehicles – Communication on FlexRay – Part 2: Communication Layer services are relevant.

7.2.2.1 Unsegmented unacknowledged data transfer with known message length

[SWS_FrTp_01007] 「 According to ISO 10681-2 [16] the FrTp module shall support an unsegmented unacknowledged data transfer with known message length as depict in Figure 5.

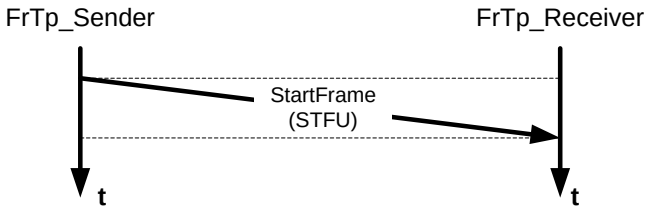


Figure 5: Frame sequence of an unsegmented unacknowledged data transfer with known message length ⌋()

7.2.2.2 Unsegmented acknowledged data transfer with known message length

[SWS_FrTp_01008] ⌈ According to ISO 10681-2 [16] the FrTp module supports an unsegmented acknowledged data transfer with known message length as depict in Figure 6.

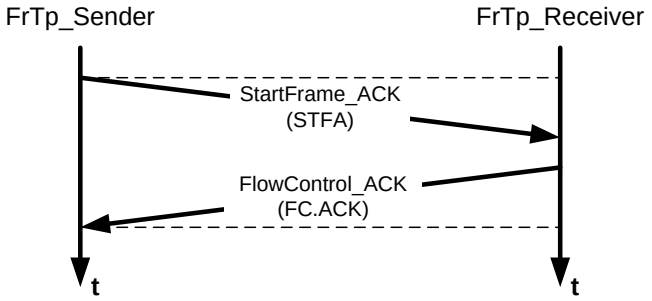


Figure 6: Frame sequence of an unsegmented acknowledged data transfer with known message length ⌋()

7.2.2.3 Segmented unacknowledged data transfer with known message length

[SWS_FrTp_01009] ⌈ According to ISO 10681-2 [16] the FrTp module shall support a segmented unacknowledged data transfer with known message length as depict in Figure 7.

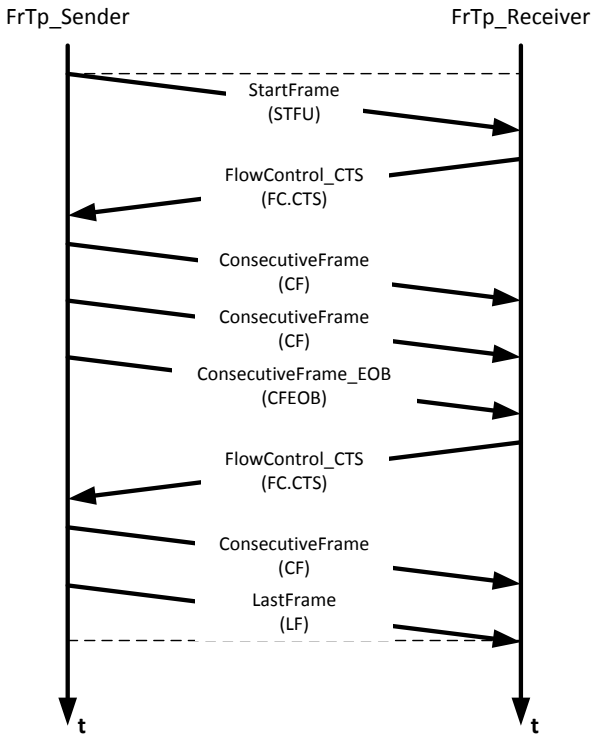


Figure 7: Frame sequence a segmented unacknowledged data transfer with known message length ()

7.2.2.4 Segmented acknowledged data transfer with known message length

[SWS_FrTp_01010] 「According to ISO 10681-2 [16] the FrTp module shall support a segmented acknowledged data transfer with known message length as depict in Figure 8.

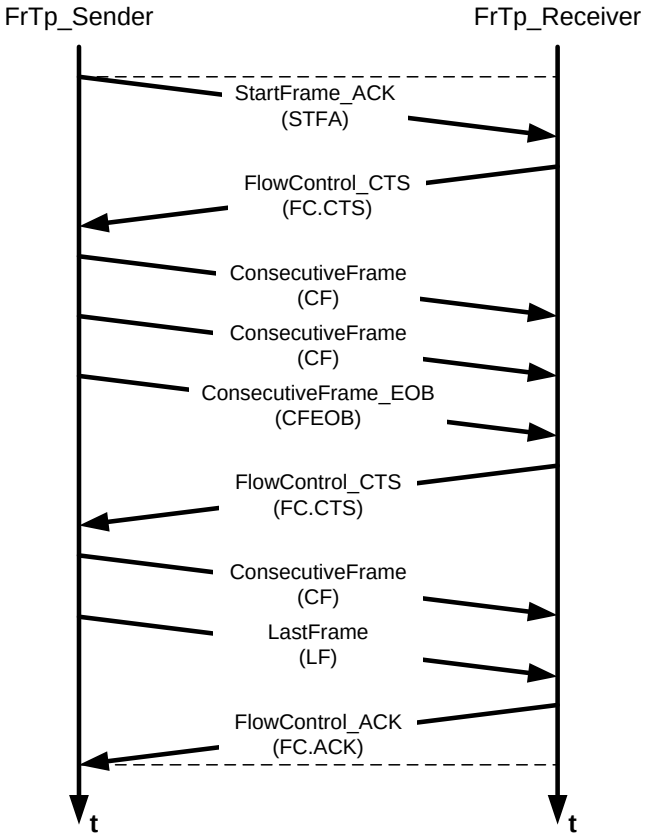


Figure 8: Frame sequence of a segmented acknowledged data transfer with known message length 」()

7.2.2.5 Segmented unacknowledged data transfer with unknown message length

[SWS_FrTp_01011] 「According to ISO 10681-2 [16] the FrTp module shall support a segmented unacknowledged data transfer with unknown message length as depict in Figure 9.

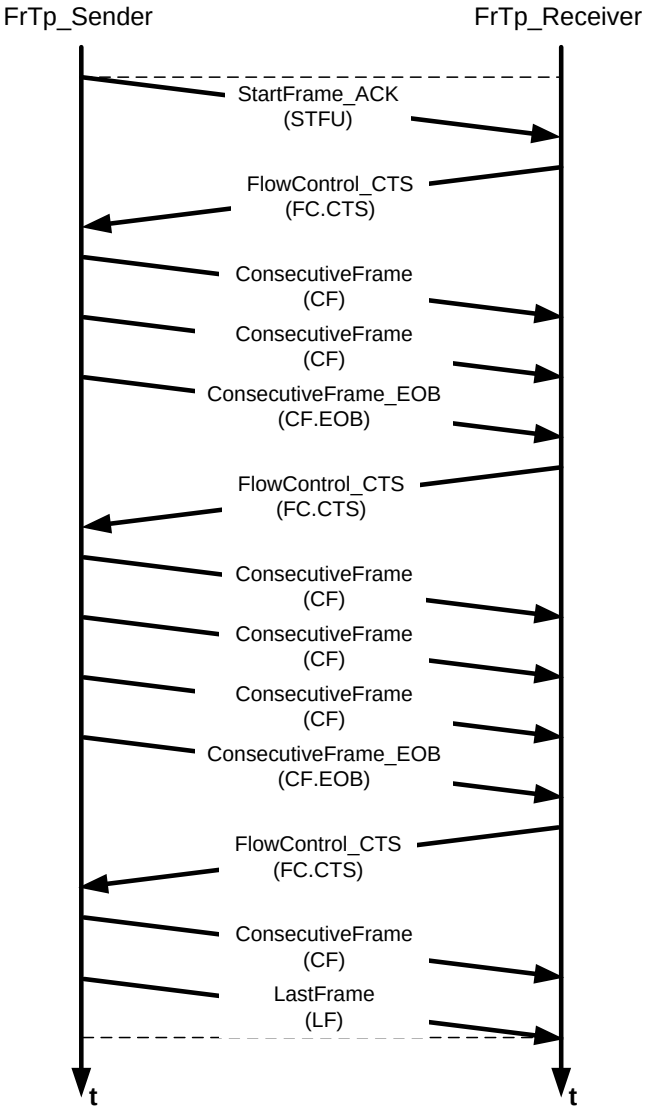


Figure 9: Frame sequence of a segmented unacknowledged data transfer with unknown message length ()

7.2.2.6 Segmented acknowledged data transfer with unknown message length

[SWS_FrTp_01012] ⌈According to ISO 10681-2 [16] the FrTp module shall support a segmented acknowledged data transfer with unknown message length as depict in Figure 10.

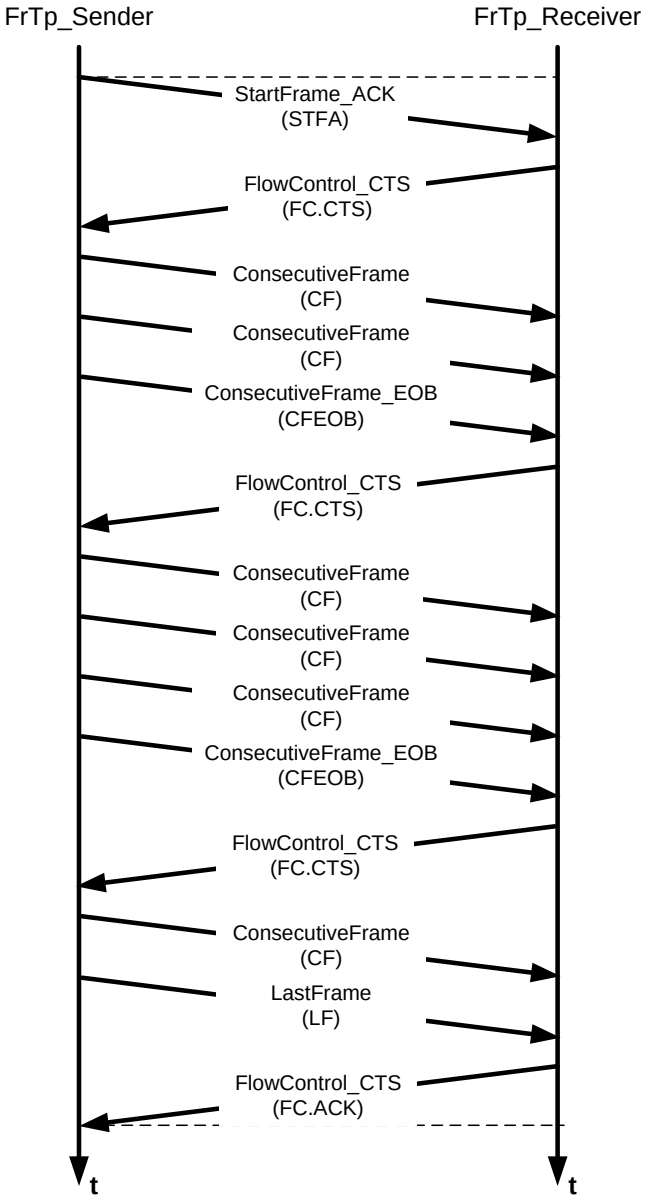


Figure 10: Frame sequence of a segmented acknowledged data transfer with unknown message length ⌋()

7.2.3 Limitation to ISO10681-2

The limitations to ISO 10681-2 are described in chapter 4.3 - Table 1.

7.3 Internal Module behavior specification

This chapter specifies the internal behaviour of the FlexRay Transport Layer module to fullfill the protocol behaviour according to ISO 10681-2 [13].

7.3.1 Overview

Figure 11 depicts an abstract overview of the FrTp layer module architecture.

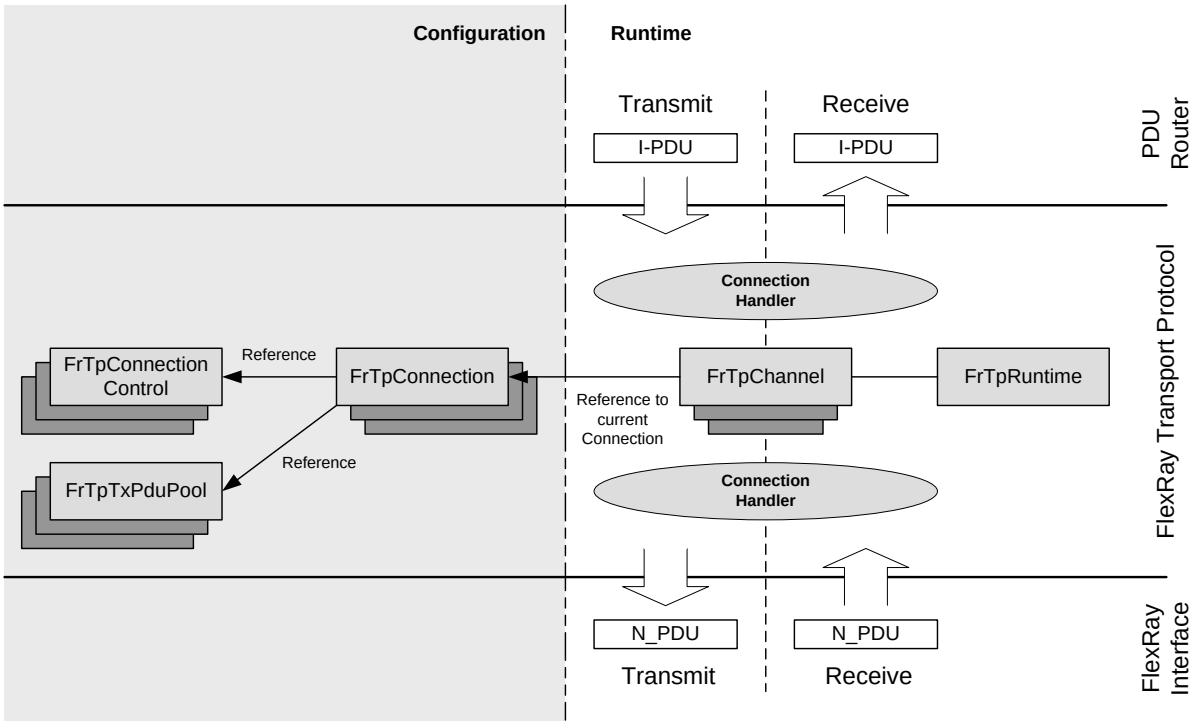


Figure 11: FlexRay Transport Module overview

Figure 11 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.
FrTpConnection	A connection configuration is a parameter set which

Control	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)
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) a target address of the receiver node(s) and
b) 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 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 (FrTpConnectionConfig).

[SWS_FrTp_01013] ⌈An `FrTpConnection` container shall implement all parameters as defined in chapter 10.2.⌋()

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

[SWS_FrTp_01015] ⌈ The FrTp module shall support a post build time configurable number of connections[3]. ⌋()

[SWS_FrTp_01017] ⌈Each `FrTpConnection` shall have a module wide unique `RemoteAddress / LocalAddress` pair `(see` section 10.2`)`.[4]⌋()

[SWS_FrTp_01018] ⌈ Each `FrTpConnection` shall have a module wide unique `FRTP_SDUID` (N-SDU ID) `(see` section 10.2`)`.⌋(SRS_Fr_05077)

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

[3] Post-build time configurable number of connections is required e.g. for gateways. If new connections are defined during vehicle lifecycle only the connection's parameter set has to be updated.
[4] The AUTOSAR local address and remote address is mapped to the ISO 10681-2 source address and target address.

each FrTpConnection has exactly one reference to one FrTpTxPduPool. (see also Figure 17). 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] ⌈ An FrTpTxPduPool container shall implement all parameters as defined in chapter 10.2.9⌋()

[SWS_FrTp_01020] ⌈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 values etc. Each FrTpConnection has an exclusive link to an FrTpConnectionControl container. FrTpConnections with equal control parameters can reference the same FrTpConnectionControl[5].

[SWS_FrTp_01021] ⌈ An FrTpConnectionControl Container shall implement all parameters as defined in chapter 10.2.⌋()

[SWS_FrTp_01022] ⌈An FrTpConnectionControl container can be referenced by different FrTpConnections.⌋()

7.3.3 Runtime data

As depict in Figure 11 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.

[5] Use case: Reducing configuration control container instances

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 ISO 10681-2.

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.

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 ISO 10681-2 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 ISO 10681-2 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.
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 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] ⌈The FrTp module shall support concurrently work of multiple `FrTpChannels`[6].⌋()

[SWS_FrTp_00088] ⌈The exact number of provided channels shall be configurable by the parameter FrTpChanNum (see section 10.2)⌋()

[SWS_FrTp_01025] ⌈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] ⌈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] ⌈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] ⌈The runtime variable *FrTpTxChannelState* shall be switched from "`busy`" state to "`idle`" state if the channel is free after an Tx connection is closed.⌋()

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

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

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[7] connection. Figure 12 depicts a Full Duplex implementation.

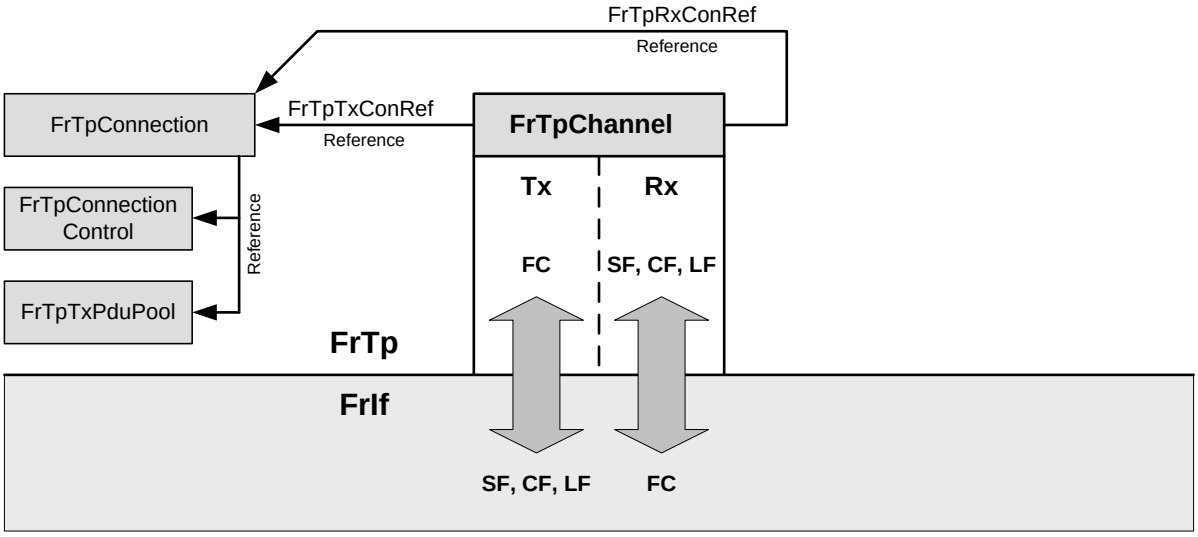


Figure 12: Full Duplex Overview

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 13 depicts a half duplex FrTp_Channel, where either a Tx or a Rx Connection is processed.

[7] 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 one 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.

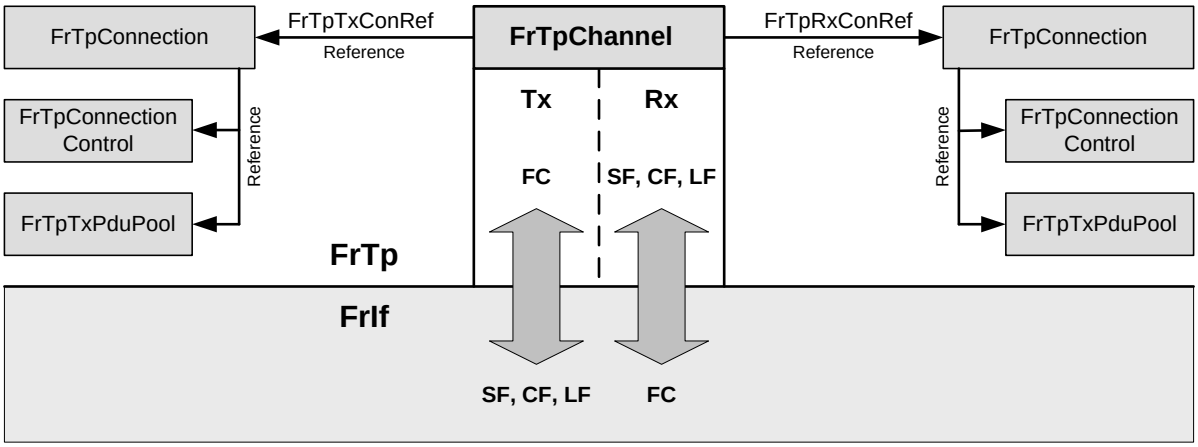


Figure 13: Half Duplex Overview

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

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] ⌈ The FrTp module shall have two internal states, `FRTP_OFF` and `FRTP_ON`.⌋()

[SWS_FrTp_01029] ⌈The FrTp module shall implement a static status variable *`FrTpState`* to denote whether the FrTp module is initialized or not[8].⌋()

[SWS_FrTp_01030] ⌈ The FrTp module shall be in the `FRTP_OFF` state after power up.⌋()

[SWS_FrTp_01032] ⌈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] ⌈The FrTp module shall performed normal FrTp operation tasks (e.g. segmentation, reassembly etc.) only when the FrTp module is in the `FRTP_ON` state[9].⌋()

[8] This variable is used for development error detection.

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

[SWS_FrTp_01034] ⌈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. ⌋(SRS_Fr_05088)

[SWS_FrTp_01035] ⌈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.⌋(SRS_Fr_05088)

[SWS_FrTp_01036] ⌈The FrTp module shall change to the internal state `FRTP_OFF` when the service primitive `FrTp_Shutdown()` has been executed successfully.⌋()

[SWS_FrTp_01037] ⌈The FrTp module shall raise an development error `FRTP_E_UNINIT` when
a) development error detection for the FrTp module is enabled and
b) any function (except `FrTp_GetVersionInfo`) is called before the function `FrTp_Init` has been called. ⌋(SRS_Fr_05089)

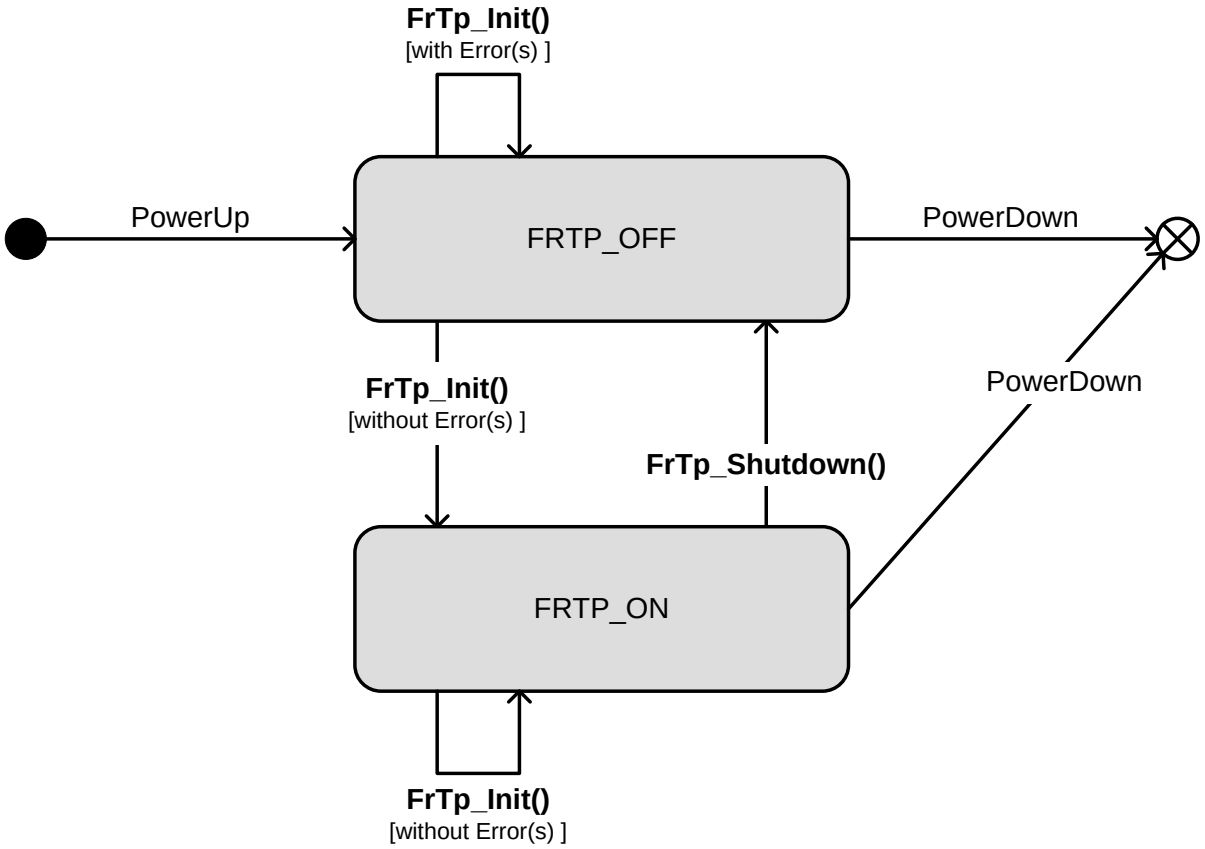


Diagram 2: FrTp Initialization and shutdown state diagram

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. 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:

a) Immediate Buffer Access Mode
b) 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] ⌈ The `TX_SDU_AVAILABLE` flag shall exist for every channel.⌋()

[SWS_FrTp_00416] ⌈ 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] ⌈The `TX_SDU_UNKNOWN_MSG_LENGTH` flag shall indicate an Fr N-SDU transmit request with unknown message length.⌋()

[SWS_FrTp_01102] ⌈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] ⌈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[10] 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] ⌈The RX_ERROR flag shall exist for every `FrTpChannel`.⌋()

[SWS_FrTp_00429] ⌈The RX_ERROR flag shall indicate that an error occurred during a segmented reception.⌋()

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

7.5.2 Transmit Data

[10] See ISO 10681-2 – chapter 6.5.7.2.3.

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.

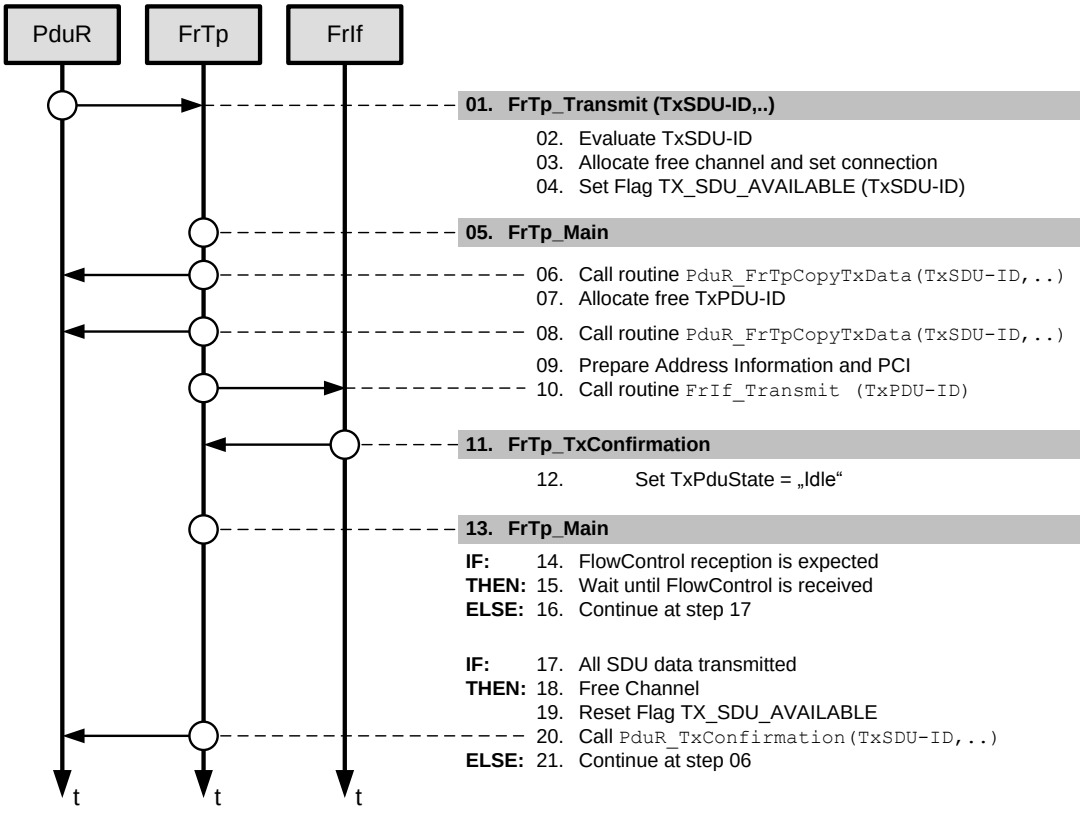


Figure 14: Transmit data overview in 'Immediate Buffer Access' mode

Figure 14 depicts the internal processing for data transmission in principle[11] if "Immediate Buffer Access" mode is configured[12]. Below there is a description of the different steps which are necessary to transmit data via FrTp.

Step 1 - 4

[SWS_FrTp_00136] ⌈Sending Fr N-SDU data shall always be initiated by the service primitive call of *FrTp_Transmit* (see chapter 8.3.3.1). ⌋()

[SWS_FrTp_01043] ⌈ The FrTp module shall evaluate the value of PduInfoType.SduLength:

SduLength = 0: Transmission with unknown message length is requested

[11] Figure 14 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.
[12] Buffer access mode is configured for each N-PDU and is referenced via the PDU-Pool.

SduLength ≠ 0: Transmission with known message length is requested.⌋()

[SWS_FrTp_01044] ⌈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** (see also chapter 7.5.1.2).⌋()

[SWS_FrTp_01134] ⌈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 `FrTpTxSduId` 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 `FrTpTxSduId` 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] ⌈An ongoing data transfer shall be signalled by the flag `TX_SDU_AVAILABLE` (see also chapter 7.5.1.1).⌋()

[SWS_FrTp_01039] ⌈ The service primitive shall set the flag `TX_SDU_AVAILABLE` only, if
a) the requested `FrTpTxSduId` is valid
b) a free channel resource is available (`FrTpTxChannelState = Idle`)⌋()

[SWS_FrTp_01040] ⌈If the current parameter `FrTpTxSduId` is not supported the service primitive `FrTp_Transmit`
a) shall be terminated and the return value shall be set to E_NOT_OK (see also chapter 8.2.1) and
b) 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] ⌈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 (see also chapter 8.2.1)[13].」()

[SWS_FrTp_01185] 「The FrTp module shall raise an development error FRTP_E_NO_CHANNEL when development error detection for the FrTp module is enabled.」()

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()[14] 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 FrIf_Transmit is called with the corresponding FrTp_TxPduId.

[SWS_FrTp_01042] 「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] 「The FrTp module shall always allocate the first free FrTpTxPdu while scanning the corresponding FrTpTxPduPool (see also chapter 7.3.2.2).」()

[SWS_FrTp_01046] 「If a free FrTpTxPdu is identified, the FrTp module shall use this FrTpTxPdu to continue current transmission process.」()

---

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

[14] 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 evaluate 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 optimisation (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_01047] ⌈If no free `FrTpTxPdu` is identified, the FrTp module shall stop processing for the corresponding connection within the current task.⌋()

[SWS_FrTp_01048] ⌈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` (see also Diagram 3). ⌋()

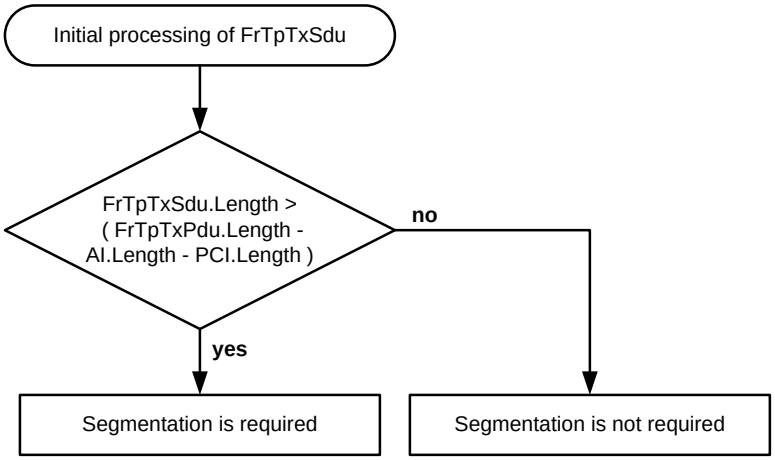


Diagram 3: Segmentation decision

Note: The decision whether segmentation is possible or not depends also on the connection mode (1:1 or 1:n). Please refer to chapter 7.5.2.4.

[SWS_FrTp_01123] ⌈ 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] ⌈The FrTp module shall prepare the Address Information and PCI according to the result of **[SWS_FrTp_01048** as defined in specification ISO 10681-2 [16].⌋()

[SWS_FrTp_01050] ⌈The FrTp module shall initiate an N-PDU data transfer by calling the service primitive `FrIf_Transmit`() with the `FrTpTxPduId` FrTpTxPduId of the recently allocated FrTpTxPdu.⌋()

[SWS_FrTp_01051] ⌈ The FrTp shall set the corresponding data length referenced by the service primitive `FrIf_Transmit`'s parameter `PduInfoType` to the exact data length of the buffer[15].⌋()

[15] FrTp transmits always the real amount of data stored 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 11 - 12

If the N-PDU was successfully transmitted by the FrIf module, the FrIf module shall call the service primitive `FrTp_TxConfirmation`. Within this service primitive the FrTp module shall reset the state of the corresponding `FrTpTxPdu`.

[SWS_FrTp_01052] ⌈The service primitive `FrTp_TxConfirmation` shall be called by the underlying layer module after a successful transmission of the corresponding N-PDU.⌋()

[SWS_FrTp_01053] ⌈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] ⌈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] ⌈The FrTp module shall implement a timing and timeout behaviour as defined in chapter 7.5.8⌋()

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] ⌈The FrTp module shall free the allocated channel (`FrTpTxChannelState = Idle`) if
a) all Tx N-SDU data are transmitted and
b) FrTp_TxConfirmation was given and
c) the final acknowledge is received in case acknowledge is configured.
⌋()

[SWS_FrTp_01057] ⌈ The FrTp module shall reset the flag `TX_SDU_AVAILABLE`, if:
a) all N-SDU data are transmitted and
b) FrTp_TxConfirmation was given and
c) the final acknowledge is received in case acknowledge is configured.
⌋()

[SWS_FrTp_01124] ⌈ The FrTp module shall reset the flag `TX_SDU_UNKNOWN_MSG_LENGTH`, if:
a) all N-SDU data are transmitted and
b) FrTp_TxConfirmation was given and
c) the final acknowledge is received in case acknowledge is configured or
d) if transmission was aborted and FrTp_TxConfirmation was given.⌋()

[SWS_FrTp_01058] ⌈The FrTp module shall always call the service primitive `PduR_FrTpTxConfirmation` for the corresponding `FrTpTxSduId` after the transmission request was accepted. The result shall be `E_OK` if
a) all N-SDU data are transmitted and
b) FrTp_TxConfirmation was given
c) the final acknowledge is received in case acknowledge is configured.
⌋()

7.5.2.2 Transmit Data via 'Decoupled Buffer Access' Mode

Figure 15 depicts the internal processing for data transmission in principle[16] if "Decoupled Buffer Access" mode is configured[17]. Below there is a description of the different steps which are necessary to transmit data via FrTp.

[16] Figure 15 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.

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

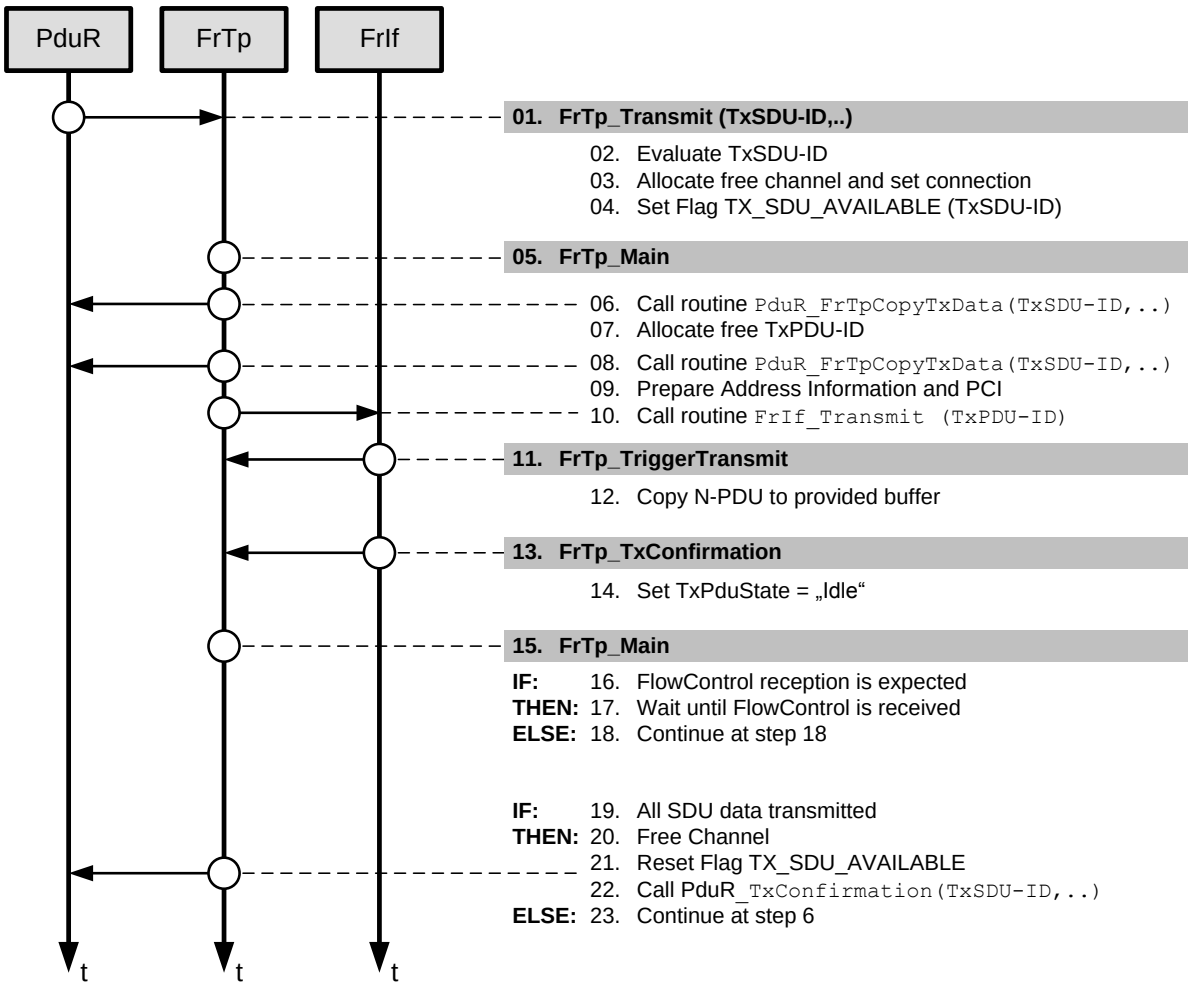


Figure 15: Transmit data overview in ‘Decoupled Buffer Access’ mode

Step 01 - 10

Step 01 - 10 in ‘decoupled access mode’ are equal to step 01 – 10 in ‘immediate access mode’. Please refer 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] 「The service primitive *FrTp_TriggerTransmit* shall be called by the FrIf module to propagate FrTp N-PDUs to the lower layers (e.g. FlexRay Driver).」()

[SWS_FrTp_01060] 「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] 「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 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] 「 The functionality to support data transmission with unknown message length shall be configurable by compiler switch.」()

[SWS_FrTp_01063] 「 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[18] and the service primitive call FrTp_Transmit shall be rejected with the return value E_NOT_OK.」()

[SWS_FrTp_01187] 「The FrTp module shall raise an development error FRTP_E_SEG_ERROR when
a) development error detection for the FrTp module is enabled and
b) a 1:n connection is requested as described in SWS_FrTp_01061.
」()

[SWS_FrTp_01064] 「 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') 」()

[18] Unknown message length data transfer requires segmentation because at least a StartFrame and a LastFrame have to be transmitted. Segmentation of 1:n connections is not allowed (see also chapter 7.5.2.4)

[SWS_FrTp_01065] ⌈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.⌋()

[SWS_FrTp_01066] ⌈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] ⌈ The FrTp module shall add all `PduInfoType.PduLength` values to calculate the total message length which is transmitted by the LastFrame (LF).⌋()

7.5.2.4 Segmentation condition for data transfer

FrTp module provides 1:1 connections as well as 1:n connections. According to ISO10681-2, the FrTp module shall refuse segmented 1:n connections. Due to the possibility of different PDU lengths within an FrTpTxPduPool the scenario of segmented 1:n connections could occur and shall be solved by the requirement(s) specified within this chapter.

[SWS_FrTp_01068] ⌈ If the `FrTpConnectionControl` parameter `FRTP_MULTIPLE_RECEIVER_CON` (see section 10.2) for the corresponding `FrTpConnection` is set, the communication handler shall not process a segmentation of an N-SDU and shall skip processing the corresponding `FrTpConnection` within the current task.⌋()

7.5.3 Receive Data

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

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

Figure 16 depicts the internal processing for data reception in principle[19]. Below there is a description of the different steps which are necessary to receive data via FrTp.

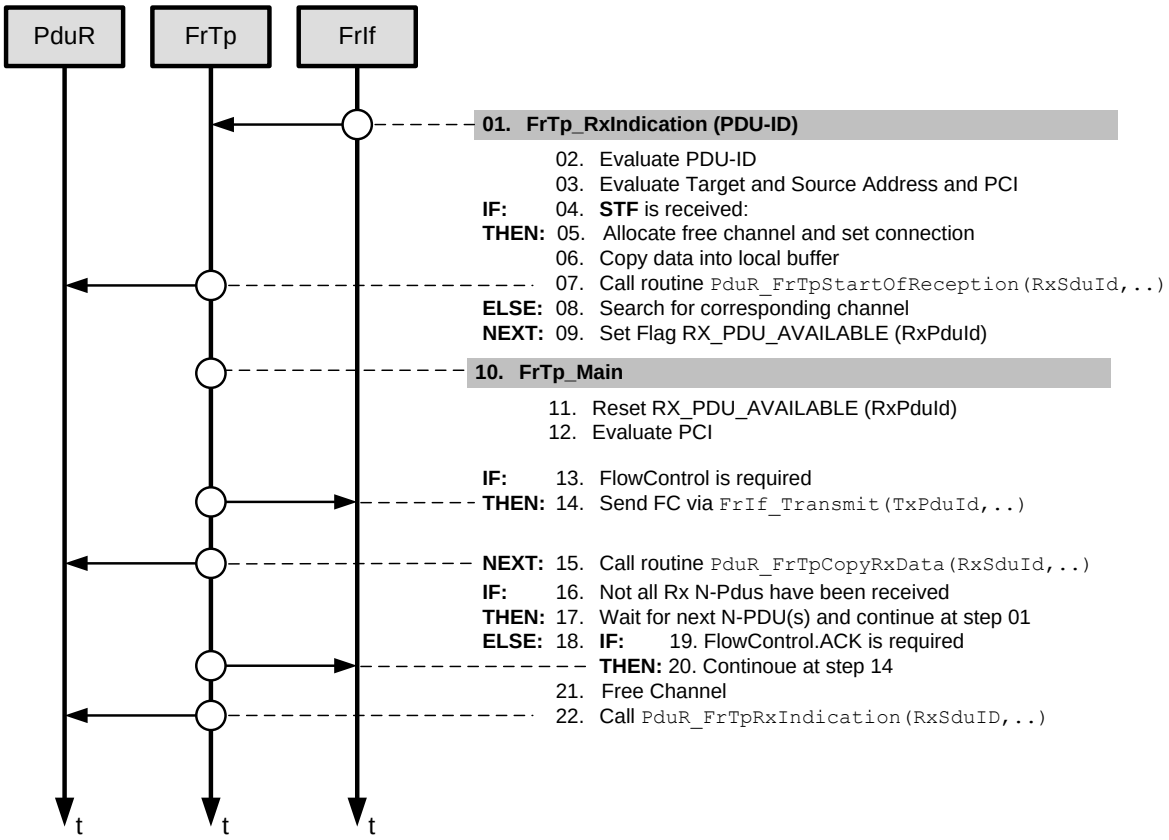


Figure 16: Receive data overview (FrTp_RxIndication() shall call PduR_FrTpStartOfReception() routine)

Step 1 - 9

[SWS_FrTp_00137] 「Receiving shall be initiated by the service primitive call *FrTp_RxIndication.*」()

[19] Figure 16 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.

The FrTp module shall validate the FrTpRxPduId. If an invalid FrTpRxPduId is received, the service primitive FrTp_RxIndication is terminated. For a valid FrTpRxPduId the FrTp module evaluates the target and source address and selects the corresponding FrTpConnection. 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.

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] ⌈The FrTp module shall process a received FrTp N-PDU only if
a) a valid FrTpRxPduId is received and
b) the FrTp N-PDU's address information matches to the configured FrTpConnection address information.⌋()

[SWS_FrTp_01070] ⌈If an invalid (undefined) FrTpRxPduId is received the FrTp module shall
a) ignore the FrTp N-PDU and
b) shall raise an development error FRTP_E_INVALID_PDU_SDU_ID when development error detection for the FrTp module is enabled.⌋()

[SWS_FrTp_01071] ⌈A matching FrTpConnection is only identified if
c) the received FrTp N-PDU's "Target Address" (see ISO 10681-2) is equal to the configured FrTpConnection's Local Address (FrTpLa, see section 10.2) and
d) the received FrTp N-PDU's "Source Address" (see ISO 10681-2) is equal to the configured FrTpConnection's Remote Address (FrTpRa, see section 10.2).⌋()

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

[SWS_FrTp_01074] ⌈ The service primitive shall set the flag RX_PDU_AVAILABLE only, if
a) the requested FrTpRxPduId is valid and
b) the address check matches to a configured FrTpConnection and
c) a free channel resource is available (FrTpRxChannelState = Idle)⌋()

[SWS_FrTp_01075] ⌈If the current parameter `FrTpRxPduId` is not supported the service primitive `FrTp_RxIndication` shall be terminated without any further action.[20]⌋()

[SWS_FrTp_01076] ⌈If no free channel is available (`FrTpRxChannelState ≠ Idle`) the service primitive `FrTp_RxIndication` shall be terminated without any further action.[21]⌋()

[SWS_FrTp_01186] ⌈The FrTp module shall raise an development error FRTP_E_NO_CHANNEL when development error detection for the FrTp module is enabled⌋()

[SWS_FrTp_01077] ⌈Within the service primitive *`FrTp_RxIndication`* the FrTp module shall copy the received StartFrame PDU into a local buffer[22]. ⌋()

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

[SWS_FrTp_01193] ⌈ 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 ISO 10681-2 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 a service primitive call *`FrIf_Transmit`* with the corresponding `FrTpTxPduId`. 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.

[20] If DET is active a corresponding error shall be set.
[21] It is not possible to signal that temporary resource lack to the upper layer because „PduR_FrTpStartOfReception“ provide no parameter fort hat case.

[22] Only the received StartFrame PDU shall be stored temporarry in a local buffer. This is necessray in case of a gateway has temporarry no free resources to process that frame. The correct protocol and timing behavour is ensured if FrTp sends a FlowControl PDU after a free channel was allocated.

[SWS_FrTp_01080] ⌈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.⌋()

Step 15

[SWS_FrTp_01079] ⌈The FrTp module shall extract the N-SDU data from the received N-PDU data according to ISO 10681-2⌋()

[SWS_FrTp_01138] ⌈The FrTp module shall initiate the copy process of the received N-SDU (fragment) by calling the service primitive PduR_FrTpCopyRxData[23].⌋()

[SWS_FrTp_00421] ⌈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 `FrTpRxSduId` to signal upper layers that an N-SDU has been received.

[SWS_FrTp_01081] ⌈The FrTp module shall free the allocated channel (`FrTpRxChannelState = Idle`) if
a) all N-SDU data are received and
b) all required response N-PDUs (FlowControl) are transmitted and TxConfirmation was given.⌋()

[SWS_FrTp_01083] ⌈The FrTp module shall always call the service primitive *`PduR_FrTpRxIndication`* after *`PduR_FrTpStartOfReception`* succeeded. The result shall be `E_OK` if
a) all N-SDU data are received and

[23] The procedure is also used for the “Routing-On-The-Fly” behaviour for gateways.

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

7.5.3.1 Receive Cancellation

[SWS_FrTp_01180] 「If development error detection is enabled:
The function FrTp_CancelReceive shall check the parameter FrTpRxSduId for being valid. If the check for FrTpRxSduId fails, the function shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.」()

[SWS_FrTp_01181] 「The FrTp shall abort the reception of the current N-SDU if the
service FrTp_CancelReceive provides a valid FrTpRxSduId.」()

[SWS_FrTp_01182] 「The FrTp shall reject the request for receive cancellation in case of an
a) unsegmented reception or
b) 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] 「If the FrTp_CancelReceive service has been successfully Executed, FrTp_CancelReceive shall return with result E_OK.」()

7.5.3.2 Receive with unknown message length

The FrTp according to ISO 10681-2 provides a method to receive data with unknown message length.

[SWS_FrTp_01184] 「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 szenario:
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 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 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.

7.5.4.1 Buffer Access Mode

[SWS_FrTp_01084] 「For Tx direction the FlexRay Transport Protocol Layer shall support
a) “Immediate Buffer Access” mode and
b) “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 depict in Figure 17 there is a direct mapping between N-PDUs and L-PDUs (done within FrIf module’s frame construction plan).

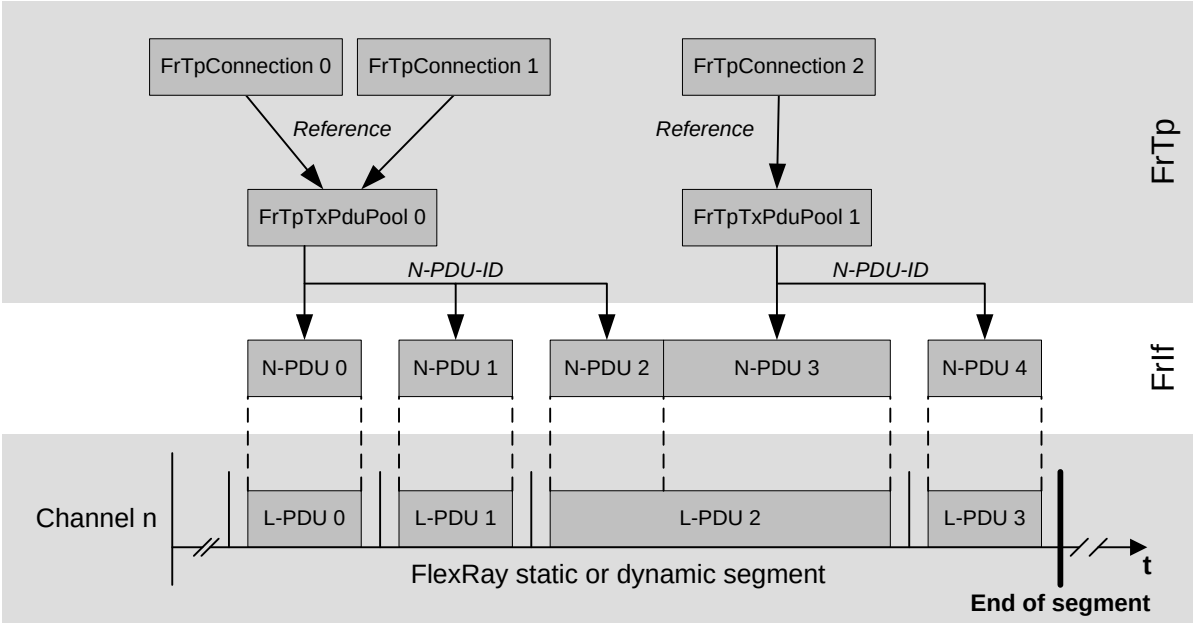


Figure 17: Mapping of N-PDUs to N-PDU-Pools

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 [24]. By supporting dynamic bandwidth assignment, a support of different communication scenarios is possible.
Figure 18 depicts two different scenarios which could be supported with only one FrTp configuration. From gateway's point of view different communication scenarios are possible:

a) single connection communication
   Complete bandwidth (slots) is assigned to one communication link (e.g. to ECU 2)
b) multiple connection communication
   Bandwidth (slots) is shared between different communication links to different ECUs (e.g. ECU 1-3).

[24] 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 parallel 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").

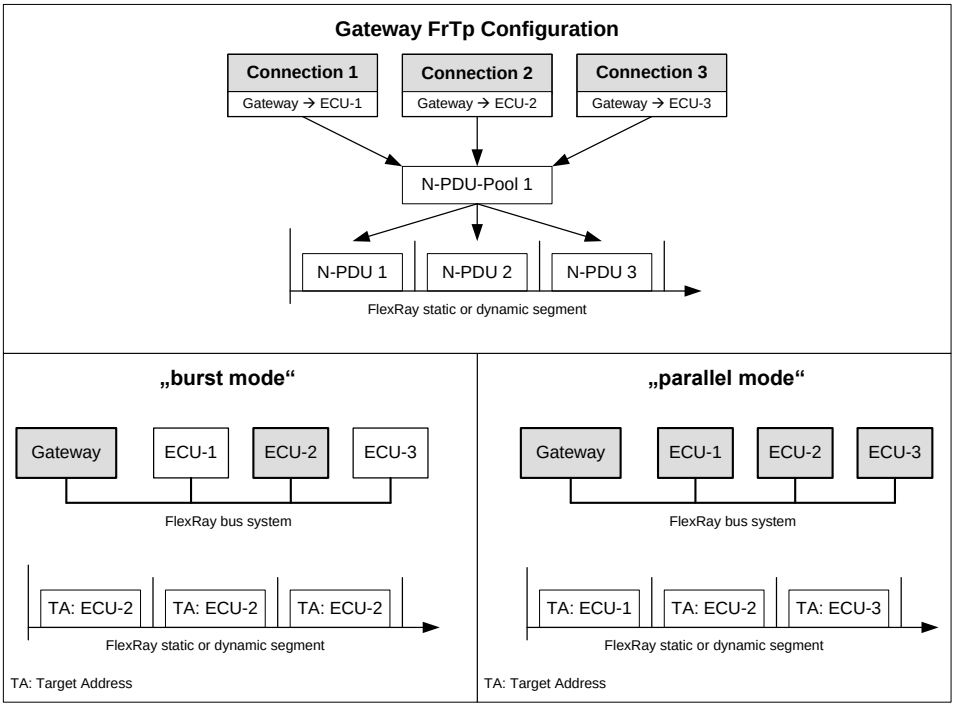


Figure 18: PDU-Pool sharing by different connections

The connection handler controls the partitioning of bandwidth (see Figure 19).

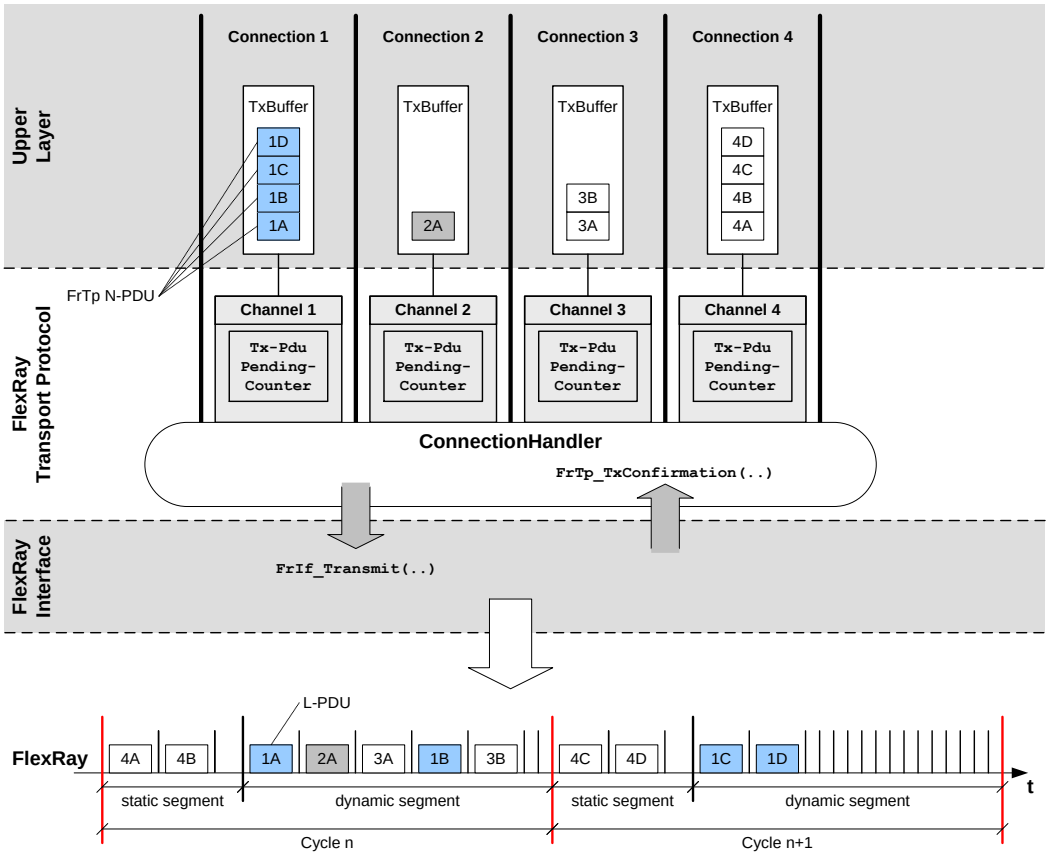


Figure 19: Connection Handler for different connections

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.[25]

[SWS_FrTp_01088] ⌈ 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[26].⌋()

Note: See also chapter 7.3.2.2

[SWS_FrTp_01089] ⌈Each FrTp Tx N-PDU within an `FrTpTxPduPool` could have an individual length.[27]⌋()

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] ⌈Each FrTpChannel shall implement a runtime variable `TxPduPendingCounter` (see chapter 7.3.3.2).⌋()

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

[SWS_FrTp_01092] ⌈ The `TxPduPendingCounter` shall be decremented each time the service primitive `FrTp_TxConfirmation` was called with the corresponding parameter `FrTpTxConfirmationPduId`.⌋()

[25] This could occure within the dynamic segment if the transfer of the last L-PDU (including a 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.

[26] 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.

[27] As depict in Figure 17, at the end of a segment 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.

[SWS_FrTp_01093] ⌈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] ⌈ 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] ⌈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] ⌈A communication handler task shall process all active `FrTpConnections` alternately[28] as long as free FrTp Tx N-PDUs are available within the referenced `FrTpTxPduPool`.⌋()

7.5.6 Transmit Cancellation

According to ISO 10681-2 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] ⌈The “Transmit Cancellation” feature shall be (de)activated by static configuration of the FrTp parameter *`FrTpTransmitCancellation`* (see section 10.2).⌋()

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

[28] Alternate means that a schedule has to be implemented which process all active FrTpConnections. It is recommandet to use a simple round-robin method but other schedules are also possible.

[SWS_FrTp_01116] ⌈ 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] ⌈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:

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

Figure 20 depicts the transmit cancellation behavior of an unsegmented data transfer.

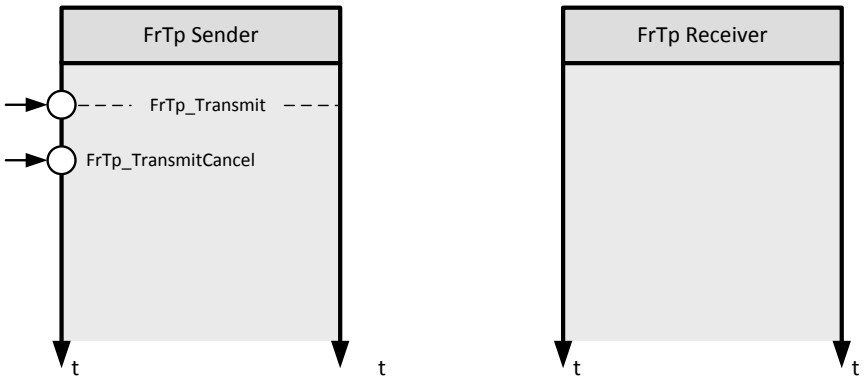


Figure 20: Transmit Cancellation at unsegmented data transfer

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:

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

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

c) 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 21 depicts the transmit cancellation behavior 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.

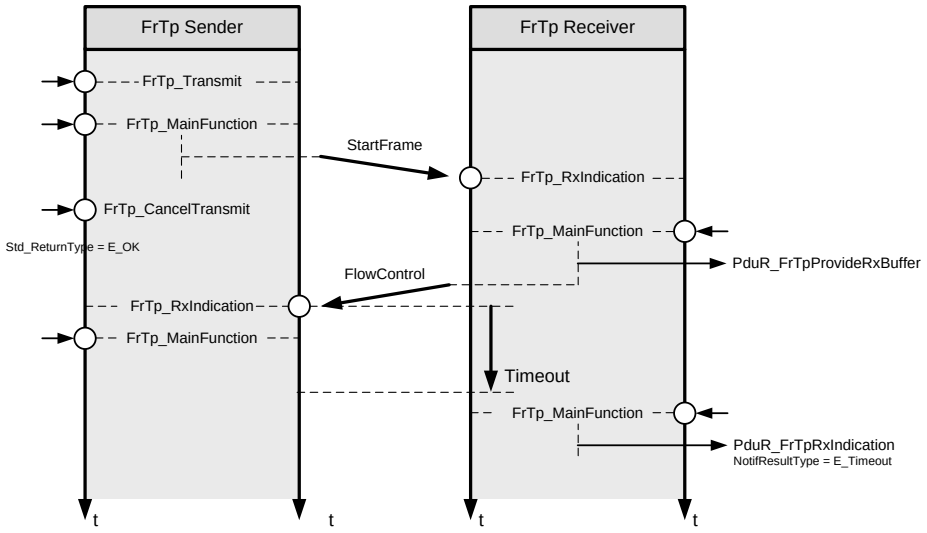


Figure 21: Transmit Cancellation at segmented data transfer

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 [29]. 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.

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

7.5.7 Change FrTp Parameter

[SWS_FrTp_00242] ⌈ The FrTp module shall change the ISO10681-2 FlowControl PDU parameter(s) of BandwidthControl (BC)
a) FrTpSCexp (please refer to ISO 10681-2)
b) FrTpMaxNbrOfNPduPerCycle (please refer to ISO 10681-2)
during runtime if the corresponding API service primitive *FrTp_ChangeParameter* is called.⌋()

[SWS_FrTp_01195] ⌈The layout of the BC parameter shall be identical to the layout in the FC(CTS) frame: The FrTpMaxNbrOfNPduPerCycle shall be placed in bits 0..2, the FrTpSCexp in the bits 3…7. The upper byte of the parameter is not used.⌋()

[SWS_FrTp_01115] ⌈A change parameter request during an ongoing reception shall be terminated with return value of E_NOT_OK.⌋()

[SWS_FrTp_01156] ⌈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 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] ⌈The FrTp module shall support the different timers and their start/stop conditions for communication handling as defined in Figure 22, Figure 23 and Table 2.⌋()

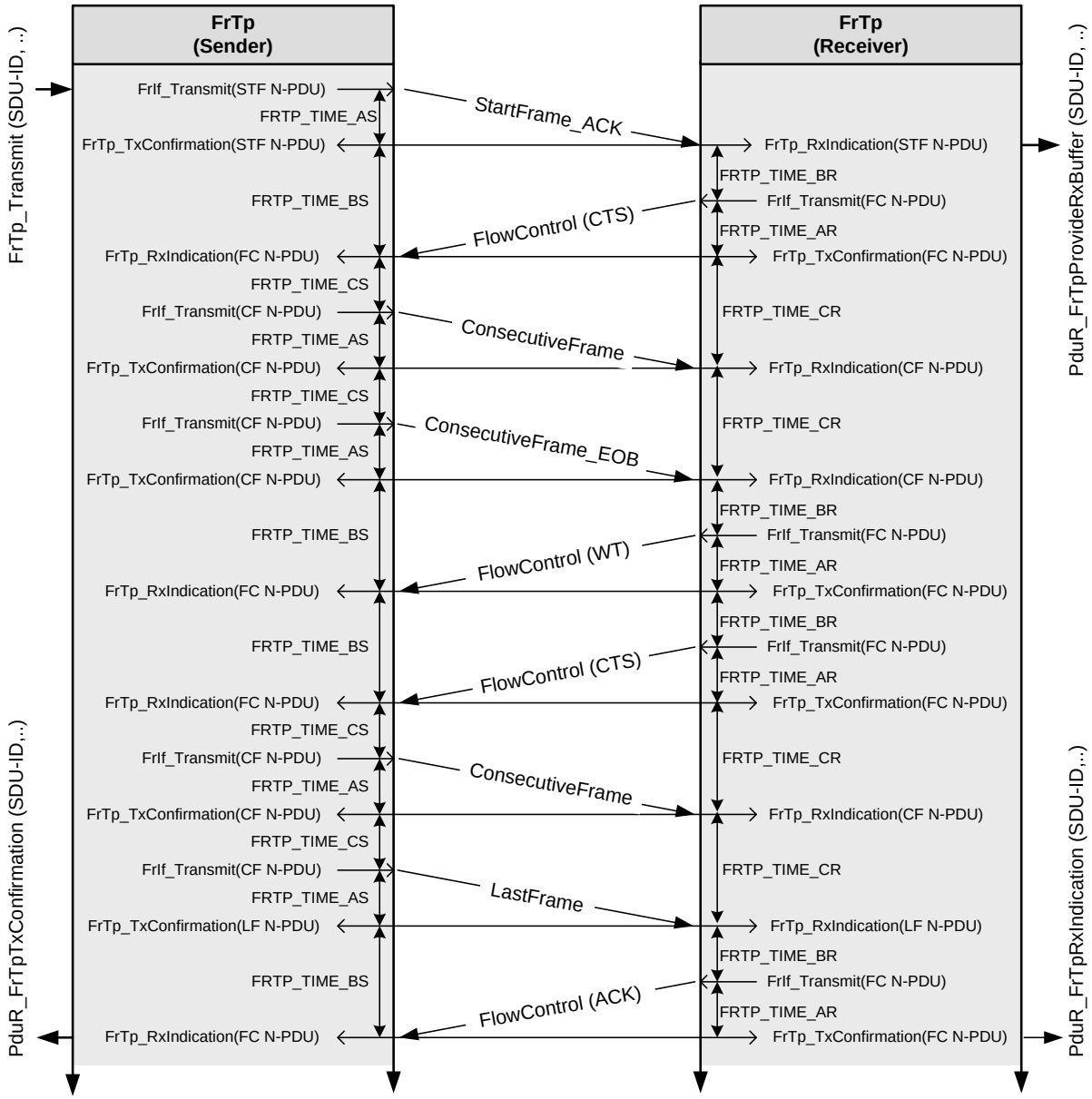


Figure 22: Timing parameter definition for one PDU transmission per Main-Function call

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 FrIf_Transmit 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 behavior (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 23.

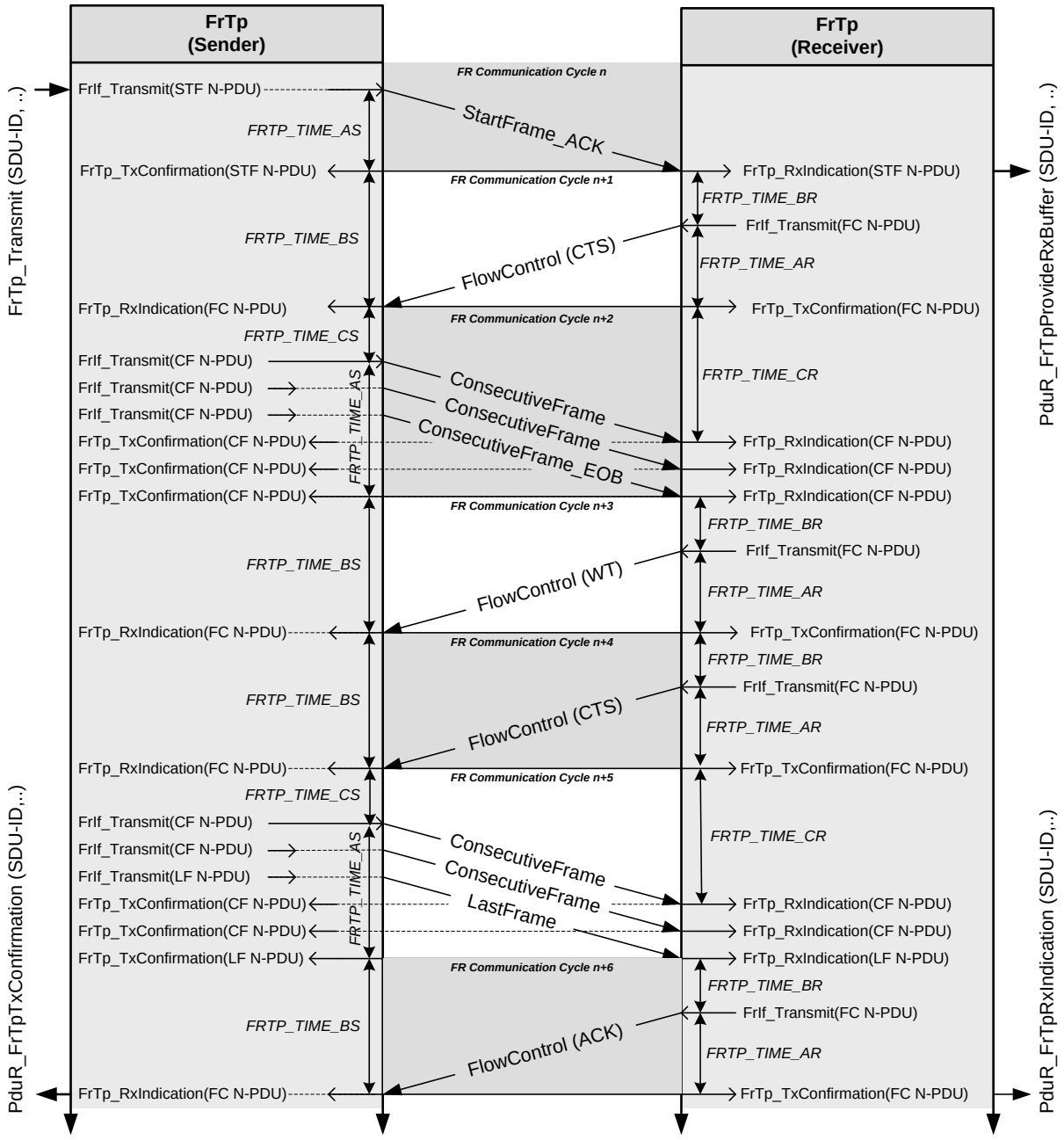


Figure 23: Timing parameter definition for multiple PDU transmission per Main-Function call

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_Main() and the corresponding FlowControl Parameters) For details please refer to chapter 7.3.3.3.

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.	FrIf_Transmit()	FrTp_TxConfirmation()
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.	FrIf_Transmit()	FrTp_TxConfirmation
FRTP_TIME_BS	Time until reception of the next FlowControl N-PDU. The maximum time is controlled by the FRTP_TIMEOUT_BS.	FrTp_TxConfirmation (STF), FrTp_RxIndication (FC), FrTp_TxConfirmation (CF), FrTp_TxConfirmation (LF)	FrTp_RxIndication (FC)
FRTP_TIME_BR	Time until transmission of the next FlowControl N-PDU.	FrTp_RxIndication (STF), FrTp_TxConfirmation (FC), FrTp_RxIndication (CF), FrTp_RxIndication (LF)	FrIf_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_TxConfirmation (FC), FrTp_RxIndication (CF)	FrTp_RxIndication (CF) FrTp_RxIndication (LF)
S/s … sender R/r … receiver			

Table 2: Timing parameter for the FrTp module

[SWS_FrTp_01100] ⌈ The FrTp module shall support the communication timeout behavior as defined in Table 3.⌋()

Timeout Parameter	Cause	Action
FRTP_TIMEOUT_AS	Any FrTp N-PDU not transmitted in time on the sender side.[30]	Abort message transmission.[31] a) Call FrIf_CancelTransmit() and free the FrTpTxPdu. b) issue PduR_FrTpTxConfirmation with the corresponding FrTpTxSduId and E_NOT_OK.
FRTP_TIMEOUT_AR	Any FrTp FC N-PDU not transmitted in time on the receiver side.[32]	Abort message reception and issue PduR_FrTpRxIndication with the corresponding FrTpRxSduId.
FRTP_TIMEOUT_BS	FlowControl N-PDU not received (lost, overwritten) on the sender side.	Abort message transmission and issue PduR_FrTpTxConfirmation with the corresponding FrTpTxSduId and E_NOT_OK.
FRTP_TIMEOUT_CR	ConsecutiveFrame or Last Frame N-PDU not received (lost, overwritten) on the receiver side.[33]	Abort message reception and issue PduR_FrTpRxIndication with the corresponding FrTpRxSduId and E_NOT_OK.

Table 3: Timeout behaviour for FrTp module

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

31 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 to 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 used in the meanwhile.

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

33 This could occur in case preceding FlowControl N-PDU not received (lost, overwritten) on overall sender side.

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.

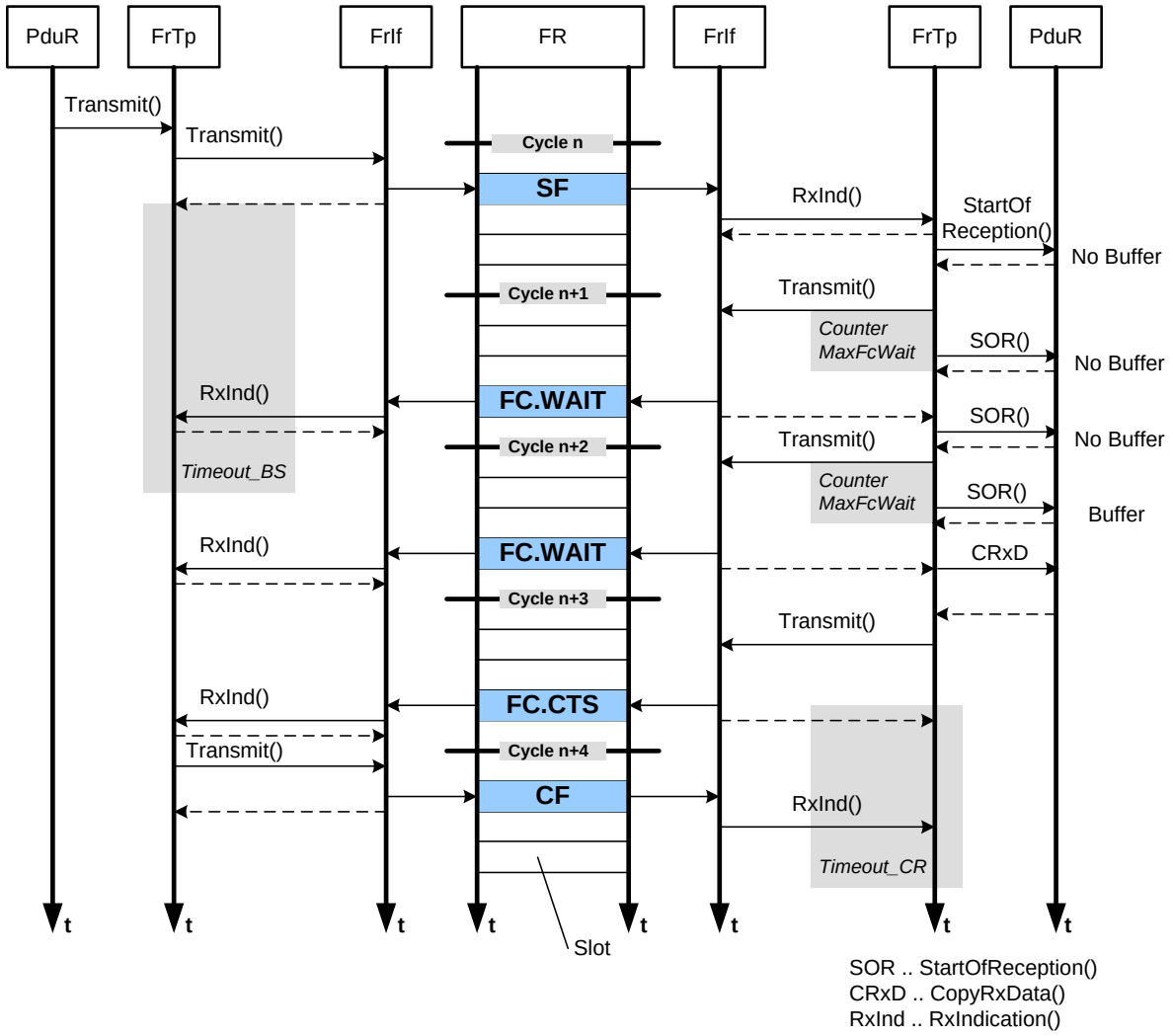


Figure 24: Counter, timer and function call interaction

[SWS_FrTp_01105] 「The FrTp module shall support the counters and corresponding limits as defined in Table 4: FrTp Counter.」()

[SWS_FrTp_01114] 「Each FrTp Counter shall be limited by a max value as defines in Table 4.」()

Counter Name	Counter Description	Limit
COUNTER_FCWT	On receiver site: Counts the number of transmittes FlowControl.Wait frames[34]	FrTpMaxFCWait
COUNTER_FRIF	Counts the attempts to send an Fr N-PDU via the service primitive *FrIf_Transmit*() in case this call returns E_NOT_OK	FrTpMaxFrIf
COUNTER_AR	Counts the attempts of the receiver to send an Fr N-PDU (FC, AF), in case a timeout AR occurs	FrTpMaxAr
COUNTER_AS	Counts the attempts to send an Fr N-PDU (SF, FF, LF) in case a timeout AS occurs	FrTpMaxAs
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

Table 4: FrTp Counter

[SWS_FrTp_01113] 「If a counter of has been reached, the FrTp module shall react as defined in Table 5.」()

Counter Name	Handling if counter has been reached
COUNTER_FCWT	Abort transmission
COUNTER_FRIF	Abort transmission
COUNTER_AR	Abort Reception
COUNTER_AS	Abort transmission
COUNTER_RX_RN	**Case a) Segmented – Acknowledged transmission** 1) Abort reception by calling service primitive PduR_FrTpRxIndication with E_NOT_OK 2) Send FlowControl.ABT (if aFlowControl is possible)

Table 5: FrTp module reaction if counters reached

[34] 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.

7.7 Error Handling

7.7.1 Error Detection

Note: If no production error is currently specified the requirement SWS_FrTp_00218 could be disregarded.

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

7.7.2 Error Notification

Note: If no production error is currently specified the requirement SWS_FrTp_01110 could be disregarded.

[SWS_FrTp_01108] ⌈The header file of the FrTp module, `FrTp.h`, shall provide a module Identifier `FRTP_MODULE_ID`.⌋()

[SWS_FrTp_01109] ⌈The module Identifier `FRTP_MODULE_ID` shall set to the value 0x24.⌋()

7.7.3 Error Classification

This section describes how the FrTp module has to manage the several error classes that may occur during the life cycle of this basic software.

According to the general requirements on basic software modules [3] all basic software modules must distinguish (according to the product life cycle) two error types:

- Development errors:
  These errors should be detected and fixed during development phase. In most cases, these errors are software errors. The detection of errors that should only occur during development can be switched off for production code (by static configuration, namely preprocessor switches).

- Production errors:
  These errors are hardware errors and software exceptions that cannot be avoided and are expected to occur in the production (i.e. series) code.

[SWS_FrTp_01111] ⌈The FrTp module shall support the error codes for development errors (Dev.) and production errors (Prod.) as defined in Table 6.⌋()

[SWS_FrTp_01132] ⌈All errors which are listed within Table 6 and marked as "*Dev.*" in the column *Relevance* are classified as development errors.⌋()

Type of error	Relevance	Related error code	Value [hex]
API service call without module initialization: Exception: a) FrTp_Init() b) FrTp_GetVersionInfo()	Dev.	FRTP_E_UNINIT	0x01
NULL-Pointer on any API call	Dev.	FRTP_E_NULL_PTR	0x02
API call with invalid SDU-ID (PduR) or PDU-ID (FrIf)	Dev.	FRTP_E_INVALID_PDU_SDU_ID	0x03
API call with invalid Parameter	Dev.	FRTP_E_INVALID_PARAMETER	0x04
Segmentation is required for a 1:n connection	Dev.	FRTP_E_SEG_ERROR	0x05
Transmission of unknown message length is detected but not configured.	Dev.	FRTP_E_UMSG_LENGTH_ERROR	0x06
No free channel available[35]	Dev.	FRTP_E_NO_CHANNEL	0x07
Dev. .. Development Prod. .. Production			

Table 6: Module error classification

[35] No free channel could occure for each ECU in principle, but especially for gateway configuration this error shall indicate architecture/configuration problems.

8 API specification

8.1 Imported types

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

[SWS_FrTp_00141] ⌈*Std_ReturnType* shall be imported from *Std_Types.h.*⌋()

[SWS_FrTp_01164] ⌈Std_VersionInfoType shall be imported from Std_Types.h.⌋()

[SWS_FrTp_01165] ⌈BufReq_ReturnType shall be imported from ComStack_Types.h.⌋()

[SWS_FrTp_01167] ⌈PduIdType shall be imported from ComStack_Types.h.⌋()

[SWS_FrTp_01168] ⌈PduInfoType shall be imported from ComStack_Types.h.⌋()

[SWS_FrTp_01169] ⌈PduLengthType shall be imported from ComStack_Types.h.⌋()

[SWS_FrTp_01170] ⌈RetryInfoType shall be imported from ComStack_Types.h.⌋()

[SWS_FrTp_01178] ⌈TPParameterType shall be imported from ComStack_Types.h.⌋()

Module	*Imported Type*
ComStack_Types	BufReq_ReturnType
	PduIdType
	PduInfoType
	PduLengthType
	RetryInfoType
	TPParameterType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

[SWS_FrTp_01133] ⌈The following FrTp specific types shall be defined in FrTp_Types.h.⌋(SRS_BSW_00305)

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**]⌈

Name:	`FrTp_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
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	

⌋()

[SWS_FrTp_01137] ⌈The type FrTp_ConfigType is an external data structure containing post-build-time configuration data of the FrTp module which shall be implemented in FrTp_PBcfg.c (see chapter 5.6.1). ⌋()

8.3 Function definitions

8.3.1 Standard functions

8.3.1.1 FrTp_GetVersionInfo

[SWS_FrTp_00215] ⌈

]

Service name:	FrTp_GetVersionInfo

Syntax:	`void FrTp_GetVersionInfo(`
	`  Std_VersionInfoType* versioninfo`
	`)`

Service ID[hex]:	0x27
Sync/Async:	Synchronous
Reentrancy:	Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	versioninfo | Pointer to where to store the version information of this module.
Return value:	None
Description:	Returns the version information.

」(SRS_BSW_00407)

[SWS_FrTp_00202] 「The function FrTp_GetVersionInfo shall return the version information of this module. The version information includes:
- Two bytes for the vendor ID
- One byte for the module ID
- Three bytes version number.

The numbering shall be vendor specific; it consists of:
- The major, the minor and the patch version number of the module. 」()

Note: The AUTOSAR specification version number is checked during compile time and therefore not required in this API.

Note: Please refer also to document: AUTOSAR_SWS_StandardTypes.pdf .

[SWS_FrTp_00498] 「The function FrTp_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: *FrTpVersionInfoApi*」()

[SWS_FrTp_01150] 「 If development error detection for the FrTp_GetVersionInfo is enabled: the function FrTp_GetVersionInfo shall check the parameter versioninfo for being valid. If the check for FrTpRxPduInfoPtr fails, the function FrTp_GetVersionInfo shall raise the development error FRTP_E_NULL_PTR and return E_NOT_OK.」()

8.3.2 Initialization and Shutdown

8.3.2.1 FrTp_Init

[SWS_FrTp_00147] 「

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).	

」(SRS_BSW_00101)

[SWS_FrTp_01151] 「If development error detection for the FrTp_Init is enabled: the function FrTp_Init shall check the parameter configPtr for being valid. If the check for configPtr fails, the function FrTp_Init shall raise the development error FRTP_E_NULL_PTR and return E_NOT_OK.」 ()

8.3.2.2 FrTp_Shutdown

[SWS_FrTp_00148] 「

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

」()

8.3.3 Normal Operation

8.3.3.1 FrTp_Transmit

[SWS_FrTp_00149].「

Service name:	FrTp_Transmit

Syntax:	`Std_ReturnType FrTp_Transmit(`
	`  PduIdType FrTpTxSduId,`
	`  const PduInfoType* FrTpTxSduInfoPtr`
	`)`

Service ID[hex]:	0x02	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrTpTxSduId	This parameter contains the FlexRay TP instance unique identifier of the FrTp N-SDU to be transmitted.
	FrTpTxSduInfoPtr	Tx N-SDU Information Structure which contains a) pointer to the FrTp Tx N-SDU b) the length of the FrTp Tx N-SDU
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The request has been accepted E_NOT_OK: The request has not been accepted, e. g. parameter check has failed or no FrTpChannel resource is free.
Description:	This service is utilized to request the transfer of data. It sets a flag for indicating that a transmit request is present. This function has to be called with FrTp's SDU-Id, i.e. the upper layer has to translate its own PDU-Id into the FrTp's SDU-ID for the corresponding message. Within the provided FrTpSduInfoPtr only SduLength is valid (no data)! If this function returns E_OK then there will arise an call of PduR_FrTpCopyTxData in order to get data for sending.	

」()

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

[SWS_FrTp_01139] 「If development error detection for the FrTp_Transmit is enabled: the function FrTp_Transmit shall check the parameter FrTpTxSduId for being valid. If the check for FrTpTxSduId fails, the function FrTp_Transmit shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.」()

[SWS_FrTp_01140] 「If development error detection for the FrTp_Transmit is enabled: the function FrTp_Transmit shall check the parameter FrTpTxSduInfoPtr for being valid. If the check for FrTpTxSduInfoPtr fails, the function FrTp_Transmit shall raise the development error FRTP_E_NULL_PTR and return E_NOT_OK.」()

8.3.3.2 FrTp_CancelTransmit

[SWS_FrTp_00150] 「

Service name:	FrTp_CancelTransmit

Syntax:	`Std_ReturnType FrTp_CancelTransmit(`
	`  PduIdType FrTpTxSduId`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTpTxSduId	This parameter contains the FlexRay TP instance unique identifier of the Fr N-SDU which transfer has to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Transmit cancellation request of the specified Fr N-SDU is accepted. E_NOT_OK: Transmit cancellation request of the specified Fr N-SDU is rejected for an unsegmented data transmission
Description:	This service primitive is used to cancel the transfer of pending Fr N-SDUs. The connection is identified by FrTpTxSduId. When the function returns, no transmission is in progress anymore with the given N-SDU identifier.	

⌋()

[SWS_FrTp_01141] ⌈ If development error detection for the FrTp_CancelTransmit is enabled: the function FrTp_CancelTransmit shall check the parameter FrTpTxSduId for being valid. If the check for FrTpTxSduId fails, the function FrTp_CancelTransmit shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK. ⌋()

8.3.3.3 FrTp_ChangeParameter:

[SWS_FrTp_00151] ⌈

Service name:	FrTp_ChangeParameter	

Syntax:	`Std_ReturnType FrTp_ChangeParameter(`
	`  PduIdType id,`
	`  TPParameterType parameter,`
	`  uint16 value`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	id	Identification of the I-PDU to which the parameter the request shall affect.
	parameter	The selected parameter that the request shall change. Only the parameter TP_BC is accepted by FrTp.
	value	The value that the request shall change to. Range: $0000 - $00FF
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: request is accepted E_NOT_OK: request is not accepted

Description:	Request to change transport protocol parameter BandwithControl.

⌋()

[SWS_FrTp_01143] ⌈ 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] ⌈ 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.⌋()

8.3.3.4 FrTp_CancelReceive

[SWS_FrTp_01172] ⌈

Service name:	FrTp_CancelReceive	
Syntax:	`Std_ReturnType FrTp_CancelReceive( PduIdType FrTpRxSduId )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTpRxSduId	SDU-Id of currently ongoing reception
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Reception was terminated sucessfully E_NOT_OK: Reception was not terminated.
Description:	By calling this API with the corresponding RxSduId, the currently ongoing data reception is terminated immediately. When function returns, no reception is in progress anymore with the given N-SDU identifier.	

⌋()

8.4 Call-back notifications

8.4.1 FrTp_TriggerTransmit

[SWS_FrTp_00154] ⌈

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.
	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU data shall be copied. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (inout):	None	
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 copy its data into the buffer provided by PduInfoPtr->SduDataPtr and update the length of the actual copied data in PduInfoPtr->SduLength.	

」()

[SWS_FrTp_01145] 「 If development error detection for the FrTp_TriggerTransmit is enabled: the function FrTp_TriggerTransmit shall check the parameter FrTpTxPduId for being valid. If the check for FrTpTxPduId fails, the function FrTp_TriggerTransmit shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK. 」()

[SWS_FrTp_01146] 「 If development error detection for the FrTp_TriggerTransmit is enabled: the function FrTp_TriggerTransmit shall check the parameter FrTpTxPduInfoPtr for being valid. If the check for FrTpTxPduInfoPtr fails, the function FrTp_TriggerTransmit shall raise the development error FRTP_E_NULL_PTR and return E_NOT_OK.」()

8.4.2 FrTp_RxIndication

[SWS_FrTp_00152] 「

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 I-PDU.

	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication interface module.	

」()

[SWS_FrTp_01147] 「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] 「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_NULL_PTR.」()

8.4.3 FrTp_TxConfirmation

[SWS_FrTp_00153] 「

Service name:	FrTp_TxConfirmation	

Syntax:	`void FrTp_TxConfirmation(`
	`  PduIdType TxPduId`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the I-PDU that has been transmitted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of an I-PDU.	

」()

[SWS_FrTp_01149] 「 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.」()

8.5 Scheduled functions

Basic Software Scheduler directly calls these functions.

8.5.1 FrTp_MainFunction

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

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

[SWS_FrTp_01152] ⌈The `FrTp_MainFunction` shall be called at least one time per FlexRay cycle.[36]⌋()

[SWS_FrTp_00162] ⌈ The `FrTp_MainFunction` shall follow the service primitive definition as described below:

Service name:	FrTp_MainFunction

Syntax:	`void FrTp_MainFunction(`
	`  void`
	`)`

Service ID[hex]:	0x10
Description:	Schedules the FlexRay TP. (Entry point for scheduling)

⌋()

8.6 Expected Interfaces

This chapter describes all expected APIs from other modules.

[36] The number of MainFunction calls depends on the global Flexray communication cycle length, the available reveive 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 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 asymetic the different segment lengths have to be considered.

8.6.1 Mandatory Interfaces

This chapter defines all mandatory interfaces (API service primitives), which are required in order t fulfill the core functionality of the FrTp module.

8.6.1.1 PDU Router Interface

[SWS_FrTp_00577] ⌈The FrTp module expects service primitives from the PDU Router as listed in Table 7.⌋()

API service primitive	Description
PduR_FrTpRxIndication	By this API service primitive, the FrTp indicates the completed (un)successful reception of a message.
PduR_FrTpStartOfReception	By this API service primitive, the FrTp indicates the start of a reception of N-SDU.
PduR_FrTpCopyRxData	By this API service primitive, the FrTp module indicates that the actual received N-SDU data shall be delivered to the receiver module (e.g. COM, DCM etc.).
PduR_FrTpCopyTxData	By this API service primitive, the FrTp module requests the actual sender module (e.g. COM, DCM etc.) of the Fr N-SDU to provide a transmit 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).

Table 7: Required PDU Router service primitives

8.6.1.2 FlexRay Interface

[SWS_FrTp_00578] ⌈The FrTp module expects service primitives from the FlexRay Interface as listed in Table 8.⌋()

API service primitive	Description
FrIf_Transmit	By this API service primitive, the FrTp module initiates a transmission of an N-PDU.
FrIf_CancelTransmit	By this API service primitive, the FrTp module could cancel a transmission of an N-PDU.

Table 8: Required FlexRay Interface service primitives

8.6.2 Optional Interfaces

8.6.2.1 Development Error Tracer

[SWS_FrTp_00579] 「Depending on the configuration parameter `FrTpDevErrorDetect`, the FrTp module expects service primitives from Development Error Tracer module as listed in Table 9」()

API service primitive	Description
Det_ReportError	By this API service primitive, development errors are reported.

Table 9: Required Development Error Tracer service primitives

8.6.3 Configurable 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.

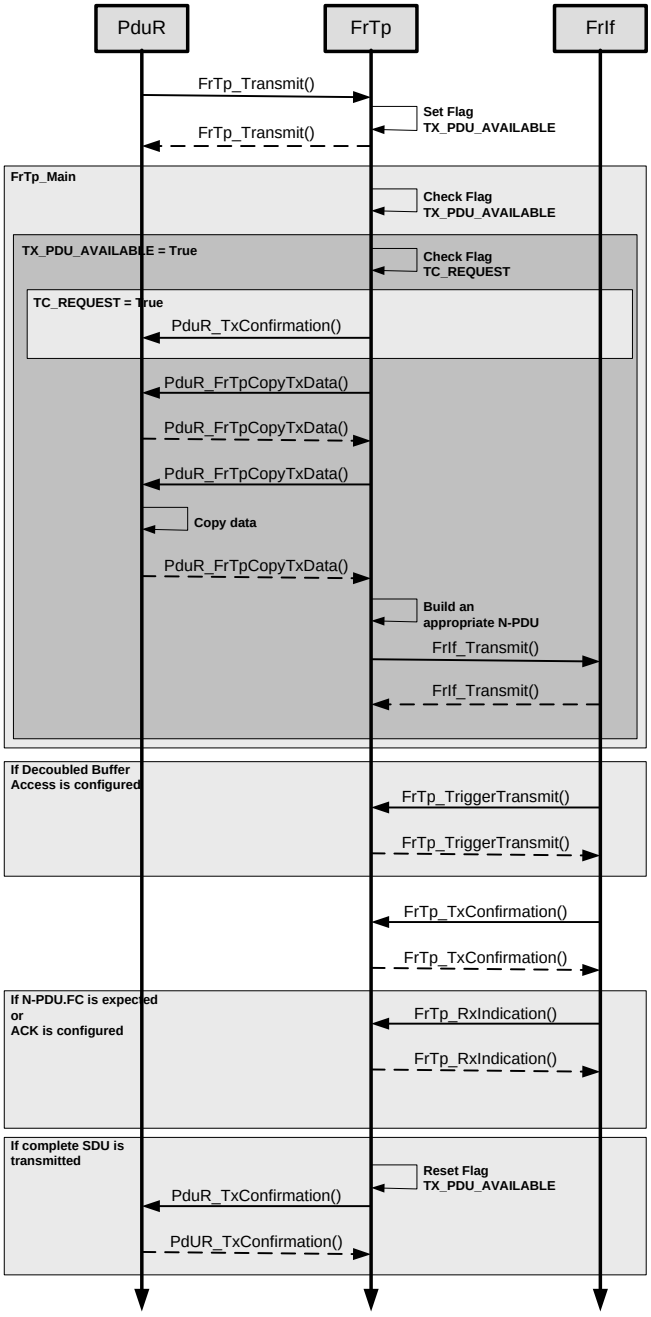


Figure 25: Sending of N-Pdus

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.

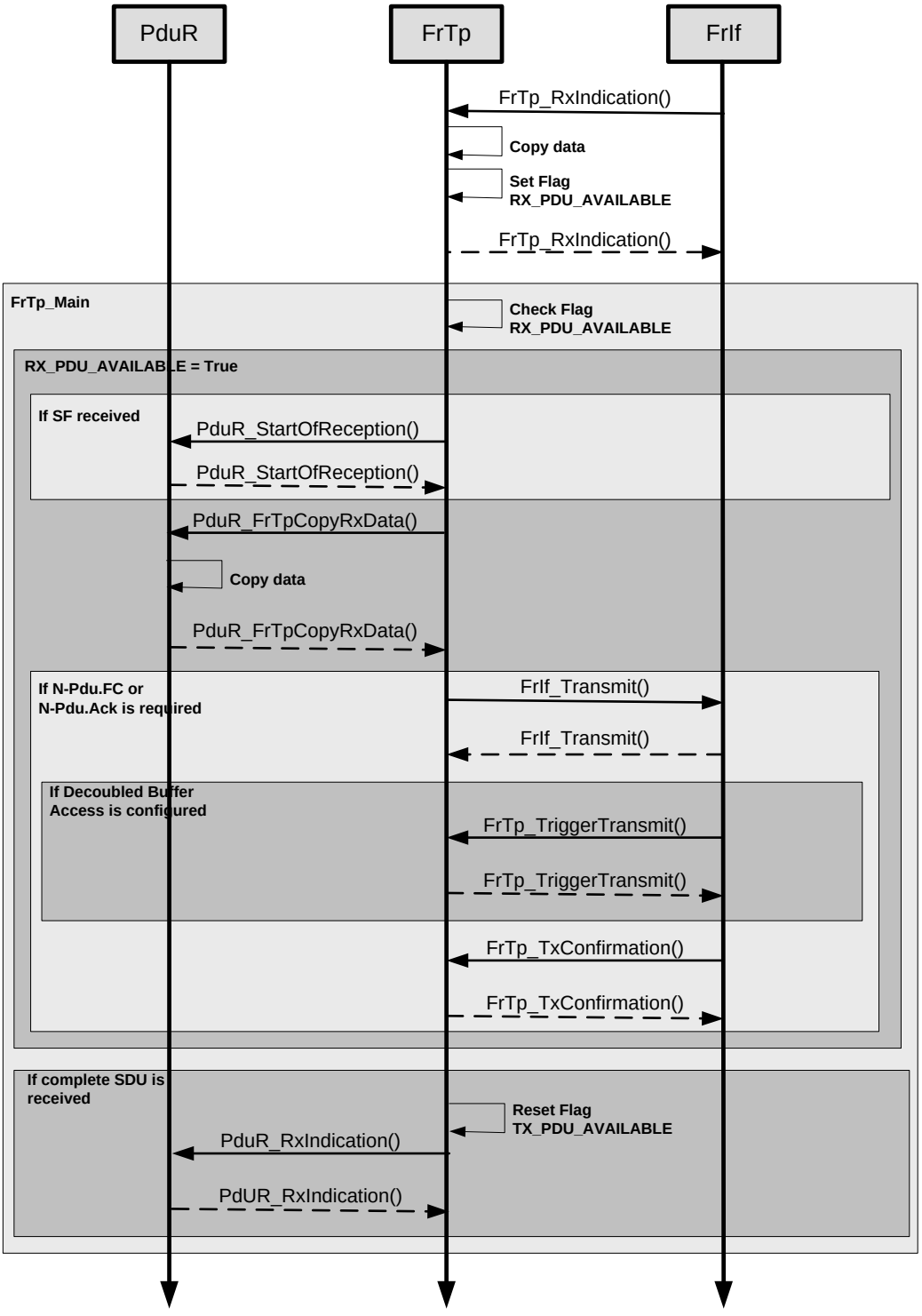


Figure 26: Receiving of N-Pdus (FrTp_MainFunction() shall call PduR_FrTpStartOfReception() routine)

10 Configuration specification

This chapter defines configuration parameters and their clustering into containers. In order to support the specification, Chapter 10.1 describes fundamentals.

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

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

[SWS_FrTp_00569] 「The configuration tool should extract all information to configure the FlexRay Transport Protocol.」()

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters for the FrTp module.

10.2.1 Variants

[SWS_FrTp_01131] 「The FrTp module shall support configuration variants as listed below:

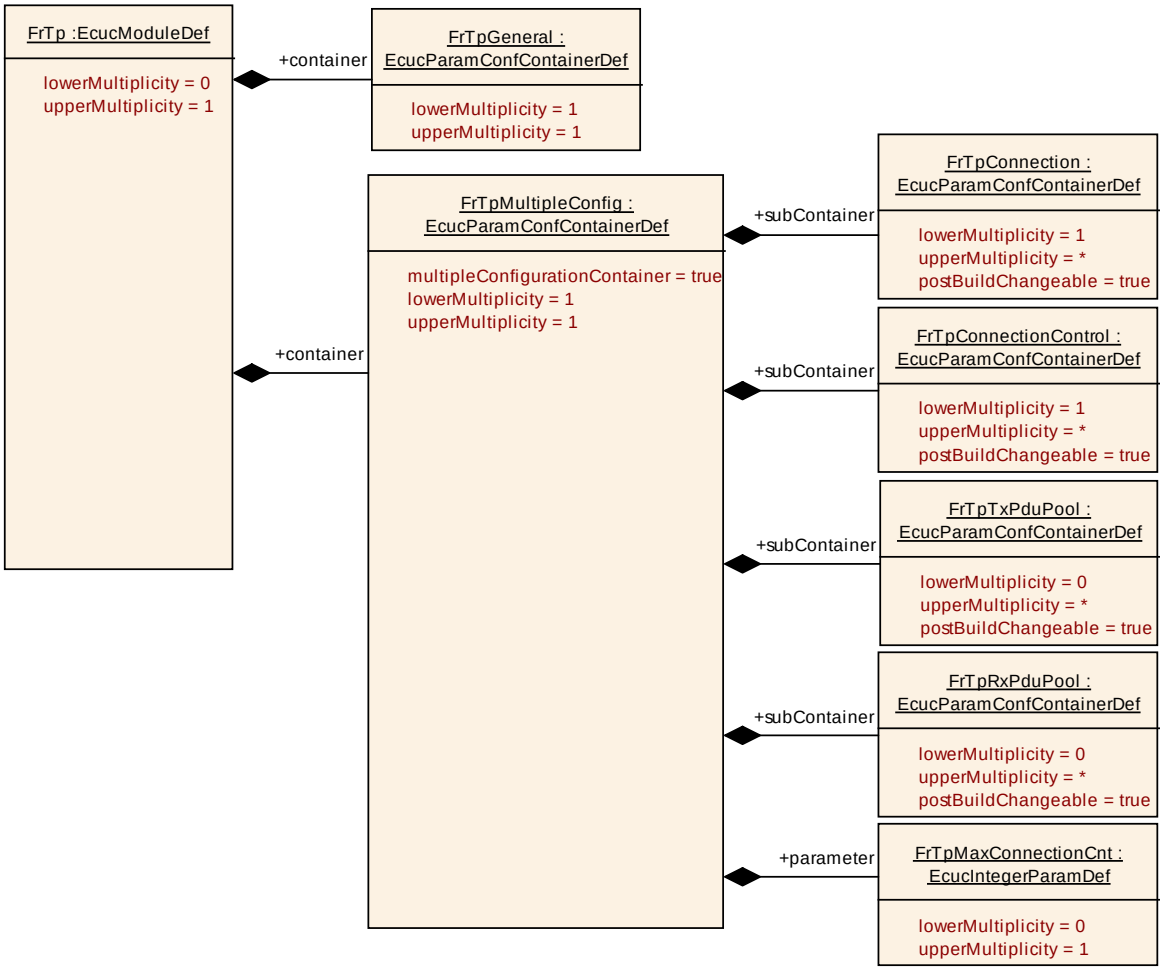
VARIANT-PRE-COMPILE	Only parameters with "Pre-compile time" configuration are allowed in this variant.
VARIANT-POST-BUILD	Parameters with "Pre-compile time" and "Post-build time" are allowed in this variant.

」()

10.2.2 FrTp

SWS Item	**ECUC_FrTp_00001 :**
Module Name	*FrTp*
Module Description	Configuration of the FlexRay Transport Protocol module according to ISO 10681-2.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrTpGeneral	1	This container contains the general configuration parameters of the FlexRay Transport Protocol module.
FrTpMultipleConfig	1	This container holds one or several multiple configuration sets.



10.2.3 FrTpGeneral

SWS Item	**ECUC_FrTp_00009 :**
Container Name	FrTpGeneral
Description	This container contains the general configuration parameters of the FlexRay Transport Protocol module.
Configuration Parameters	

SWS Item	**ECUC_FrTp_00002 :**		
Name	FrTpAckRt {FRTP_HAVE_ACKRT}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00004 :**
Name	FrTpChanNum {FRTP_CHAN_NUM}
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00052 :**		
Name	FrTpChangeParamApi {FRTP_CHANGE_PARAMETER_API}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00008 :**		
Name	FrTpDevErrorDetect {FRTP_DEV_ERROR_DETECT}		
Description	Preprocessor switch for enabling development error detection. True: Development Error Detection is enabled False: Development Error Detection is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00051 :**		
Name	FrTpFullDuplexEnable {FRTP_FULL_DUPLEX_ENABLE}		
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		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00011 :**	
Name	FrTpMainFuncCycle {FRTP_MAINFUNC_CYCLE}	
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	--	

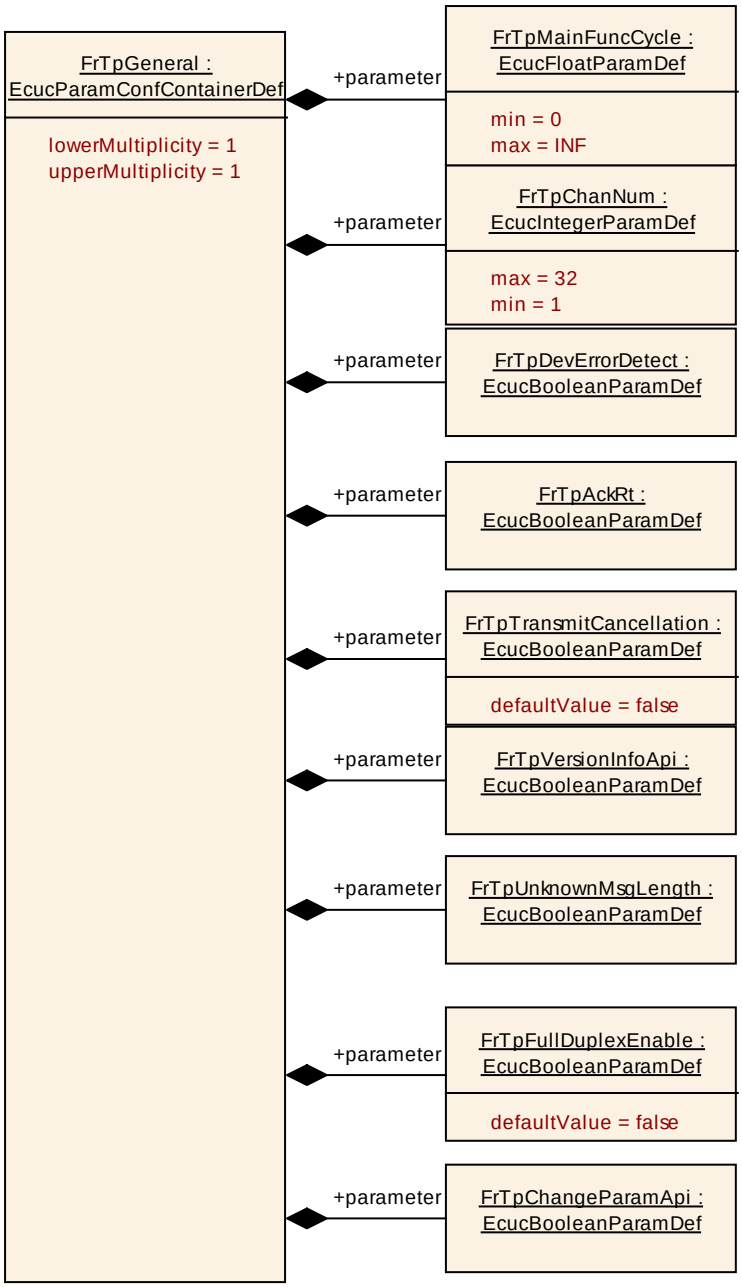
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00036 :**		
Name	FrTpTransmitCancellation {FRTP_HAVE_TC}		
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		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00044 :**		
Name	FrTpUnknownMsgLength		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00045 :**		
Name	FrTpVersionInfoApi {FRTP_VERSION_INFO_API}		
Description	Preprocessor switch for enabling the Version info API. True: Version Info API is enabled False: Version Info API is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers



10.2.4 FrTpMultipleConfig

SWS Item	**ECUC_FrTp_00018 :**
Container Name	FrTpMultipleConfig [Multi Config Container]
Description	This container holds one or several multiple configuration sets.
Configuration Parameters	

SWS Item	**ECUC_FrTp_00054 :**	
Name	FrTpMaxConnectionCnt {FRTP_MAX_CONNECTION_CNT}	
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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.

10.2.5 FrTpConnection

SWS Item	**ECUC_FrTp_00006 :**
Container Name	FrTpConnection{FrTpConnectionConfiguration}
Description	This container contains the connection specific parameters to transfer N-PDUs via FlexRay TP. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_FrTp_00050 :**		
Name	FrTpBandwidthLimitation {FRTP_BW_LIMIT}		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00010 :**		
Name	FrTpLa {FRTP_LA}		
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.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00019 :**		
Name	FrTpMultipleReceiverCon		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00021 :**		
Name	FrTpRa {FRTP_RA}		
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.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

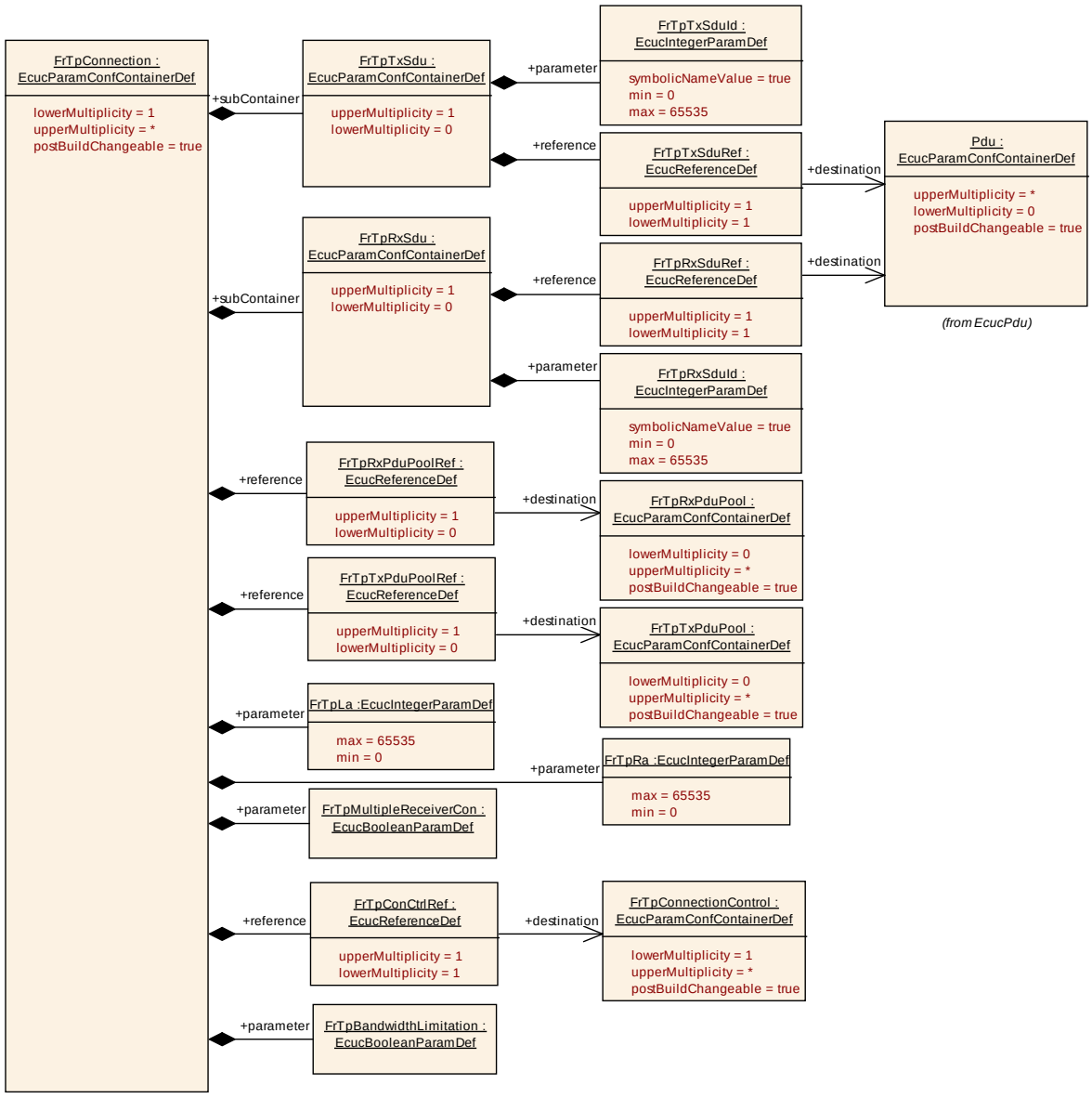
SWS Item	**ECUC_FrTp_00005 :**		
Name	FrTpConCtrlRef {FRTP_CON_CTRL_REF}		
Description	FrTpConnectionControlReference: This parameter defines a reference to a connection control container.		
Multiplicity	1		
Type	Reference to [FrTpConnectionControl]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00025 :**		
Name	FrTpRxPduPoolRef {FRTP_RX_PDU_POOL_REF}		
Description	This parameter defines a reference to a RxPduPool.		
Multiplicity	0..1		
Type	Reference to [FrTpRxPduPool]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00039 :**
Name	FrTpTxPduPoolRef {FRTP_TX_PDU_POOL_REF}
Description	This parameter defines a reference to a TxPduPool.
Multiplicity	0..1
Type	Reference to [FrTpTxPduPool]

ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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.
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.



10.2.6 FrTpTxSdu

SWS Item	**ECUC_FrTp_00041 :**
Container Name	FrTpTxSdu
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.
Configuration Parameters	

SWS Item	**ECUC_FrTp_00042 :**		
Name	FrTpTxSduId {FRTP_SDUID}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00043 :**		
Name	FrTpTxSduRef		
Description	Reference to a PDU in the global PDU structure.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.7 FrTpRxSdu

SWS Item	**ECUC_FrTp_00027 :**
Container Name	FrTpRxSdu
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.
Configuration Parameters	

SWS Item	**ECUC_FrTp_00053 :**		
Name	FrTpRxSduId {FRTP_SDUID}		
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	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00028 :**
Name	FrTpRxSduRef
Description	Reference to a PDU in the global PDU structure.

Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.2.8 FrTpConnectionControl

SWS Item	**ECUC_FrTp_00007 :**
Container Name	FrTpConnectionControl{FrTpConnectionControl}
Description	This container contains the configuration parameters to control a FlexRay TP connection. **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_FrTp_00003 :**		
Name	FrTpAckType {FRTP_ACKTYPE}		
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
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00012 :**		
Name	FrTpMaxAr {FRTP_MAX_AR}		
Description	This parameter defines the maximum number of trying to send a frame when a TIMEOUT AR occurs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00013 :**		
Name	FrTpMaxAs {FRTP_MAX_AS}		
Description	This parameter defines the maximum number of trying to send a frame when a TIMEOUT AS occurs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE

	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00055 :**		
Name	FrTpMaxBufReq {FRARTP_MAX_BUFREQ}		
Description	This parameter is used to limit the number of retries for PduR_FrTpCopyTxData when no timer is active.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00015 :**		
Name	FrTpMaxBufferSize {FRTP_MAXBFS}		
Description	Limits the maximal buffer size the FrTp can choose in order to limit the amount of Tx buffer that will be requested at the sender side in a segmented transfer.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00014 :**		
Name	FrTpMaxFCWait {FRTP_MAX_FCWAIT}		
Description	This parameter defines the maximum number of FlowControl N-PDUs with FlowState "WAIT"		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00016 :**		
Name	FrTpMaxFrIf {FRTP_MAX_FRIF}		
Description	This parameter defines the maximum number of trying to send a frame when the FrIf returns an error.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00029 :**		
Name	FrTpMaxNbrOfNPduPerCycle {FRTP_MAX_NUMBER_OF_NPDU_PER_CYCLE}		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00017 :**		
Name	FrTpMaxRn {FRTP_MAX_RN}		
Description	This parameter defines the maximum number of retries (if retry is configured).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00020 :**		
Name	FrTpSCexp {FRTP_SEPARATION_CYCLE_EXPONENT}		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00047 :**		
Name	FrTpTimeBr {FRTP_TIME_BR}		
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	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: local

SWS Item	**ECUC_FrTp_00030 : (Obsolete)**		
Name	FrTpTimeBuffer {FRTP_TIME_BUFFER}		
Description	This parameter is deprecated and will be removed in the future. Old description: This parameter defines the time in seconds of waiting for the next try to get a Tx or Rx buffer. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.1.1		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00031 :**		
Name	FrTpTimeFrIf {FRTP_TIME_FRIF}		
Description	This parameter defines the time in seconds of waiting for the next try (if retry is activated) to send via FrIf_Transmit.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 0.255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00032 :**		
Name	FrTpTimeoutAr {FRTP_TIMEOUT_AR}		
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).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local 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).		

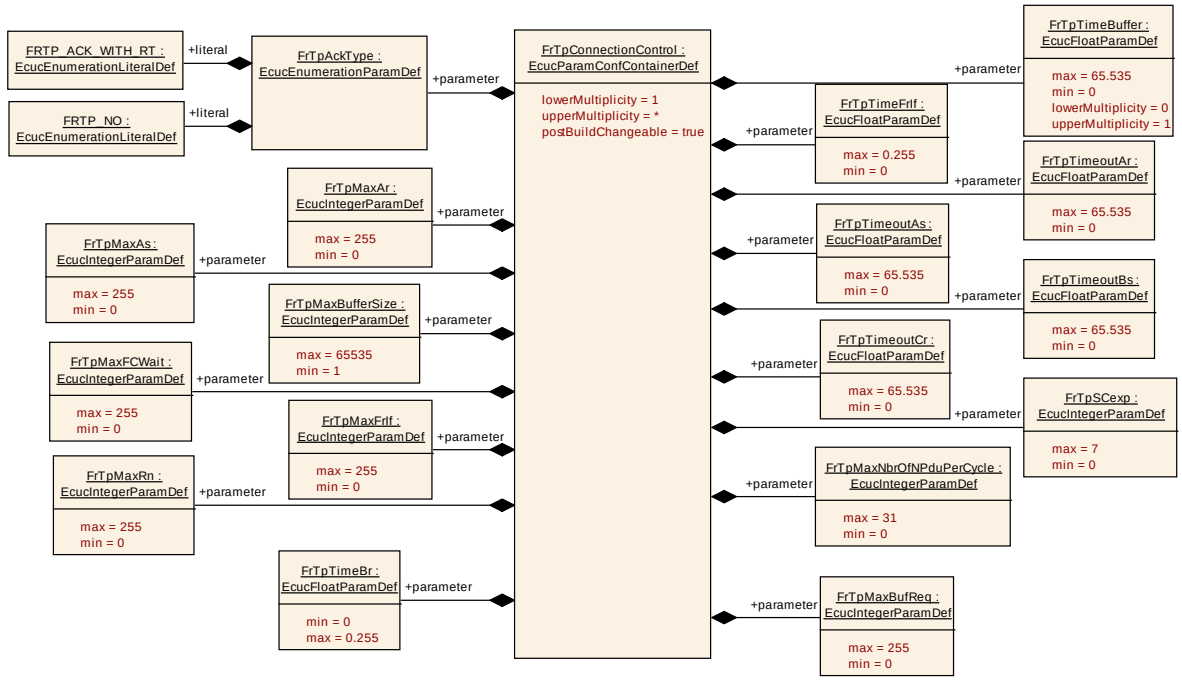
SWS Item	**ECUC_FrTp_00033 :**
Name	FrTpTimeoutAs {FRTP_TIMEOUT_AS}
Description	This parameter specifies the timeout in seconds the FrIf shall confirm a transmitted Pdu to the FrTp.
Multiplicity	1
Type	EcucFloatParamDef

Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local 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).		

SWS Item	**ECUC_FrTp_00034 :**		
Name	FrTpTimeoutBs {FRTP_TIMEOUT_BS}		
Description	This parameter defines the timeout in seconds for waiting for an FC or AF on the sender side in a 1:1 connection.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local 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).		

SWS Item	**ECUC_FrTp_00035 :**		
Name	FrTpTimeoutCr {FRTP_TIMEOUT_CR}		
Description	This parameter defines the timeout value in seconds a receiver is waiting for a CF or a LF.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local 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).		

No Included Containers



10.2.9 FrTpTxPduPool

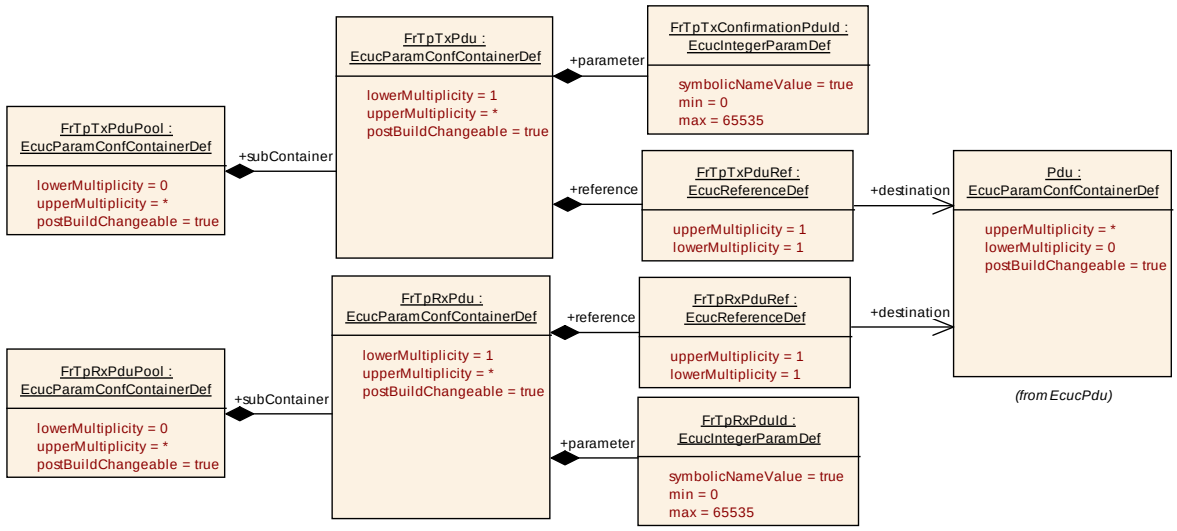
SWS Item	**ECUC_FrTp_00038 :**
Container Name	FrTpTxPduPool{PduPoolContainer}
Description	This container contains all Pdus that are assigned to that Pdu Pool. **Attributes:** postBuildChangeable=true
Configuration Parameters	

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

10.2.10 FrTpRxPduPool

SWS Item	**ECUC_FrTp_00024 :**
Container Name	FrTpRxPduPool{PduPoolContainer}
Description	This container contains all Pdus that are assigned to that Pdu Pool. **Attributes:** postBuildChangeable=true
Configuration Parameters	

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



10.2.11 FrTpTxPdu

SWS Item	**ECUC_FrTp_00037 :**
Container Name	FrTpTxPdu
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType **Attributes:** postBuildChangeable=true
Configuration Parameters	

SWS Item	**ECUC_FrTp_00049 :**		
Name	FrTpTxConfirmationPduId		
Description	Handle Id to be used by the FrIf to confirm the transmission of the FrTpTxPdu to the FrIf module (FrTp_TxConfirmation) and for TriggerTransmit (FrTp_TriggerTransmit).		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00040 :**		
Name	FrTpTxPduRef		
Description	Reference to a PDU in the global PDU structure.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.12 FrTpRxPdu

SWS Item	**ECUC_FrTp_00022 :**		
Container Name	FrTpRxPdu		
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType **Attributes:** postBuildChangeable=true		
Configuration Parameters			

SWS Item	**ECUC_FrTp_00023 :**		
Name	FrTpRxPduId		
Description	This is a unique identifier for a received message which is forwarded from the FrIf to the FrTp. ImplementationType: PduIdType		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTp_00026 :**		
Name	FrTpRxPduRef		
Description	Reference to a PDU in the global PDU structure.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

No Included Containers

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

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. This 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 27 the FrTp module has basically two different retry behaviours:

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

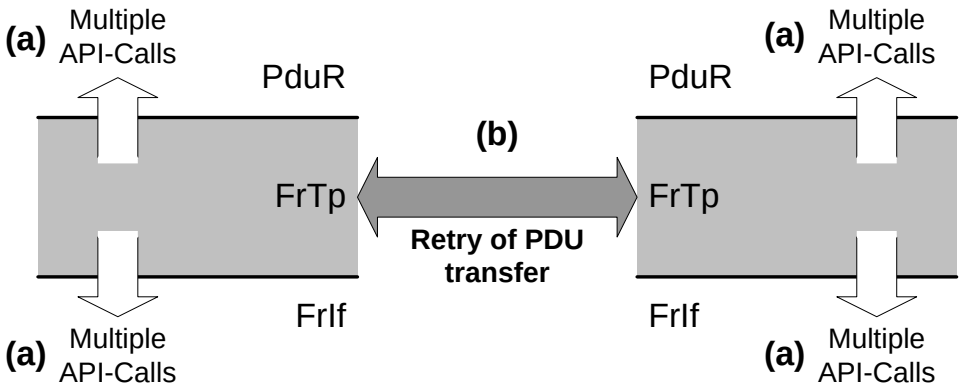


Figure 27: FrTp Retry Scenarios

Only for case (b) Retry of PDU transfer a global switch for enabling and disabling is defined (FRTP_HAVE_ACKRT). All other cases depend on the configuration of the different counters for the API calls:
FRTP_MAX_BUFREQ, 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] ⌈If configuration parameter *FrTpMultipleReceiverCon* is set for a connection, no Acknowledgement and retry shall be supported irrespective whether the configuration parameter *FRTP_HAVE_ACKRT* 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.

→ *Hinweise Configurationshinweis*

[SWS_FrTp_01154] ⌈For timeout As configuration it shall be considered that *FRTP_TIMEOUT_AS > FRTP_TIME_AS* ⌋()

[SWS_FrTp_01155] ⌈For timeout Ar configuration it shall be considered that *FRTP_TIMEOUT_AR > FRTP_TIME_AR* ⌋()

[SWS_FrTp_00599] ⌈For timeout Bs configuration it shall be considered that *FRTP_TIMEOUT_BS > FRTP_TIME_BR + FRTP_TIME_AR* ⌋()

[SWS_FrTp_01153] ⌈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 ISO 10681-2 [16]

[SWS_FrTp_00180] ⌈ The FrTp configuration shall ensure that $2^{SeparationCycleExponent}-1 \times t_{CycleTime} \leq$ FRTP_TIMEOUT_CR .[37] ⌋()

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[38]. In that case the bandwidth has to be limited by the receiver. There are three possibilities to limit the bandwidth for a connection link:

[37] This is to prevent a timeouts. Please refer to ISO 10681-2.
[38] 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.

a) Limit Bandwidth by Parameter FlowControl.BandwidthControl.MaxPduPerCycle in combination with Hardware FIFO buffer mechanisms.[39]
b) Limit Bandwidth by Parameter FlowControl.BandwidthControl.MaxPduPerCycle to reduce the number of allowed PDUs of the currently selected PDU-Pool.
c) 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” ISO 10681-2 behaviour with focus on FlexRay 3.0 Hardware FIFO buffer mechanisms. The figure below shows the dependencies.

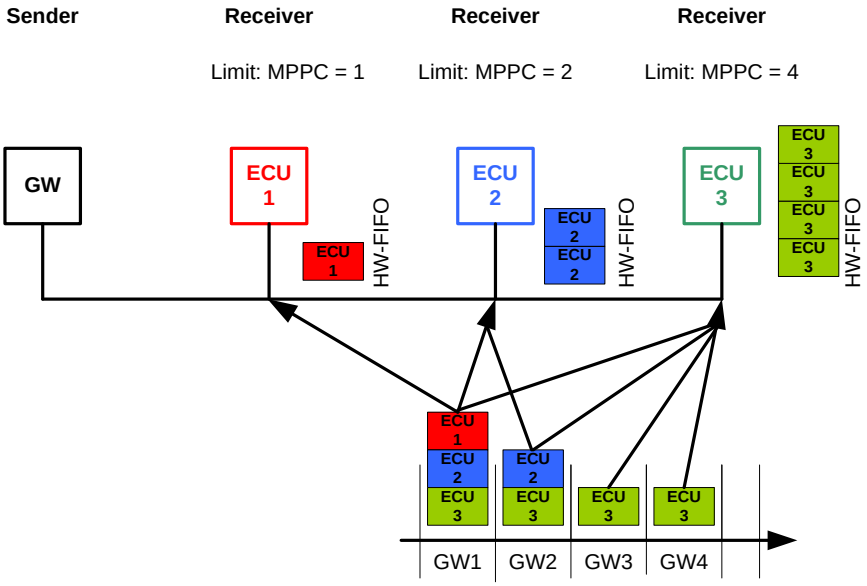


Figure 28: BandwidthControl by FlowControl Parameters in combination with HW FIFO buffer

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.

[39] This is an esential mechanism of FlexRay 3.0 protocol.

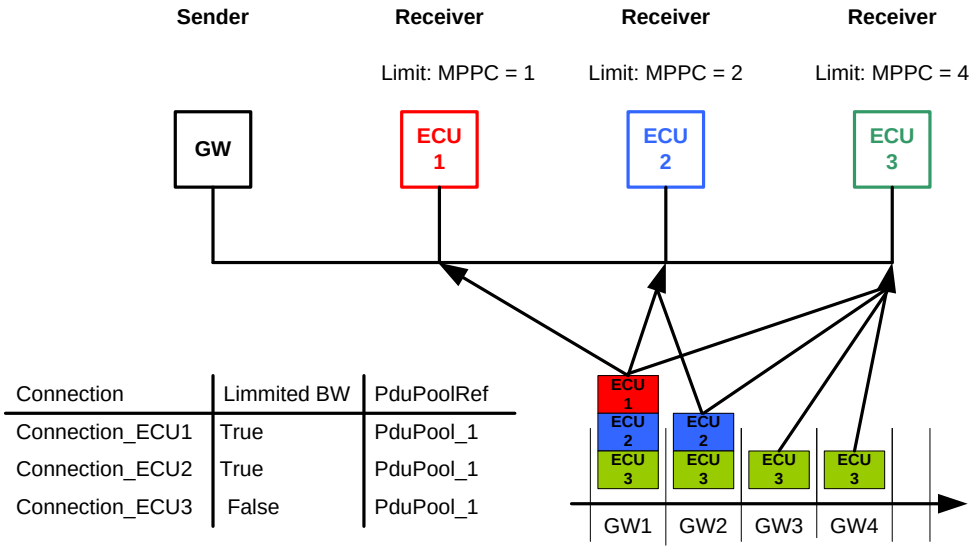


Figure 29: BandwidthControl by FlowControl Parameters

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] ⌈ 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] ⌈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.⌋()

[SWS_FrTp_01175] ⌈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.⌋()

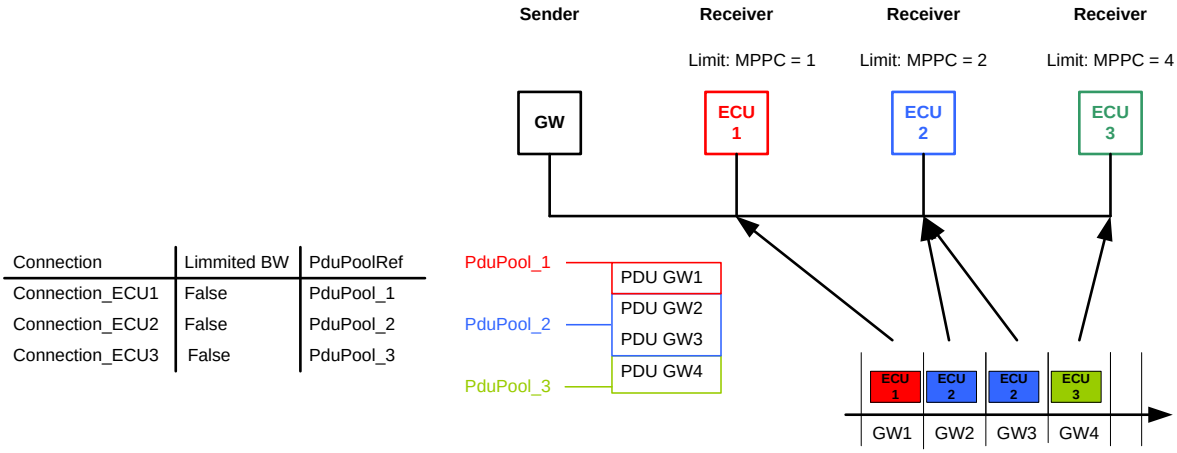
[SWS_FrTp_01177] ⌈If the attribute “FrTpBandwidthLimitation“ is set to "TRUE", a Rx-connection shall use the first Pdu of the referenced Tx-Pdu-Pool for sending the required FlowControl frame to continue a communication link.⌋()

10.4.4.3 BandwidthControl via different PDU Pools

If BandwidthControl is realized by different PDU Pools two different szenarios 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:

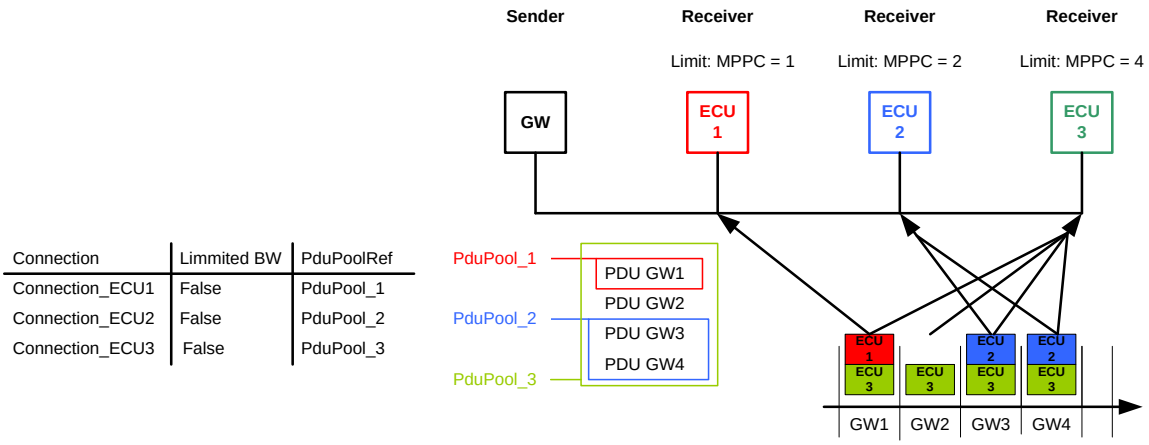


Connection	Limmited BW	PduPoolRef
Connection_ECU1	False	PduPool_1
Connection_ECU2	False	PduPool_2
Connection_ECU3	False	PduPool_3

Figure 30: 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:



Connection	Limmited BW	PduPoolRef
Connection_ECU1	False	PduPool_1
Connection_ECU2	False	PduPool_2
Connection_ECU3	False	PduPool_3

Figure 31: 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[40].

[SWS_FrTp_01176] 「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.

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.

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

11 Not applicable requirements

[SWS_FrTp_09999]「These requirements are not applicable to this specification.」
(SRS_BSW_00306, SRS_BSW_00312, SRS_BSW_00314, SRS_BSW_00323, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00328, SRS_BSW_00329, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00333, SRS_BSW_00335, SRS_BSW_00341, SRS_BSW_00343, SRS_BSW_00345, SRS_BSW_00347, SRS_BSW_00350, SRS_BSW_00358, SRS_BSW_00370, SRS_BSW_00371, SRS_BSW_00373, SRS_BSW_00375, SRS_BSW_00376, SRS_BSW_00377, SRS_BSW_00386, SRS_BSW_00387, SRS_BSW_00401, SRS_BSW_00405, SRS_BSW_00409, SRS_BSW_00410, SRS_BSW_00413, SRS_BSW_00414, SRS_BSW_00415, SRS_BSW_00417, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00159, SRS_BSW_00160, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00167, SRS_BSW_00168, SRS_BSW_00172, BSW05082)