Document Title	Specification of FlexRay ISO Transport Layer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	589
Document Classification	Standard

Document Version	4.0.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
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.

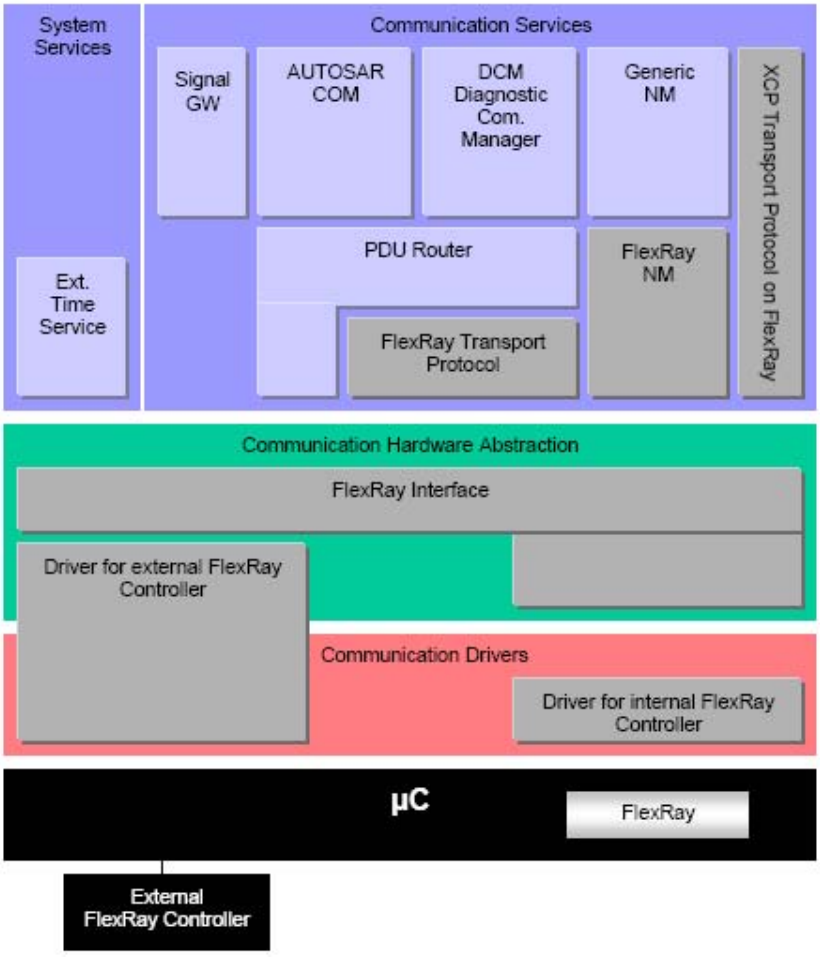


Figure 1: AUTOSAR FlexRay Layered Architecture

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).

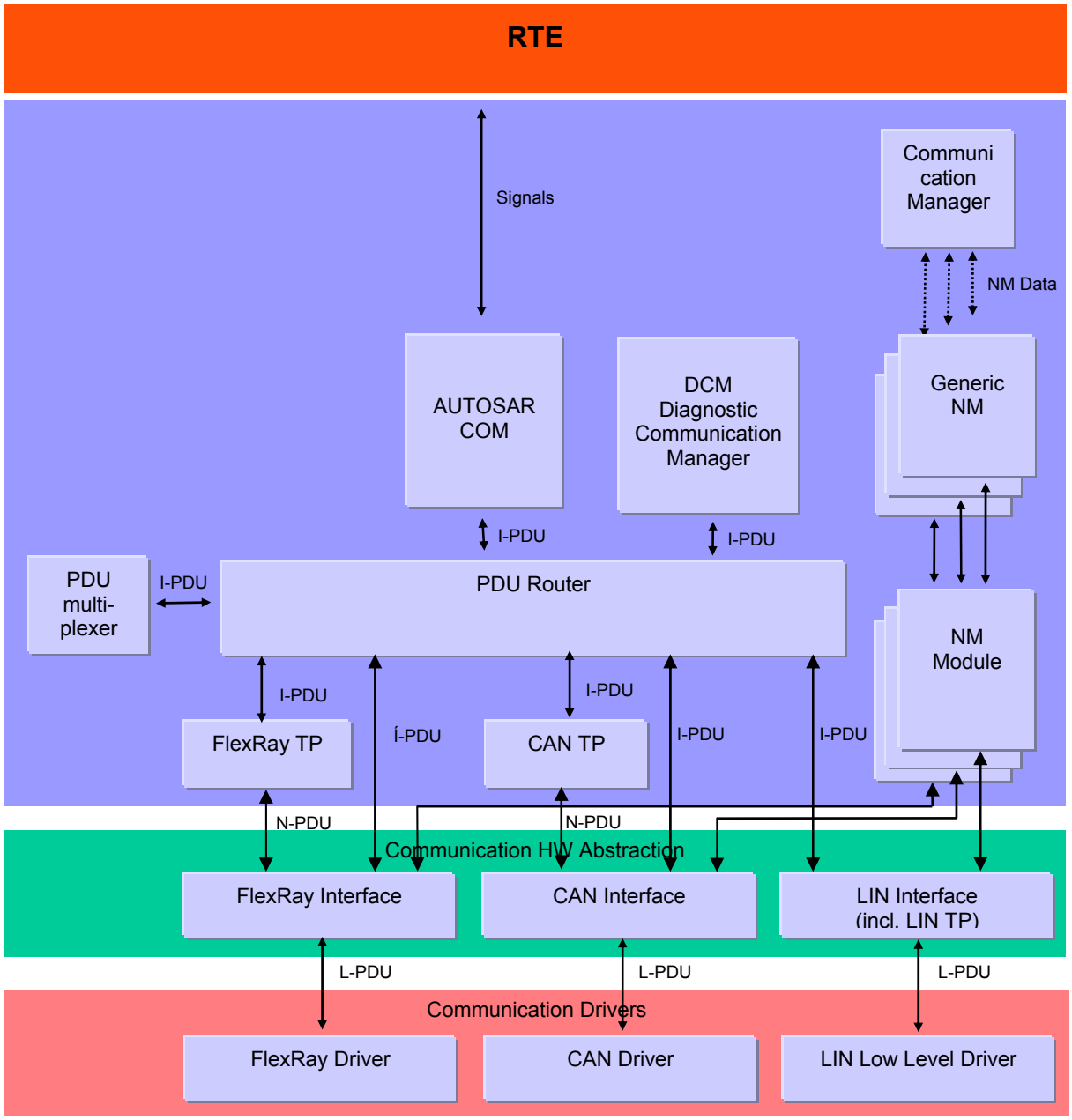


Figure 2 : AUTOSAR Signal Nomenclature

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 [15]. 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 notes from FrTp's point of view (transferring Fr N-PDUs).
Connection	A connection specifies a communication link between different communication notes from FrTp's point of view. A connection defines the sender / receiver relation of communication notes
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

3.2 Related standards and norms

[14] FlexRay Communications System Protocol Specification Version 2.1

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

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

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.

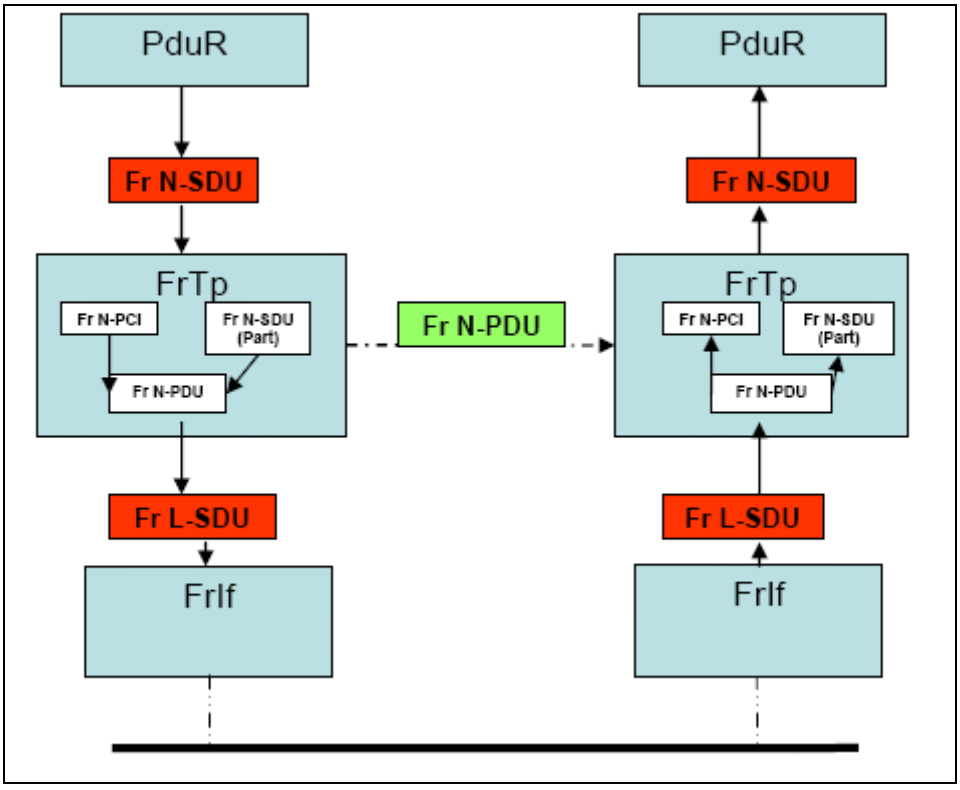


Figure 3: FrTp interactions

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).
- ***PduR_FrTpChangeParameterConfirmation***
  By this API service primitive, the FrTp module confirms the (un)successful change of module parameter(s).

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 (EcuM2859):

- ***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

[FRTP1000] ⌈The code-file structure of the FrTp module shall include the file `FrTp.c` containing the source code.⌋(BSW00380, BSW00381, BSW00419)

[FRTP1001] ⌈The code-file structure of the FrTp module shall include the file `FrTp_Lcfg.c` containing the link time configurable parameters.⌋ (BSW00344, BSW00380, BSW00381, BSW00419)

[FRTP1002] ⌈The code-file structure of the FrTp module shall include the file `FrTp_PBcfg.c` containing the post-build time configurable parameters.⌋(BSW00380, BSW00381, BSW00404, BSW00419)

5.6.2 Header file structure

[FRTP195] ⌈The header file structure of the FrTp module shall include the file FrTp.h.⌋()

[FRTP1157] ⌈FrTp.h shall contain all data exported from the FrTp – API declarations (except callbacks), extern types and global data.⌋()

[FRTP1003] ⌈The header file structure of the FrTp module shall include the file FrTp_Cfg.h containing pre-compile time configuration parameters.⌋ (BSW00412)

[FRTP1135] ⌈The header file structure of the FrTp module shall include the file FrTp_Cbk.h containing all callback function prototypes to be used by the lower layers.⌋()

[FRTP1004] ⌈The header file structure of the FrTp module shall include the following files:

- FrIf.h – header file of FrIf,
- 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

」()

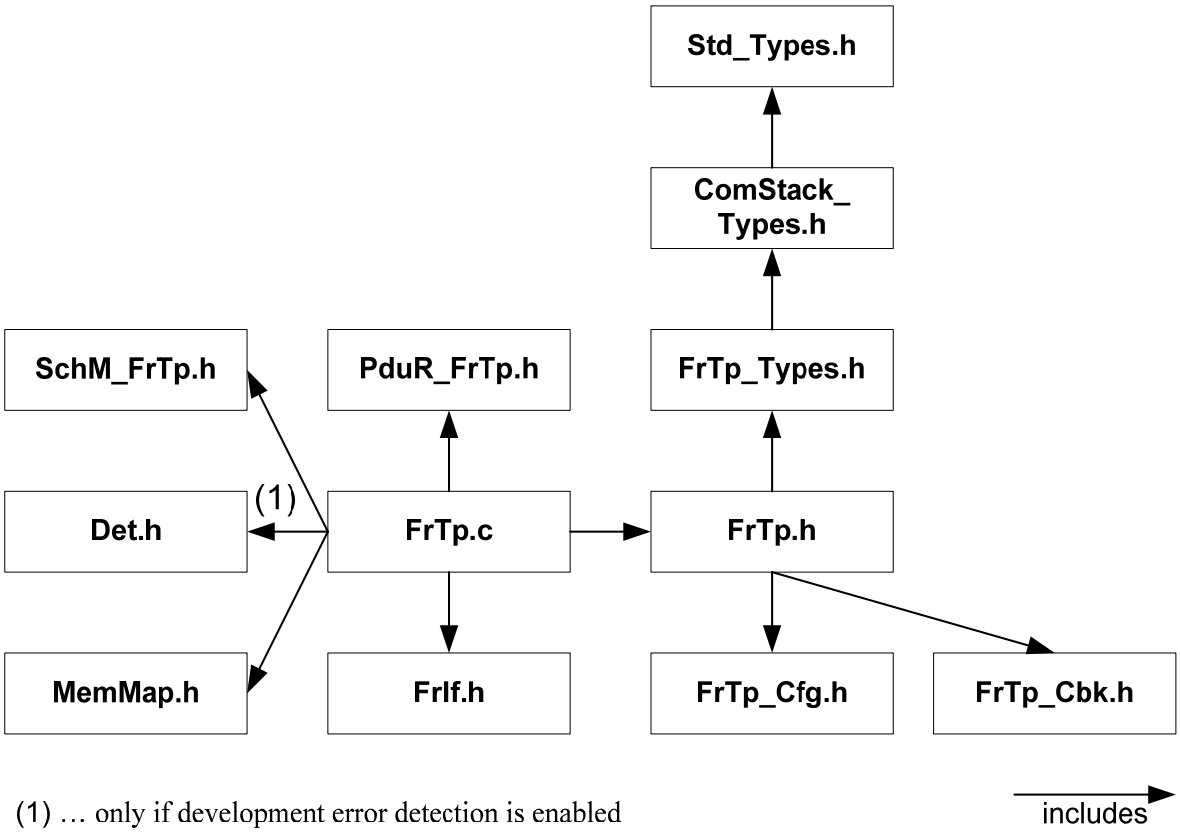


Diagram 1: File Structure

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

[FRTP200] 「Each code (*.c) file of the FrTp module shall provide data of version identification as defined in chapter 10.3.」(BSW004)

[FRTP1158] 「Each header (*.h) file of the FrTp module shall provide data of version identification as defined in chapter 10.3.」()

[FRTP1189] 「The FrTp module shall perform inter module checks to avoid integration of incompatible files. The imported included files shall be checked by preprocessing directives.」()

[FRTP1190] ⌈The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION

Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the FrTp module.⌋()

[FRTP1191] ⌈If the values are not identical to the expected values, an error shall be reported.⌋()

5.6.4 Design rules

[FRTP208] ⌈The code of the FrTp module (as long as it is written in C) shall conform to the HIS subset of the MISRA C Standard [16].⌋(BSW007)

[FRTP209] ⌈The source code of the FrTp module shall be neither compiler (tool) nor platform (processor) dependent.[2] ⌋()

[FRTP210] ⌈The code of the FrTp module (as long as it is written in C) shall indicate all global data with read-only purposes by explicitly assigning the `const` keyword. ⌋()

[FRTP211] ⌈The code of the FrTp module may provide macros instead of functions where source code is used and runtime is critical.⌋()

[FRTP212] ⌈All data/variables that shall be debugged and traced by AUTOSAR Debugging have to be defined as global variables. ⌋()

[FRTP1159] ⌈Type definitions of global variables which shall be debugged and traced by AUTOSAR shall be accessible by the standard module header file FrTp.h.[3].⌋()

[FRTP1129] ⌈The FlexRay Transport Layer module architecture shall support configuration modification by a dedicated update process (e.g. flash reprogramming)⌋(BSW05123)

[2] No compiler specific keywords shall be used.
[3] This allows the debugging toolchain to calculate the size of elements by C-"sizeof" and to (optionally) decode the structure elements.

6 Requirements traceability

Requirement	Satisfied by
-	FRTP1007
-	FRTP1157
-	FRTP1114
-	FRTP412
-	FRTP1081
-	FRTP206
-	FRTP1051
-	FRTP424
-	FRTP203
-	FRTP1185
-	FRTP202
-	FRTP1132
-	FRTP1139
-	FRTP1170
-	FRTP1147
-	FRTP1093
-	FRTP209
-	FRTP212
-	FRTP1138
-	FRTP1046
-	FRTP1102
-	FRTP1160
-	FRTP405
-	FRTP557
-	FRTP1116
-	FRTP1182
-	FRTP1161
-	FRTP1159
-	FRTP1040
-	FRTP153
-	FRTP1123
-	FRTP1004
-	FRTP1056
-	FRTP1029
-	FRTP1134
-	FRTP1054
-	FRTP580
-	FRTP1148

-	FRTP1063
-	FRTP386
-	FRTP179
-	FRTP1105
-	FRTP1045
-	FRTP1189
-	FRTP385
-	FRTP142
-	FRTP151
-	FRTP1143
-	FRTP1010
-	FRTP1021
-	FRTP1083
-	FRTP1181
-	FRTP1100
-	FRTP1092
-	FRTP1079
-	FRTP415
-	FRTP1190
-	FRTP1124
-	FRTP1145
-	FRTP1121
-	FRTP429
-	FRTP423
-	FRTP1115
-	FRTP1135
-	FRTP1168
-	FRTP1180
-	FRTP1172
-	FRTP1011
-	FRTP195
-	FRTP1036
-	FRTP421
-	FRTP1158
-	FRTP1091
-	FRTP1030
-	FRTP088
-	FRTP1080
-	FRTP1038
-	FRTP1049
-	FRTP1068
-	FRTP418

-	FRTP400
-	FRTP152
-	FRTP1162
-	FRTP1141
-	FRTP149
-	FRTP1039
-	FRTP1061
-	FRTP1014
-	FRTP1066
-	FRTP430
-	FRTP1072
-	FRTP141
-	FRTP1097
-	FRTP150
-	FRTP1020
-	FRTP137
-	FRTP1060
-	FRTP1169
-	FRTP1108
-	FRTP1078
-	FRTP1151
-	FRTP1013
-	FRTP1112
-	FRTP1042
-	FRTP1050
-	FRTP1106
-	FRTP1059
-	FRTP1088
-	FRTP1012
-	FRTP1052
-	FRTP1071
-	FRTP402
-	FRTP1047
-	FRTP1043
-	FRTP1184
-	FRTP416
-	FRTP1117
-	FRTP1041
-	FRTP1149
-	FRTP1067
-	FRTP422
-	FRTP399

-	FRTP1026
-	FRTP1167
-	FRTP1156
-	FRTP1015
-	FRTP1164
-	FRTP1009
-	FRTP555
-	FRTP1022
-	FRTP1074
-	FRTP1150
-	FRTP1017
-	FRTP1104
-	FRTP1075
-	FRTP1090
-	FRTP1048
-	FRTP1101
-	FRTP217
-	FRTP1062
-	FRTP1191
-	FRTP148
-	FRTP1032
-	FRTP1188
-	FRTP218
-	FRTP1064
-	FRTP162
-	FRTP1028
-	FRTP1136
-	FRTP1186
-	FRTP1183
-	FRTP1033
-	FRTP1019
-	FRTP1166
-	FRTP1053
-	FRTP1152
-	FRTP498
-	FRTP1025
-	FRTP384
-	FRTP1058
-	FRTP1057
-	FRTP1095
-	FRTP1110
-	FRTP1120

-	FRTP1113
-	FRTP1118
-	FRTP154
-	FRTP1076
-	FRTP1096
-	FRTP577
-	FRTP1044
-	FRTP1144
-	FRTP1055
-	FRTP1107
-	FRTP579
-	FRTP1099
-	FRTP1008
-	FRTP210
-	FRTP1094
-	FRTP1140
-	FRTP1084
-	FRTP242
-	FRTP211
-	FRTP1163
-	FRTP1165
-	FRTP1065
-	FRTP428
-	FRTP1069
-	FRTP1077
-	FRTP1146
-	FRTP1111
-	FRTP1070
-	FRTP578
-	FRTP1109
-	FRTP205
-	FRTP1187
-	FRTP1178
-	FRTP136
-	FRTP1089
-	FRTP228
BSW00305	FRTP1133
BSW00306	FRTP9999
BSW00312	FRTP9999
BSW00314	FRTP9999
BSW00323	FRTP9999
BSW00325	FRTP9999

BSW00326	FRTP9999
BSW00328	FRTP9999
BSW00329	FRTP9999
BSW00330	FRTP9999
BSW00331	FRTP9999
BSW00333	FRTP9999
BSW00335	FRTP9999
BSW00341	FRTP9999
BSW00343	FRTP9999
BSW00344	FRTP1001
BSW00345	FRTP9999
BSW00347	FRTP9999
BSW00350	FRTP9999
BSW00358	FRTP9999
BSW00370	FRTP9999
BSW00371	FRTP9999
BSW00373	FRTP9999
BSW00375	FRTP9999
BSW00376	FRTP9999
BSW00377	FRTP9999
BSW00380	FRTP1002, FRTP1001, FRTP1000
BSW00381	FRTP1002, FRTP1001, FRTP1000
BSW00386	FRTP9999
BSW00387	FRTP9999
BSW004	FRTP200
BSW00401	FRTP9999
BSW00404	FRTP1002
BSW00405	FRTP9999
BSW00409	FRTP9999
BSW00410	FRTP9999
BSW00412	FRTP1003
BSW00413	FRTP9999
BSW00414	FRTP9999
BSW00415	FRTP9999
BSW00417	FRTP9999
BSW00419	FRTP1002, FRTP1001, FRTP1000
BSW00423	FRTP9999
BSW00424	FRTP9999
BSW00425	FRTP9999
BSW00426	FRTP9999
BSW00427	FRTP9999
BSW00428	FRTP9999

BSW00429	FRTP9999
BSW00431	FRTP9999
BSW00432	FRTP9999
BSW00433	FRTP9999
BSW00434	FRTP9999
BSW005	FRTP9999
BSW006	FRTP9999
BSW007	FRTP208
BSW009	FRTP9999
BSW010	FRTP9999
BSW05073	FRTP1006, FRTP1005
BSW05077	FRTP1018
BSW05082	FRTP9999
BSW05083	FRTP1006, FRTP1005
BSW05084	FRTP1006, FRTP1005
BSW05088	FRTP1034, FRTP1035
BSW05089	FRTP1037
BSW05093	FRTP1006, FRTP1005
BSW05095	FRTP1006, FRTP1005
BSW05123	FRTP1129
BSW101	FRTP147
BSW159	FRTP9999
BSW160	FRTP9999
BSW161	FRTP9999
BSW162	FRTP9999
BSW164	FRTP9999
BSW167	FRTP9999
BSW168	FRTP9999
BSW172	FRTP9999

6.1 General Requirements on Basic Software Modules

Requirement		***Satisfied by***
BSW003	Version identification	
BSW00305	Self-defined data types naming convention	[FRTP1133
BSW00306	Avoid direct use of compiler and platform specific keywords	n/a
BSW00312	Shared code shall be reentrant	n/a
BSW00314	Separation of interrupt frames and service routines	n/a
BSW00318	Format of module version numbers	[FRTP
BSW00321	Enumeration of module version numbers	[FRTP
BSW00323	API parameter checking	n/a
BSW00325	Runtime of interrupt service routines	n/a
BSW00326	Transition from ISRs to OS tasks	n/a
BSW00328	Avoid duplication of code	n/a
BSW00329	Avoidance of generic interfaces	n/a

BSW00330	Usage of macros / inline functions instead of functions	n/a
BSW00331	Separation of error and status values	n/a
BSW00333	Documentation of callback function context	n/a
BSW00335	Status values naming convention	n/a
BSW00336	Shutdown interface	Chapter 7.4
BSW00337	Classification of errors	Chapter 7.7.3
BSW00338	Detection and Reporting of development errors	Chapter 7.7.3
BSW00339	Reporting of production relevant error status	Chapter 7.7.3
BSW00341	Microcontroller compatibility documentation	n/a
BSW00343	Specification and configuration of time	n/a
BSW00344	Reference to link-time configuration	[FRTP1001
BSW00345	Pre-compile-time configuration	n/a
BSW00346	Basic set of module files	Chapter 5.6
BSW00347	Naming separation of different instances of BSW drivers	n/a
BSW00348	Standard type header	Chapter 5.6
BSW00350	Development error detection keyword	n/a
BSW00353	Platform specific type header	Chapter 5.6
BSW00357	Standard API return type	Chapter 8.3
BSW00358	Return type of init() functions	n/a
BSW00359	Return type of callback functions	Chapter 8.2.1
BSW00360	Parameters of callback functions	Chapter 8.2.1
BSW00361	Compiler specific language extension header	Chapter 5.6
BSW00369	Do not return development error codes via API	Chapter 7.7.3
BSW00370	Separation of callback interface from API	n/a
BSW00371	Do not pass function pointers via API	n/a
BSW00373	Main processing function naming convention	n/a
BSW00374	Module vendor identification	[FRTP
BSW00375	Notification of wake-up reason	n/a
BSW00376	Return type and parameters of main processing functions	n/a
BSW00377	Module specific API return types	n/a
BSW00379	Module identification	[FRTP
BSW00380	Separate C-Files for configuration parameters	[FRTP1000, [FRTP1001, [FRTP1002
BSW00381	Separate configuration header file for pre-compile time parameters	[FRTP1000, [FRTP1001, [FRTP1002
BSW00383	List dependencies of configuration files	Chapter 5
BSW00384	List dependencies to other modules	Chapter 5
BSW00385	List possible error notifications	Chapter 8.2.1
BSW00386	Configuration for detecting an error	n/a
BSW00387	Specify the configuration class of callback function	n/a
BSW00388	Introduce containers	Chapter 10.2
BSW00389	Containers shall have names	Chapter 10.2
BSW00390	Parameter content shall be unique within the module	Chapter 10.2
BSW00391	Parameter shall have unique names	Chapter 10.2
BSW00392	Parameters shall have a type	Chapter 10.2
BSW00393	Parameters shall have a range	Chapter 10.2
BSW00394	Specify the scope of the parameters	Chapter 10.2
BSW00395	List the required parameters (per parameter)	Chapter 10.2
BSW00396	Configuration classes	Chapter 10.2
BSW00397	Pre-compile-time parameters	Chapter 10.2
BSW00398	Link-time parameters	Chapter 10.2
BSW00399	Loadable Post-build time parameters	Chapter 10.2
BSW004	Version check	[FRTP200, FRTP201
BSW00400	Selectable Post-build time parameters	Chapter 10.2
BSW00401	Documentation of multiple instances of configuration parameters	n/a

BSW00402	Published information	Chapter 10.3
BSW00404	Reference to post build time configuration	[FRTP1002
BSW00405	Reference to multiple configuration sets	n/a
BSW00406	Check module initialization	Chapter 7.4
BSW00407	Function to read out published parameters	[FRTP215
BSW00409	Header files for production code error IDs	n/a
BSW00410	Compiler switches shall have defined values	n/a
BSW00412	Separate H-File for configuration parameters	[FRTP1003
BSW00413	Accessing instances of BSW modules	n/a
BSW00414	Parameter of init function	n/a
BSW00415	User dependent include files	n/a
BSW00416	Sequence of Initialization	Chapter 7.4
BSW00417	Reporting of Error Events by Non-Basic Software	n/a
BSW00419	Separate C-Files for pre-compile time configuration parameters	[FRTP1000, [FRTP1001, [FRTP1002
BSW00423	Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	n/a
BSW00424	BSW main processing function task allocation	n/a
BSW00425	Trigger conditions for schedulable objects	n/a
BSW00426	Exclusive areas in BSW modules	n/a
BSW00427	ISR description for BSW modules	n/a
BSW00428	Execution order dependencies of main processing functions	n/a
BSW00429	Restricted BSW OS functionality access	n/a
BSW00431	The BSW Scheduler module implements task bodies	n/a
BSW00432	Modules should have separate main processing functions for read/receive and write/transmit data path	n/a
BSW00433	Calling of main processing functions	n/a
BSW00434	The Schedule Module shall provide an API for exclusive areas	n/a
BSW00435	Header File Structure for the Basic Software Scheduler	Chapter 5.6.2
BSW00436	Module Header File Structure for the Memory Mapping	Chapter 5.6.2
BSW005	No hard coded horizontal interfaces within MCAL	n/a
BSW006	Platform independency	n/a
BSW007	HIS MISRA C	[FRTP208
BSW009	Module User Documentation	n/a
BSW010	Memory resource documentation	n/a
BSW101	Initialization interface	[FRTP147
BSW158	Separation of configuration from implementation	Chapter 5.6.1
BSW159	Tool-based configuration	n/a
BSW160	Human-readable configuration data	n/a
BSW161	Microcontroller abstraction	n/a
BSW162	ECU layout abstraction	n/a
BSW164	Implementation of interrupt service routines	n/a
BSW167	Static configuration checking	n/a
BSW168	Diagnostic Interface of SW components	n/a
BSW171	Configurability of optional functionality	Chapter 10.4
BSW172	Compatibility and documentation of scheduling strategy	n/a

6.2 Requirements on FlexRay

Requirement		*Satisfied by*
BSW05073	Compliance with ISO 10681-2	[FRTP1005, [FRTP1006
BSW05074	Location of the FlexRay Transport Layer software module.	Chapter 1
BSW05075	Independence of the Network Configuration.	Chapter 1
BSW05076	Support of at least 32 logical FlexRay Transport Layer Channels being used concurrently.	FRTP004_Conf :

BSW05077	Identification of each N-SDU with a unique identifier.	[FRTP1018
BSW05079	Individual properties of each N-SDU.	Chapter 10.2
BSW05082	Support acknowledgement without retry.	n/a
BSW05083	Support acknowledgement with retry.	[FRTP1005, [FRTP1006
BSW05084	Maximum length of PDUs	[FRTP1005, [FRTP1006
BSW05088	Interface for initialization.	[FRTP1034, [FRTP1035
BSW05089	The FlexRay Transport Layer services shall not be operational before initializing the module.	[FRTP1037
BSW05090	Support of the Change Parameter Service according to ISO 10681-2.	Chapter 7.5.7
BSW05093	Provide a transmit cancellation service to the sender and the receiver.	[FRTP1005, [FRTP1006, Chapter 7.5.6
BSW05095	Dynamic control of used bandwidth.	[FRTP1005, [FRTP1006
BSW05123	Modification of configuration data by a flashing process	[FRTP1129

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 [15].

[FRTP1005] 「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 [15].」(BSW05073, BSW05083, BSW05084, BSW05093, BSW05095)

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.

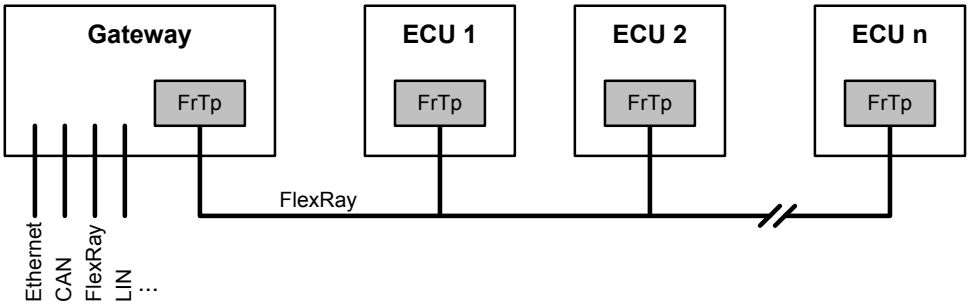


Figure 4: FrTp usage scenarios

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)

[FRTP1006] ⌈The FrTp module shall support the Fr N-PDU formats as defined in [15] ISO10681-2, Road vehicles – Communication on FlexRay – Part 2: Communication Layer services⌋(BSW05073, BSW05083, BSW05084, BSW05093, BSW05095)

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 [15] 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

[FRTP1007] ⌈ According to ISO 10681-2 [15] the FrTp module shall support an unsegmented unacknowledged data transfer with known message length as depict in Figure 5.

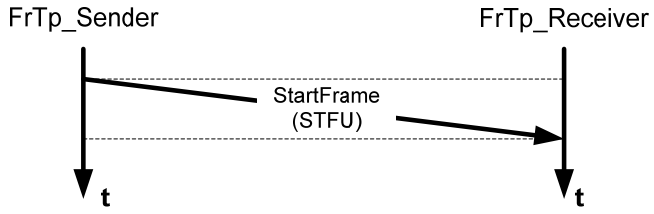


Figure 5: Frame sequence of an unsegmented unacknowledged data transfer with known message length ⌋()

7.2.2.2 Unsegmented acknowledged data transfer with known message length

[FRTP1008] ⌈ According to ISO 10681-2 [15] the FrTp module supports an unsegmented acknowledged data transfer with known message length as depict in Figure 6.

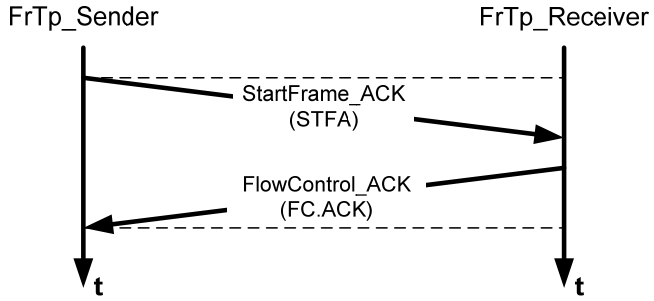


Figure 6: Frame sequence of an unsegmented acknowledged data transfer with known message length ⌋()

7.2.2.3 Segmented unacknowledged data transfer with known message length

[FRTP1009] ⌈ According to ISO 10681-2 [15] the FrTp module shall support a segmented unacknowledged data transfer with known message length as depict in Figure 7.

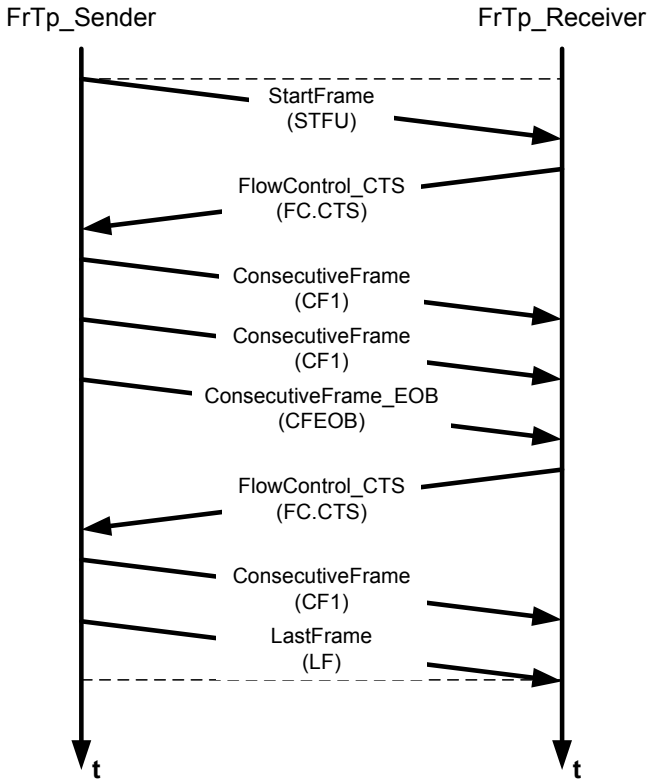


Figure 7: Frame sequence a segmented unacknowledged data transfer with known message length ()

7.2.2.4 Segmented acknowledged data transfer with known message length

[FRTP1010] ⌈According to ISO 10681-2 [15] the FrTp module shall support a segmented acknowledged data transfer with known message length as depict in Figure 8.

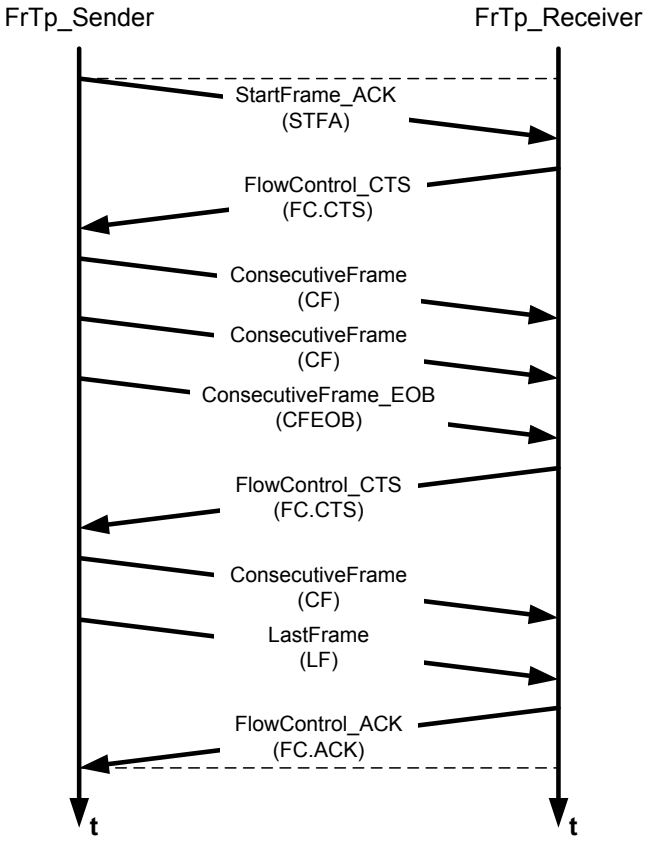


Figure 8: Frame sequence of a segmented acknowledged data transfer with known message length 」()

7.2.2.5 Segmented unacknowledged data transfer with unknown message length

[FRTP1011] 「According to ISO 10681-2 [15] the FrTp module shall support a segmented unacknowledged data transfer with unknown message length as depict in Figure 9.

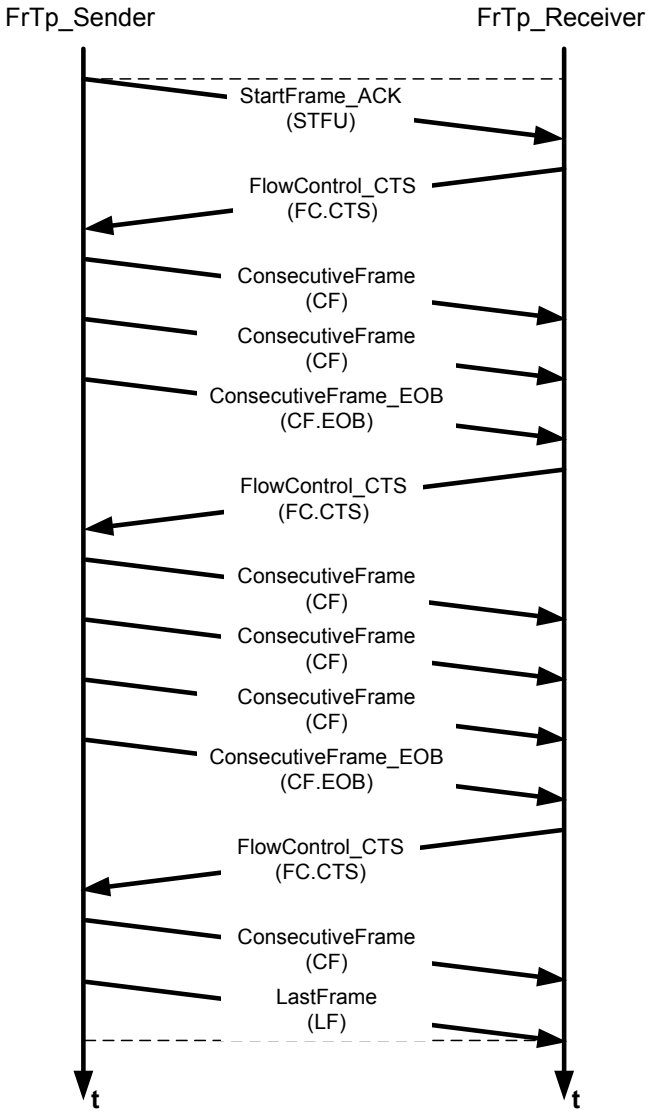


Figure 9: Frame sequence of a segmented unacknowledged data transfer with unknown message length ()

7.2.2.6 Segmented acknowledged data transfer with unknown message length

[FRTP1012] ⌈According to ISO 10681-2 [15] the FrTp module shall support a segmented acknowledged data transfer with unknown message length as depict in Figure 10.

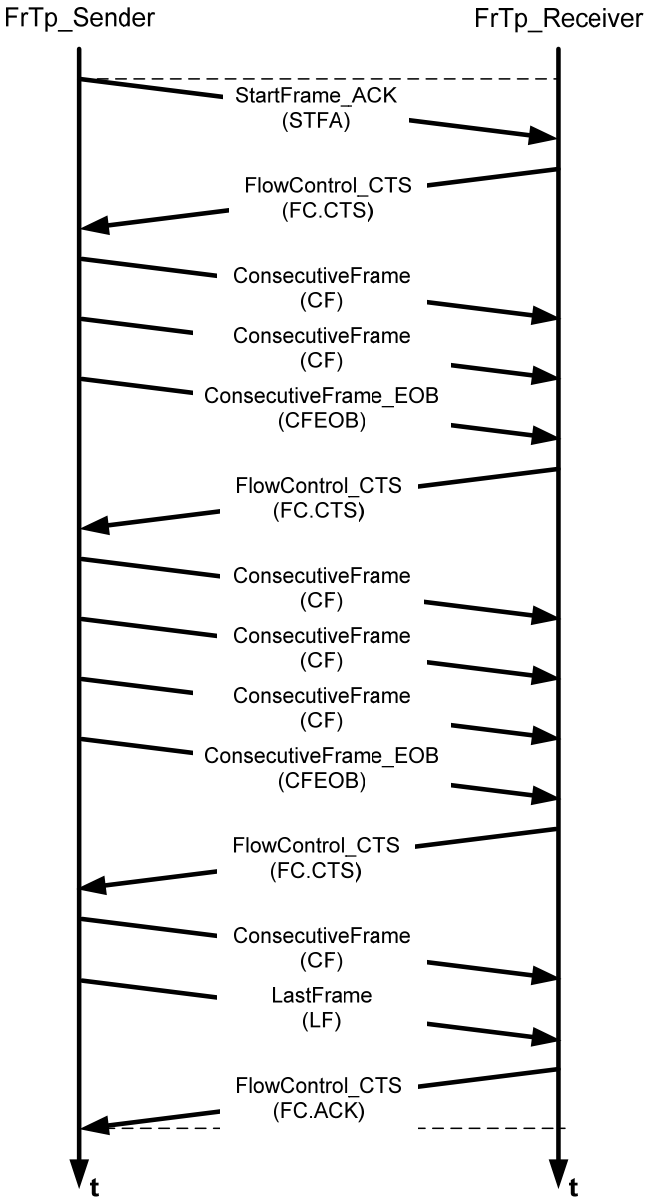


Figure 10: Frame sequence of a segmented acknowledged data transfer with unknown message length ⌋()

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 [14].

7.3.1 Overview

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

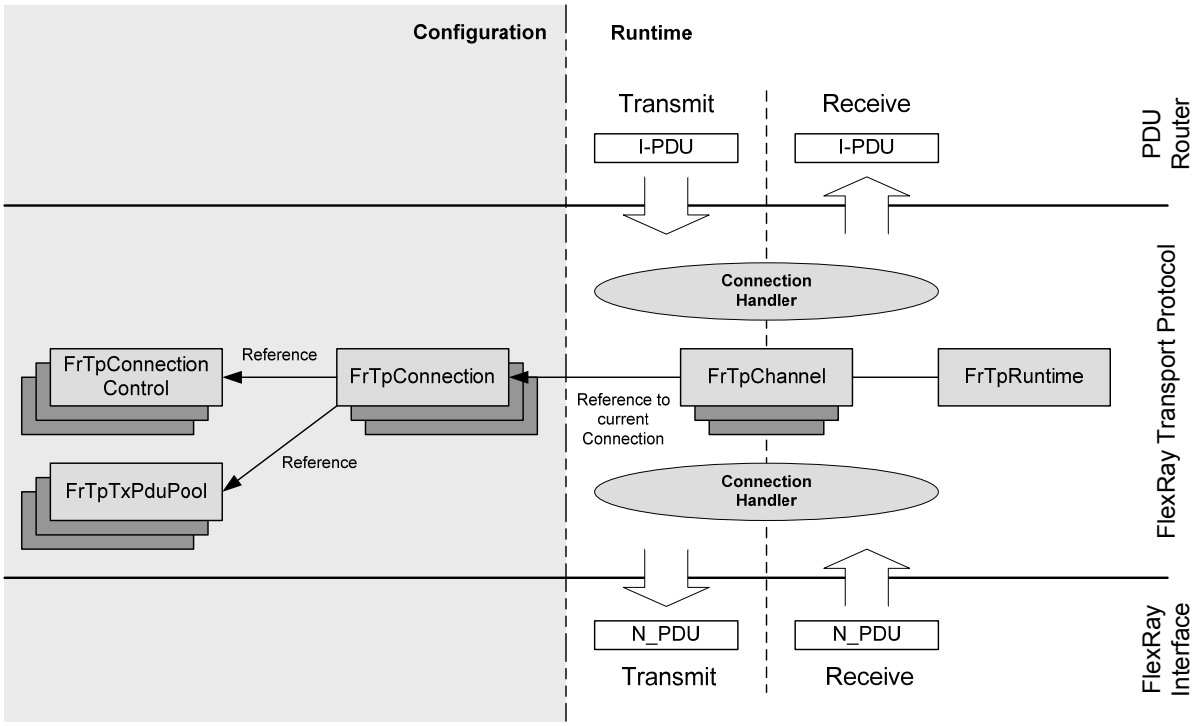


Figure 11: FlexRay Transport Module overview

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).

[FRTP1013] ⌈An `FrTpConnection` container shall implement all parameters as defined in chapter 10.2.⌋()

[FRTP1014] ⌈For each connection link the FrTp module shall handle, a new instance of `FrTpConnection` container shall be created. ⌋()

[FRTP1015] ⌈The FrTp module shall support a post build time configurable number of connections[4]. ⌋()

[FRTP1017] ⌈Each `FrTpConnection` shall have a module wide unique `RemoteAddress / LocalAddress` pair `(see` section 10.2`)`.[5]⌋()

[FRTP1018] ⌈ Each `FrTpConnection` shall have a module wide unique `FRTP_SDUID` (N-SDU ID) `(see` section 10.2`)`.⌋(BSW05077)

7.3.2.2 FrTpTxPduPool

The `FrTpTxPduPool` contains a list of N-PDUs configured for sending FrTp N-PDUs. An `FrTpTxPduPool` could be referenced by different `FrTpConnections` but each `FrTpConnection` has exactly one reference to one `FrTpTxPduPool`. (see also Figure 17). The `FrTpTxPduPools` are necessary to support dynamic bandwidth assignment for connections.

[4] 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.
[5] The AUTOSAR local address and remote address is mapped to the ISO 10681-2 source address and target address.

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.

[FRTP1019] ⌈ An `FrTpTxPduPool` container shall implement all parameters as defined in chapter 10.2.9.⌋()

[FRTP1020] ⌈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`[6].

[FRTP1021] ⌈ An `FrTpConnectionControl` Container shall implement all parameters as defined in chapter 10.2.⌋()

[FRTP1022] ⌈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.

7.3.3.1 FrTpRuntime

Module Name	FrTpRuntime

[6] Use case: Reducing configuration control container instances

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	

[FRTP228] ⌈ The FrTp module shall support concurrently work of multiple `FrTpChannels`[7].⌋()

[FRTP088] ⌈The exact number of provided channels shall be configurable by the parameter FrTpChanNum (see section 10.2)⌋()

[FRTP1025] ⌈ The runtime variable *FrTpRxChannelState* shall be switched from "`idle`" state to "`busy`" state if the channel is allocated for an Rx connection.⌋()

[FRTP1026] ⌈ The runtime variable *FrTpRxChannelState* shall be switched from "`busy`" state to "`idle`" state if the channel is free after an Rx connection is closed.⌋()

[FRTP1117] ⌈ The runtime variable *FrTpTxChannelState* shall be switched from "`idle`" state to "`busy`" state if the channel is allocated for an Tx connection.⌋()

[FRTP1118] ⌈ The runtime variable *FrTpTxChannelState* shall be switched from "`busy`" state to "`idle`" state if the channel is free after an Tx connection is closed.⌋()

Note: The error handling for the case if no FrTpChannel resource is available (*FrTpTxChannelState* ≠ idle) for data transmission is specified in [FRTP1041.

7 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.

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

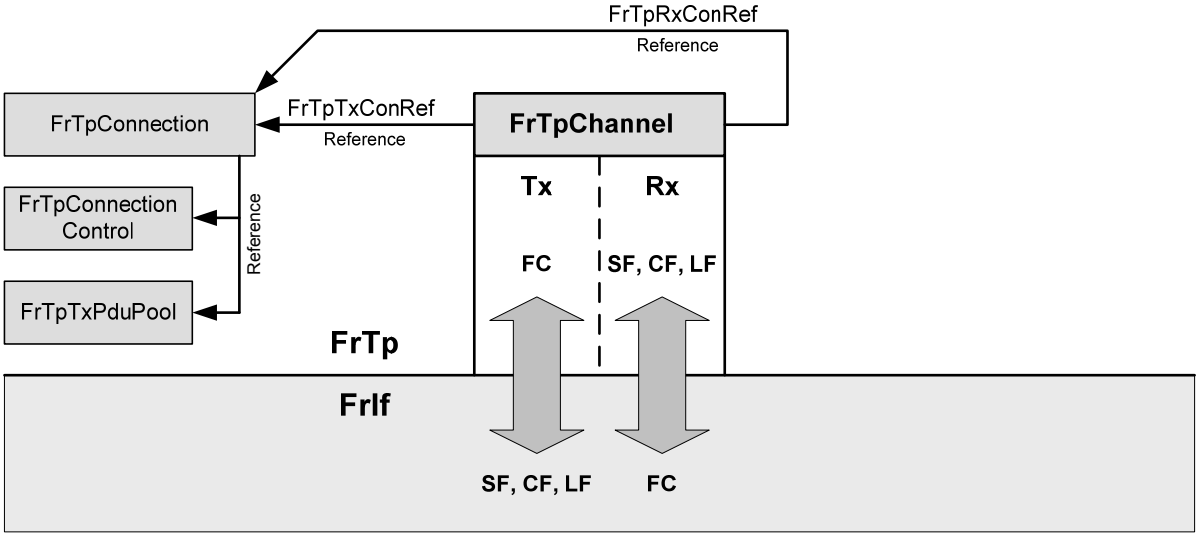


Figure 12: Full Duplex Overview

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.

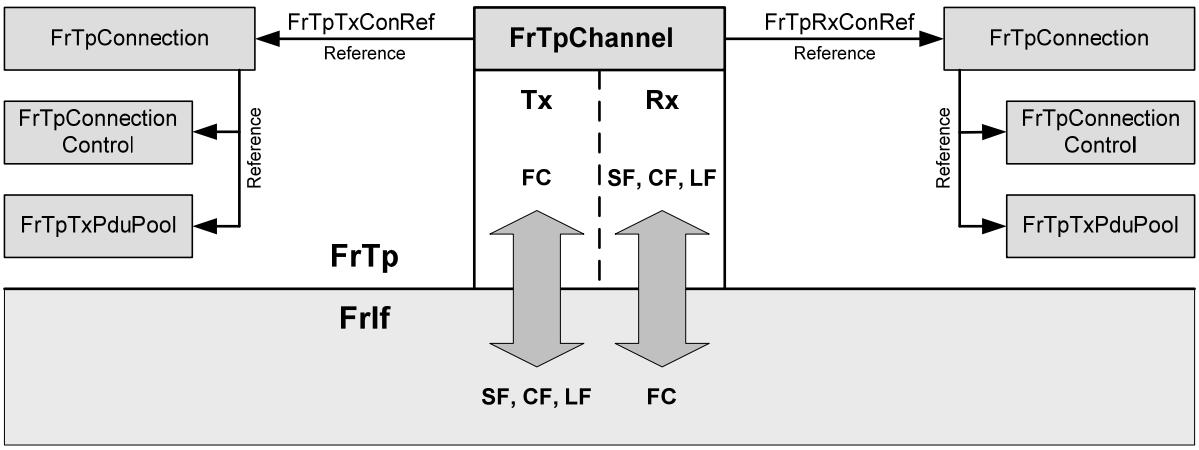


Figure 13: Half Duplex Overview

[8] 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.

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

[FRTP1028] ⌈The FrTp module shall have two internal states, `FRTP_OFF` and `FRTP_ON`.⌋()

[FRTP1029] ⌈The FrTp module shall implement a static status variable *`FrTpState`* to denote whether the FrTp module is initialized or not[9].⌋()

[FRTP1030] ⌈ The FrTp module shall be in the `FRTP_OFF` state after power up. ⌋()

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

[FRTP1032] ⌈The FrTp module shall change to the internal state `FRTP_ON` when the FrTp has been successfully initialized by the service primitive `FrTp_Init()`.⌋()

[FRTP1033] ⌈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[10].⌋()

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

[FRTP1034] ⌈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. ⌋(BSW05088)

[FRTP1035] ⌈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. ⌋ (BSW05088)

[FRTP1036] ⌈The FrTp module shall change to the internal state `FRTP_OFF` when the service primitive `FrTp_Shutdown()` has been executed successfully.⌋()

[FRTP1120] ⌈ Additionally to [FRTP1035 and FRTP1036 if `TransmitCancellation` is enabled, all currently active Tx connections have to be canceled and the cancellations have to be reported to upper layers. ⌋()

[FRTP1037] ⌈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. ⌋(BSW05089)

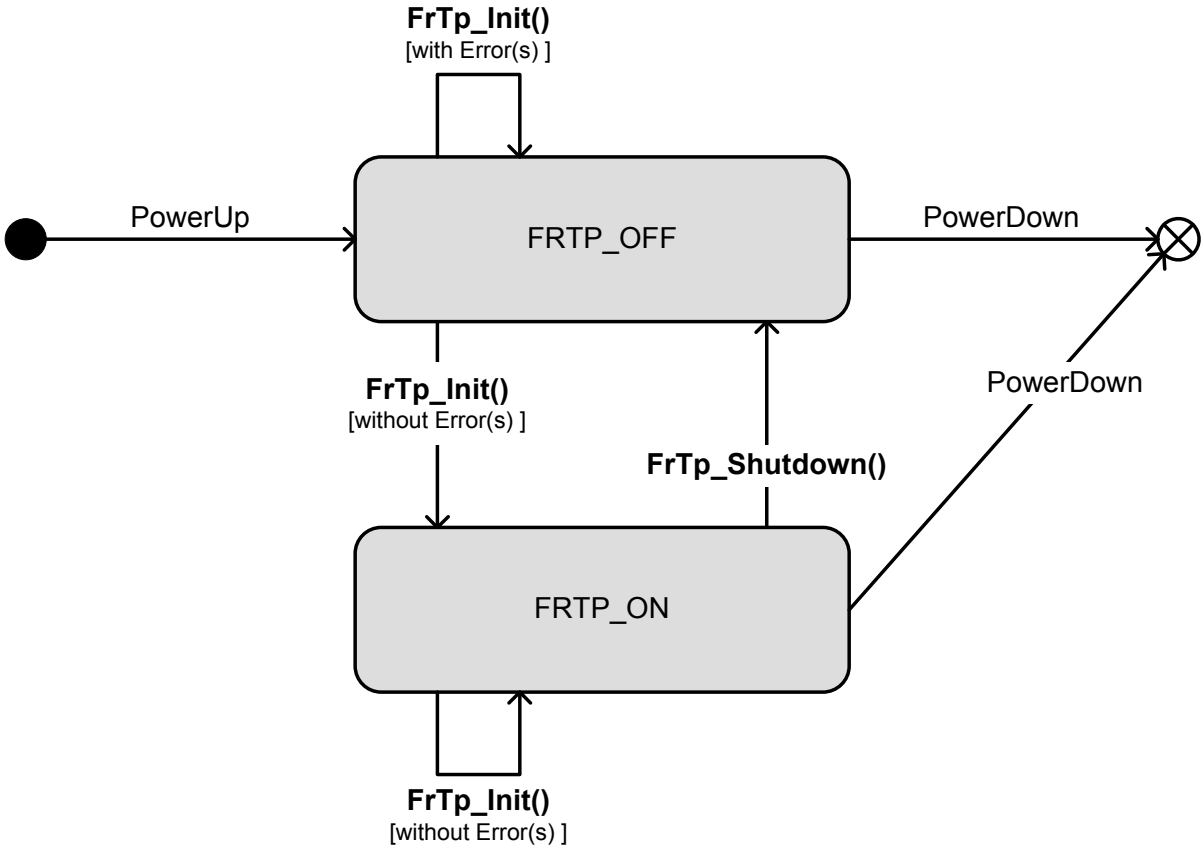


Diagram 2: FrTp Initialization and shutdown state diagram

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.

[FRTP415] ⌈The `TX_SDU_AVAILABLE` flag shall exist for every channel.⌋()

[FRTP416] ⌈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 [FRTP1057.

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.

[FRTP1101] ⌈The `TX_SDU_UNKNOWN_MSG_LENGTH` flag shall indicate an Fr N-SDU transmit request with unknown message length.⌋()

[FRTP1102] ⌈ 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 [FRTP1124.

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.

[FRTP418] ⌈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 [FRTP1074.

7.5.1.4 TC_REQUEST (Transmit Cancelation)

[FRTP422] ⌈The TC_REQUEST flag shall exist for every channel (twice for Full Duplex channels).⌋()

[FRTP423] ⌈The TC_REQUEST flag shall be set by the service primitive call *FrTp_CancelTransmit* if the service returns E_OK. ⌋()

[FRTP1104] ⌈The TC_REQUEST flag shall be processed either

a) before copying TxData (PduR_FrTpCopyTxData) or
b) before sending (*FrIf_Transmit or FrTp_TriggerTransmit)* an FrTpN-PDU.⌋()

[FRTP424] ⌈ The TC_REQUEST flag shall be cleared after processing the cancellation request.⌋()

7.5.1.5 RX_ERROR

The RX_ERROR[11] flag is required in case transmission with acknowledgement and retry is configured. During a segmented data reception an error could occur but

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

sending FlowControl is currently not possible. In that case the information about an error has to be stored until sending a FlowControl is allowed.

[FRTP428] ⌈The RX_ERROR flag shall exist for every `FrTpChannel`.⌋()

[FRTP429] ⌈The RX_ERROR flag shall indicate that an error occurred during a segmented reception.⌋()

[FRTP430] ⌈The RX_ERROR flag shall be cleared after the reaction (Retry, Negative Acknowledgement, abortion).⌋()

7.5.2 Transmit Data

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.

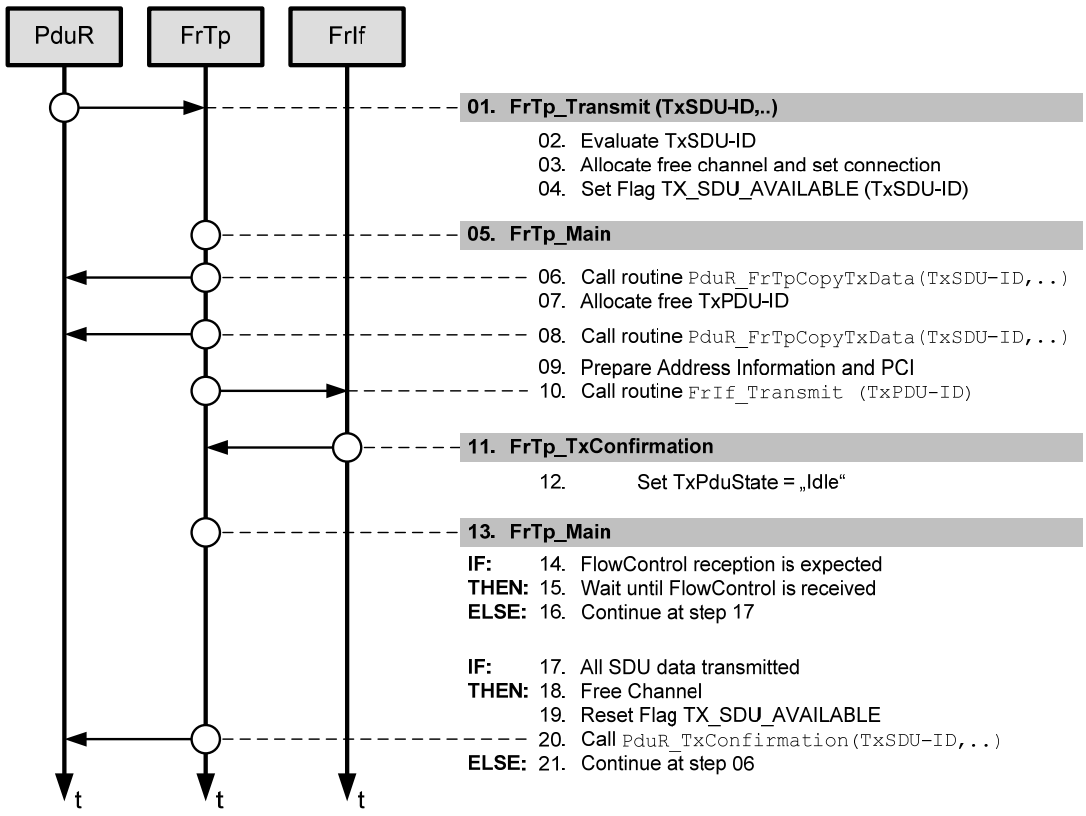


Figure 14: Transmit data overview in 'Immediate Buffer Access' mode

Figure 14 depicts the internal processing for data transmission in principle[12] if "Immediate Buffer Access" mode is configured[13]. Below there is a description of the different steps which are necessary to transmit data via FrTp.

Step 1 - 4

[FRTP136] ⌈Sending Fr N-SDU data shall always be initiated by the service primitive call of *FrTp_Transmit* (see chapter 8.3.3.1).⌋()

[FRTP1043] ⌈The FrTp module shall evaluate the value of PduInfoType.SduLength:

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

[FRTP1044] ⌈If support unknown message length is configured the FrTp module shall set the flag `TX_SDU_UNKNOWN_MSG_LENGTH` according to the result of [FRTP1043 (see also chapter 7.5.1.2).⌋()

[FRTP1134] ⌈The FRTP module shall raise an development error FRTP_E_UMSG_LENGTH_ERROR when
a) a transmission with unknown message length is detected and
b) support of unknown message legth is not configured and
c) 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.

[FRTP1038] ⌈An ongoing data transfer shall be signalled by the flag `TX_SDU_AVAILABLE` (see also chapter 7.5.1.1).⌋()

[FRTP1039] ⌈The service primitive shall set the flag `TX_SDU_AVAILABLE` only, if
a) the requested `FrTpTxSduId` is valid

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

b) a free channel resource is available (FrTpTxChannelState = Idle)」()

[FRTP1040] 「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.」()

[FRTP1041] 「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)[14].」()

[FRTP1185] 「 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()[15] 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.

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

[15] 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.

[FRTP1042] ⌈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.⌋()

[FRTP1045] ⌈The FrTp module shall always allocate the first free `FrTpTxPdu` while scanning the corresponding `FrTpTxPduPool` (see also chapter 7.3.2.2).⌋()

[FRTP1046] ⌈If a free `FrTpTxPdu` is identified, the FrTp module shall use this `FrTpTxPdu` to continue current transmission process.⌋()

[FRTP1047] ⌈If no free `FrTpTxPdu` is identified, the FrTp module shall stop processing for the corresponding connection within the current task.⌋()

[FRTP1048] ⌈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). ⌋()

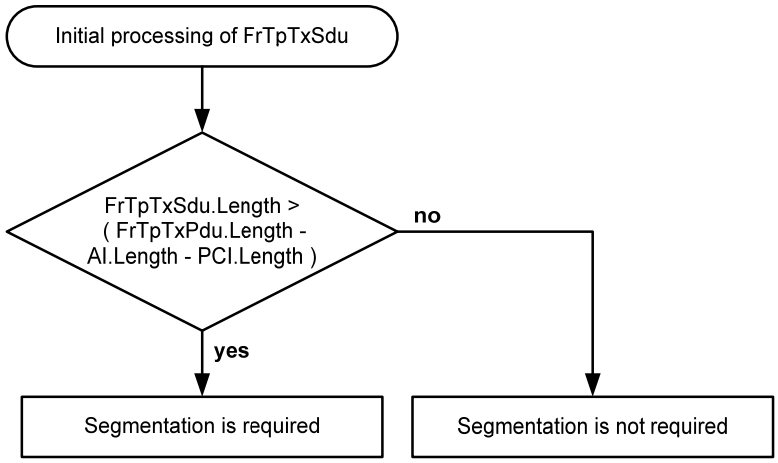


Diagram 3: Segmentation decision

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.

[FRTP1123] ⌈ 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.⌋()

[FRTP1049] ⌈The FrTp module shall prepare the Address Information and PCI according to the result of [FRTP1048 as defined in specification ISO 10681-2 [15].⌋()

[FRTP1050] ⌈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.⌋()

[FRTP1051] ⌈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[16].⌋()

[16] 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`.

[FRTP1052] ⌈The service primitive `FrTp_TxConfirmation` shall be called by the underlying layer module after a successful transmission of the corresponding N-PDU.⌋()

[FRTP1053] ⌈The service primitive `FrTp_TxConfirmation` shall be called by the underlying layer module with the corresponding `FrTpTxPduId` of the successful transmitted PDU.⌋()

[FRTP1054] ⌈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.

[FRTP1055] ⌈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.

[FRTP1056] ⌈ 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.
⌋()

[FRTP1057] ⌈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.
⌋()

[FRTP1124] ⌈ 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.⌋()

[FRTP1058] ⌈ The FrTp module shall call the service primitive `PduR_FrTpTxConfirmation` for the corresponding `FrTpTxSduId` 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[17] if "Decoupled Buffer Access" mode is configured[18]. Below there is a description of the different steps which are necessary to transmit data via FrTp.

[17] 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.

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

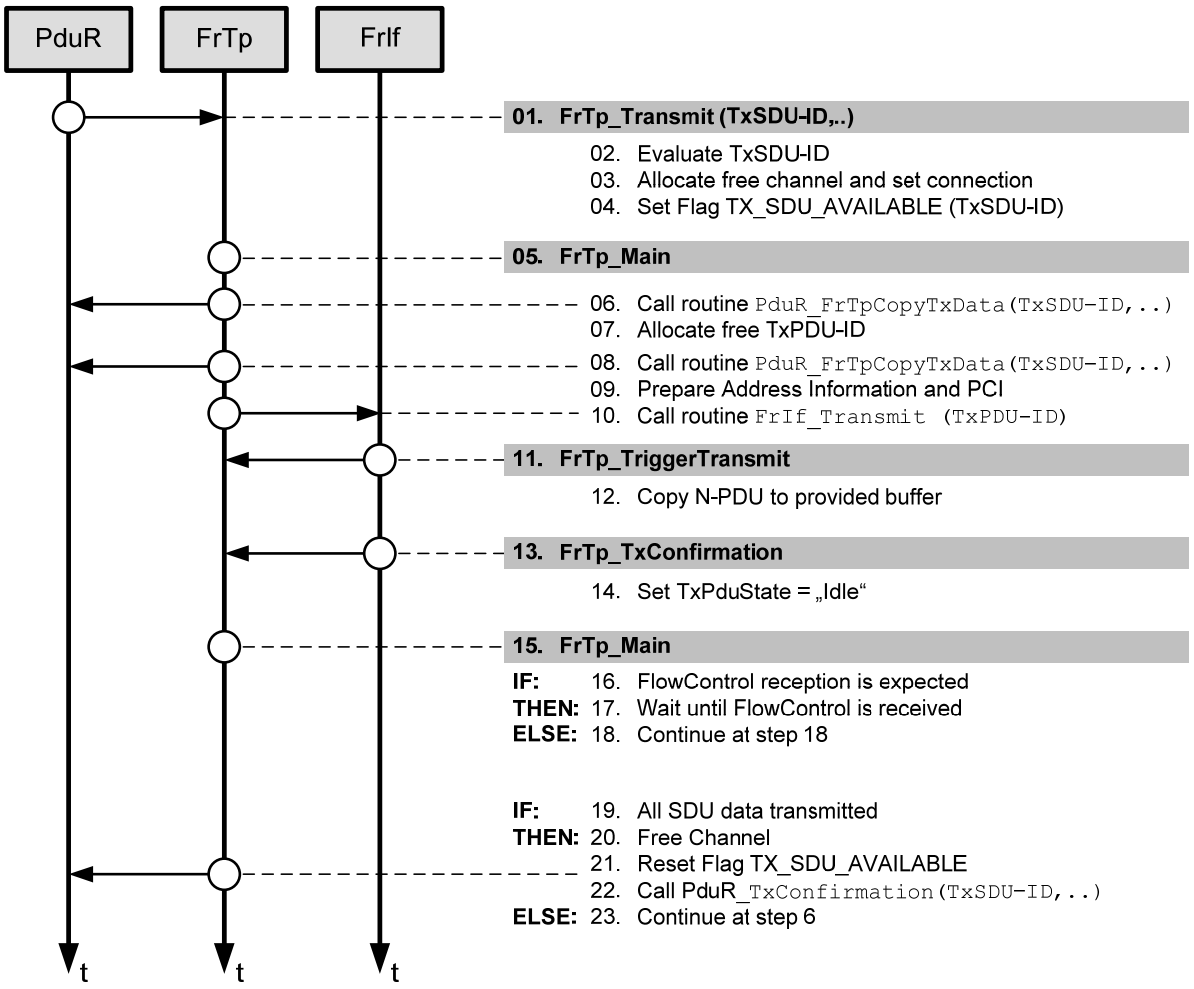


Figure 15: Transmit data overview in ‘Decoupled Buffer Access’ mode

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

[FRTP1059] ⌈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).⌋()

[FRTP1060] ⌈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`.」()

[FRTP1061] 「 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.

[FRTP1062] 「The functionality to support data transmission with unknown message length shall be configurable by compiler switch.」()

[FRTP1063] 「 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[19] and the service primitive call FrTp_Transmit shall be rejected with the return value E_NOT_OK.」()

[FRTP1187] 「 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 FRTP1061. 」()

[FRTP1064] 「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') 」()

[FRTP1065] 「During an ongoing data transfer with unknown message length the service primitive parameter `Length` of the service primitive

[19] 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)

PduR_FrTpCopyTxData() shall be set to the value of the currently stored Tx-Buffer's data bytes.」()

[FRTP1066] 「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').」()

[FRTP1067] 「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.

[FRTP1068] 「 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).

Figure 16 depicts the internal processing for data reception in principle[20]. Below there is a description of the different steps which are necessary to receive data via FrTp.

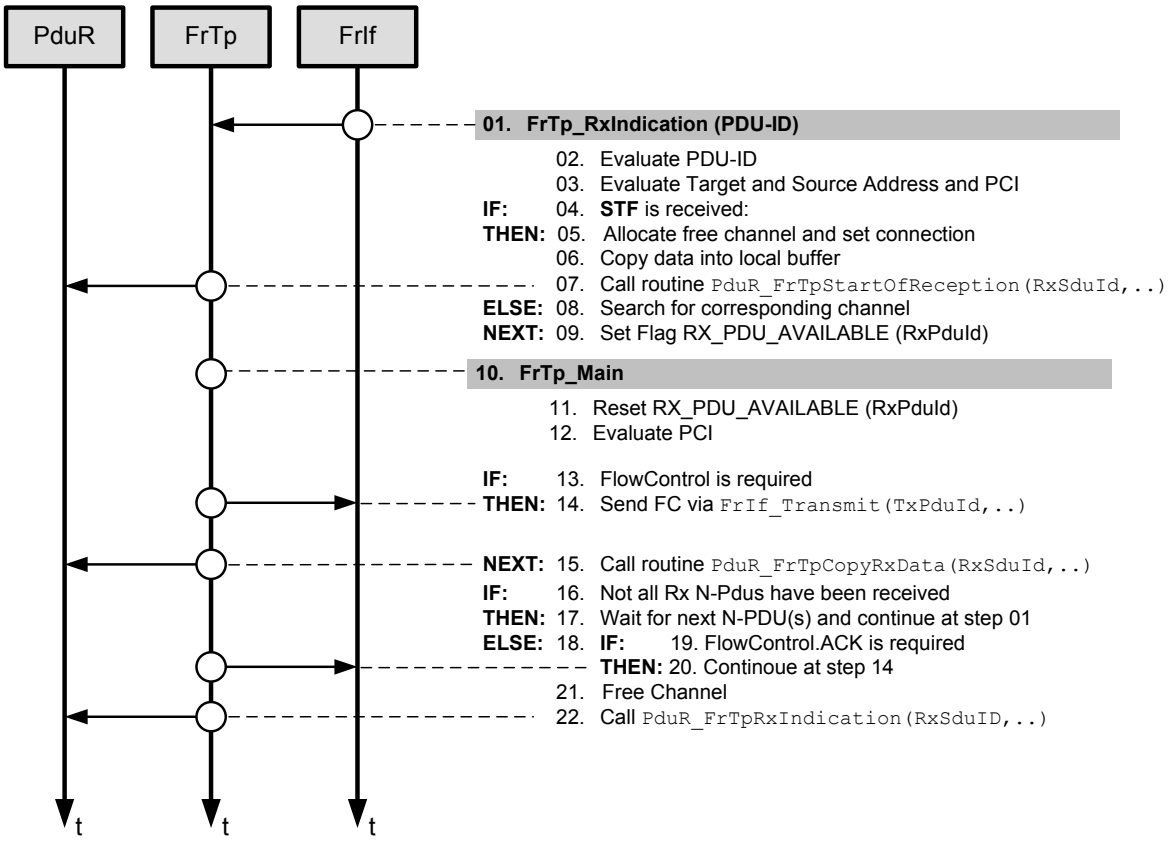


Figure 16: Receive data overview

Step 1 - 9

[FRTP137] ⌈ Receiving shall be initiated by the service primitive call *FrTp_RxIndication.* ⌋()

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

[20] 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 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.

[FRTP1069] ⌈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.⌋()

[FRTP1070] ⌈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.⌋()

[FRTP1071] ⌈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).⌋()

[FRTP1072] ⌈ If the address check doesn't match to any configured FrTpConnection the received FrTp N-PDU shall be ignored.⌋()

[FRTP1074] ⌈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)⌋()

[FRTP1075] ⌈If the current parameter FrTpRxPduId is not supported the service primitive FrTp_RxIndication shall be terminated without any further action.[21]⌋()

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

[FRTP1076] ⌈If no free channel is available (`FrTpRxChannelState ≠ Idle`) the service primitive `FrTp_RxIndication` shall be terminated without any further action.[22] ⌋()

[FRTP1186] ⌈ The FrTp module shall raise an development error FRTP_E_NO_CHANNEL when development error detection for the FrTp module is enabled⌋()

[FRTP1077] ⌈Within the service primitive *`FrTp_RxIndication`* the FrTp module shall copy the received StartFrame PDU into a local buffer[23]. ⌋()

[FRTP1078] ⌈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.⌋()

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.

[FRTP1080] ⌈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

[FRTP1079] ⌈The FrTp module shall extract the N-SDU data from the received N-PDU data according to ISO 10681-2⌋()

[22] It is not possible to signal that temporary resource lack to the upper layer because „PduR_FrTpStartOfReception“ provide no parameter fort hat case.

[23] 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.

[FRTP1138] ⌈The FrTp module shall initiate the copy process of the received N-SDU (fragment) by calling the service primitive PduR_FrTpCopyRxData[24].⌋()

[FRTP421] ⌈ 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.

[FRTP1081] ⌈ 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.⌋()

[FRTP1083] ⌈The FrTp module shall call the service primitive *`PduR_FrTpRxIndication`* if
a) all N-SDU data are received and
b) all required response N-PDUs (FlowControl) are transmitted and TxConfirmation was given.⌋()

7.5.3.1 Receive Cancellation

[FRTP1180] ⌈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.⌋()

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

[FRTP1181] ⌈The FrTp shall abort the reception of the current N-SDU if the service FrTp_CancelReceive provides a valid FrTpRxSduId.⌋()

[FRTP1182] ⌈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.⌋()

[FRTP1183] ⌈If the FrTp_CancelReceive service has been successfully executed the FrTp shall call the PduR_FrTpRxIndication with notification result NTFRSLT_E_CANCELATION_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.

[FRTP1184] ⌈If a data reception with unknown message length shall be established, the FrTp shall call the API PduR_StartOfReception () with an expected data length of zero (“0”).⌋()

Note: If the API PduR_StartOfReception () 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

7.5.4.1 Buffer Access Mode

[FRTP1084] ⌈For Tx direction the FlexRay Transport Protocol Layer shall support
a) “Immediate Buffer Access” mode and
b) “Decoupled Buffer Access” mode.⌋()

7.5.4.2 Request for Buffer

The FrTp module does not provide message buffers, neither for sending nor for receiving. Instead the FrTp module works directly on the memory area of the upper layers (e.g. PduR, DCM, or COM). To access these memory areas, the FrTp module uses the indicator returned by the service primitive *PduR_FrTpCopyTxData* or *PduR_FrTpStartOfReception and PduR_FrTpCopyRxData*.

7.5.4.2.1 Request for RxBuffer

If a reception is being started, there is a difference between the first request for a receive buffer (see FRTP399) and the subsequent requests (see FRTP400).

[FRTP399] ⌈If a reception is being started, the FrTp module shall state the N-SDU length it expects within the request of service primitive PduR_FrTpStartOfReception.⌋()

[FRTP400] ⌈If a reception is being started and a new buffer is requested, the FrTp module shall call the service primitive *PduR_FrTpCopyRxData*.⌋()

If the call for an RxBuffer (service primitive PduR_FrTpStartOfReception or PduR_FrTpCopyRxData) does not provide a valid buffer, the FrTp module shall react as listed below:

[FRTP405] ⌈ If service primitive's PduR_FrTpStartOfReception or PduR_FrTpCopyRxData return BufRequest_ReturnType value as BUFREQ_E_NOT_OK, then FrTp module shall
1) abort reception and
2) send an N-PDU FlowControl.ABT.⌋()

[FRTP1160] ⌈ If service primitive's PduR_FrTpStartOfReception or PduR_FrTpCopyRxData return BufRequest_ReturnType value as BUFREQ_E_BUSY, then FrTp module shall
1) try up to FrTpMaxFcWait (see section 10.2) times to get a valid buffer and
2) shall send an N-PDU FlowControl.WAIT.⌋()

[FRTP1161] ⌈ If service primitive's PduR_FrTpStartOfReception or PduR_FrTpCopyRxData return BufRequest_ReturnType value as BUFREQ_E_OVFL, then FrTp module shall
1) abort reception and
2) shall send an N-PDU FlowControl.OVFL.⌋()

[FRTP1136] ⌈In case of a BufRequest_ReturnType = BUFREQ_E_BUSY a retry shall be performed at the next FrTp_MainFunction invocation.⌋()

[FRTP412] ⌈The FrTp module shall use an N-PDU FlowControl parameter <Buffer Size> depending on the size of the provided RxBuffer (returned by the service primitive call *PduR_FrTpStartOfReception or PduR_CopxRxData)*, to ensure that the transferred data bytes within an upcoming block could be stored into the provided receive buffer.⌋()

7.5.4.2.2 Request for TxBuffer

[FRTP402] ⌈If the call for a TxBuffer (service primitive *PduR_FrTpCopyTxData*) does not provide a valid buffer and if service primitive notification result type value is *BUFREQ_E_BUSY*, then FrTp module shall try up to *FrTpMaxFcWait* times to get a valid buffer.⌋()

[FRTP1162] ⌈If the call for a TxBuffer (service primitive *PduR_FrTpCopyTxData*) does not provide a valid buffer and if service primitive notification result type value is *BUFREQ_E_NOT_OK*, then FrTp module shall abort the transfer.⌋()

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).

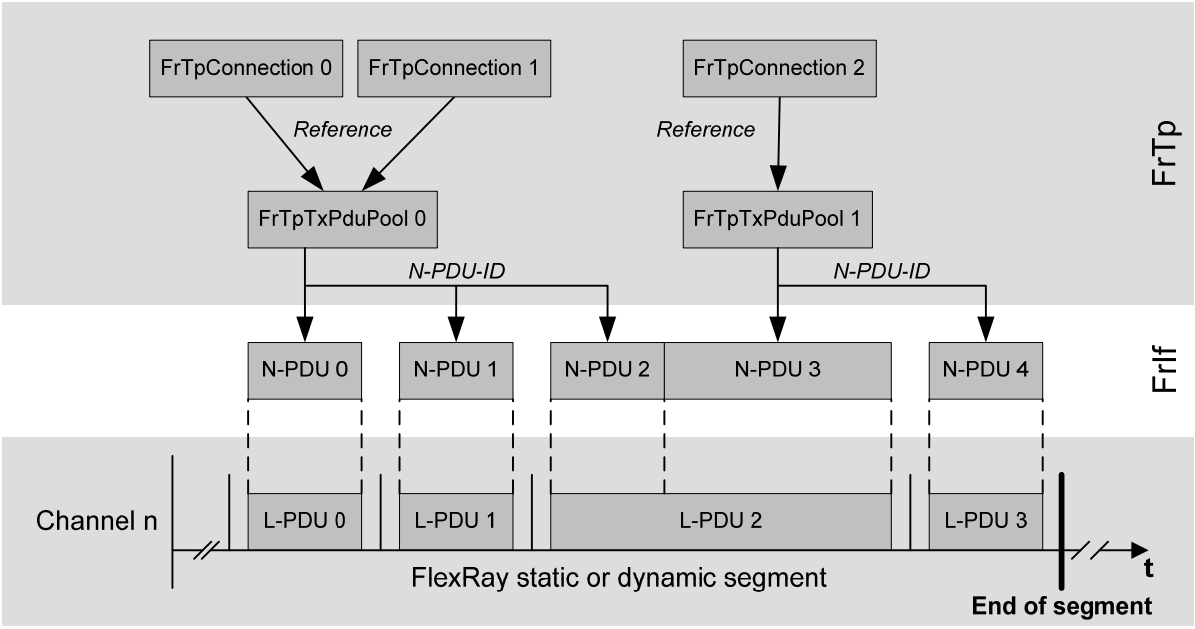


Figure 17: Mapping of N-PDUs to N-PDU-Pools

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 [25]. 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).

[25] 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").

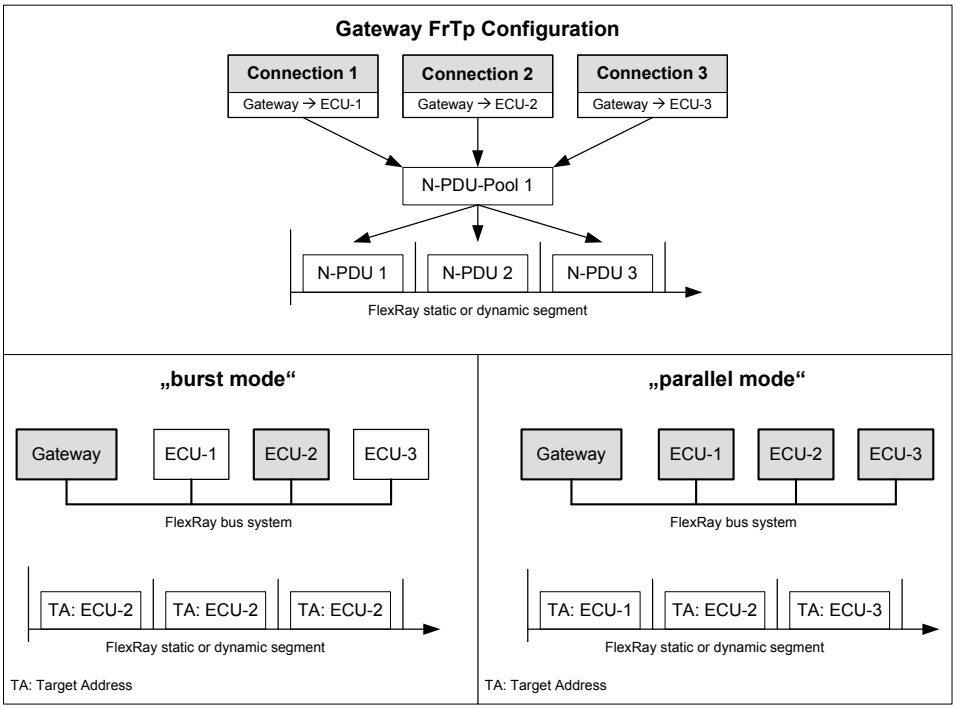


Figure 18: PDU-Pool sharing by different connections

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

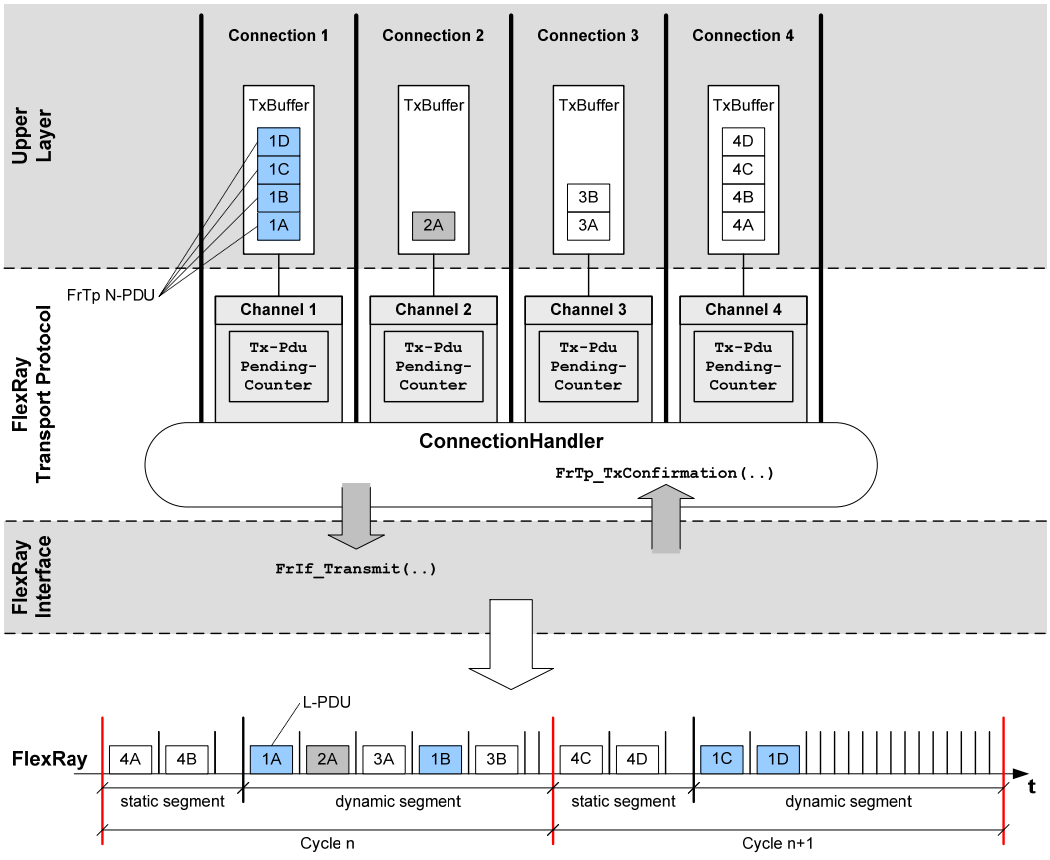


Figure 19: Connection Handler for different connections

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.[26]

[FRTP1088] ⌈ 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[27].⌋()

Note: See also chapter 7.3.2.2

[FRTP1089] ⌈Each FrTp Tx N-PDU within an `FrTpTxPduPool` could have an individual length.[28]⌋()

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.

[FRTP1090] ⌈ Each FrTpChannel shall implement a runtime variable `TxPduPendingCounter` (see chapter 7.3.3.2).⌋()

[FRTP1091] ⌈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`).⌋()

[FRTP1092] ⌈The `TxPduPendingCounter` shall be decremented each time the service primitive `FrTp_TxConfirmation` was called with the corresponding parameter `FrTpTxPduId`.⌋()

[26] 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.

[27] 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.

[28] 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.

[FRTP1093] ⌈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 `FrTpTxPduId`.⌋()

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.

[FRTP1094] ⌈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.⌋ ()

[FRTP1095] ⌈If the `TxPduPendingCouter` is unequal to zero (“0”) the processing for the corresponding `FrTpConnection` shall skipped for the current communication handler task.⌋()

[FRTP1096] ⌈ A communication handler task shall process all active `FrTpConnections` alternately[29] 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 only on sender site. This functionality could disable by a global compiler switch. On receiver side a “Cancellation” shall be handled by timeouts (e.g. not responding a required FlowControl etc.)

[FRTP1097] ⌈The “Transmit Cancellation” feature shall be (de)activated by static configuration of the FrTp parameter *`FrTpTransmitCancellation`* (see section 10.2).⌋()

[FRTP384] ⌈A Transmit Cancellation request shall initiated by the call of the service primitive *`FrTp_CancelTransmit()`* (see FRTP150).⌋()

[29] 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.

[FRTP1116] ⌈A transmit cancellation shall be confirmed by the call of the service primitive *PduR_FrTpTxConfirmation* and the notification result NTFRSLT_E_CANCELLATION_OK. (please refer to [6])⌋()

[FRTP385] ⌈The call of service primitive *FrTp_CancelTransmit* (see FRTP150) shall set the TC_REQUEST flag (see Section 7.5.1.4) of the corresponding connection.⌋()

[FRTP386] ⌈The TC_REQUEST flag (see Section 7.5.1.4) shall be checked at the sender side each time before starting the processing task for an active connection (i.e. *FrTp_MainFunction()* and *FrTp_TriggerTransmit()*).⌋()

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.

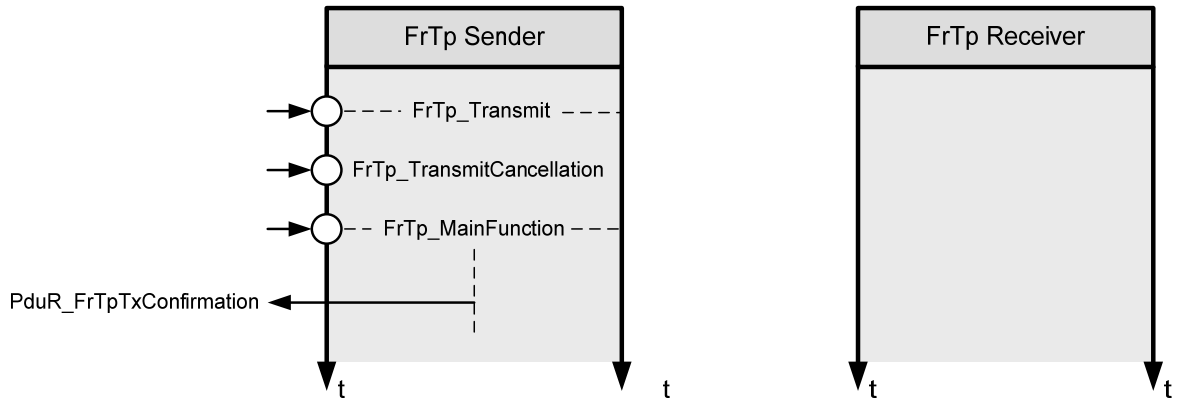


Figure 20: Transmit Cancellation at unsegmented data transfer

If a requested but currently not started data transfer shall be cancelled the service primitive *FrTp_CancelTransmit()* is called and the TC_REQUEST flag is set. The FrTp module's main task shall cancel the requested data transfer and confirm the cancellation by a service primitive *PduR_FrTpTxConfirmation()*call with the corresponding notification result type (see [FRTP1116*)*. 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 and the TC_REQUEST flag is set. The FrTp module's main task shall cancel the requested data transfer and confirm the cancellation by a service primitive *PduR_FrTpTxConfirmation()*call with the corresponding notification result type (see [FRTP1116). On receiver side no data transfer is recognized.

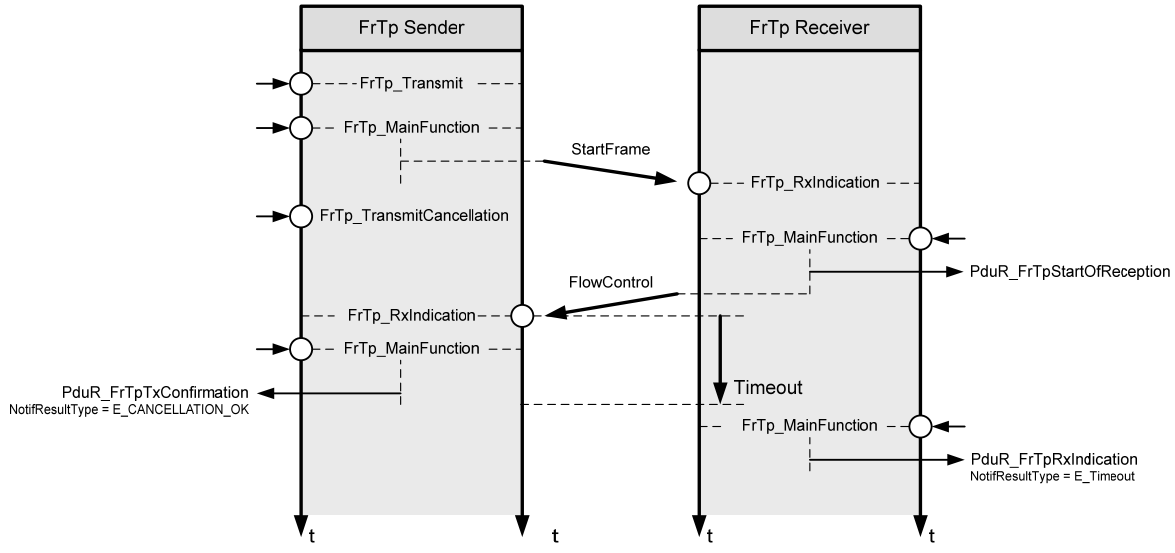


Figure 21: Transmit Cancellation at segmented data transfer

If an ongoing data transfer shall be cancelled, the service primitive *FrTp_CancelTransmit()* is called and the TC_REQUEST flag is set. The FrTp module's main task 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 [30]. After a configured amount of retries the final timeout is used to cancel the current reception by calling the service primitive *PduR_FrTpRxIndication* with the corresponding notification result error code.

7.5.7 Change FrTp Parameter

[FRTP242] ⌈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.⌋()

[FRTP1115] ⌈A change parameter request during an ongoing reception shall be terminated by the service primitive call *PduR_FrTpChangeParameterConfirmation* with result value <NTFRSLT_E_RX_ON> (see chapter 8.2.1.2).⌋()

[FRTP1156] ⌈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.

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

[FRTP1099] 「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.」()

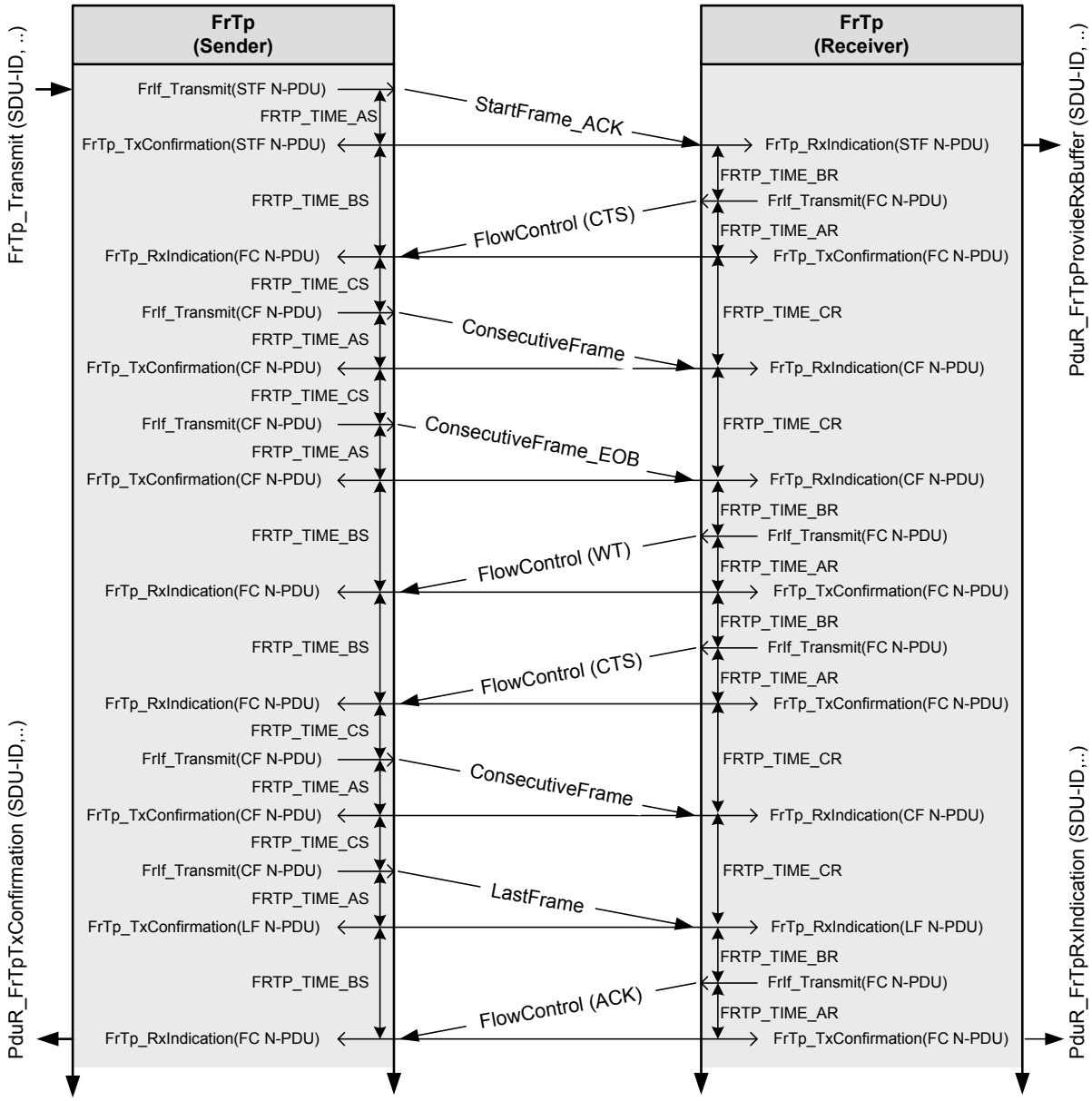


Figure 22: Timing parameter definition for one PDU transmission per Main-Function call

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.

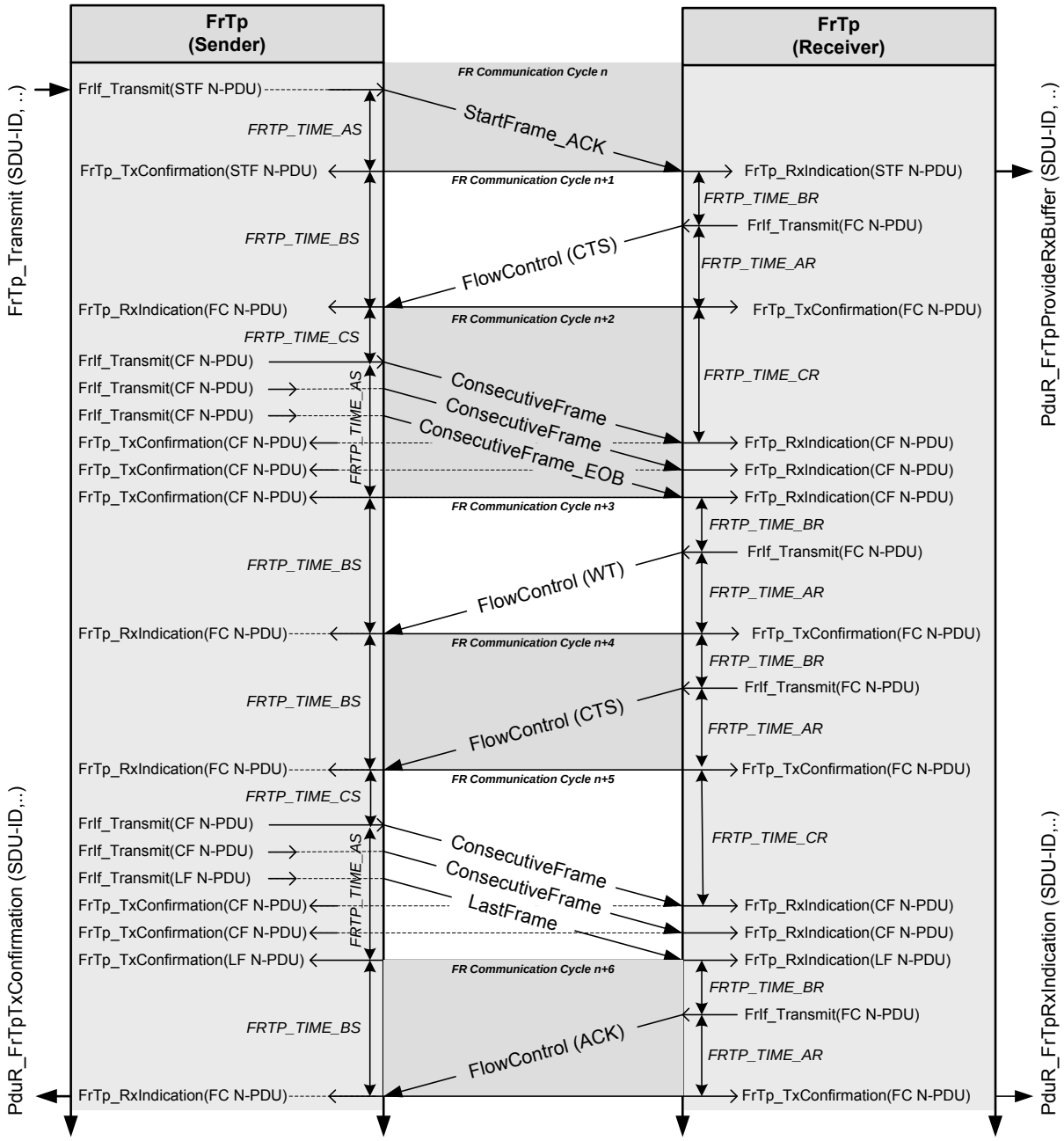


Figure 23: Timing parameter definition for multiple PDU transmission per Main-Function call

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

[FRTP1100] 「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.[31]	Abort message transmission.[32] a) Call FrIf_CancelTransmit() and free the FrTpTxPdu. b) issue PduR_FrTpTxConfirmation with the corresponding FrTpTxSduId and <NotifResultType> = NTFRSLT_E_TIMEOUT_A.
FRTP_TIMEOUT_AR	Any FrTp N-PDU not transmitted in time on the receiver side.[33]	Abort message transmission.[34] a) Call FrIf_CancelTransmit() and free the FrTpTxPdu. b) Issue PduR_FrTpTxConfirmation with the corresponding FrTpTxSduId and <NotifResultType> = NTFRSLT_E_TIMEOUT_A.
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 <NotifResultType> = NTFRSLT_E_TIMEOUT_BS.
FRTP_TIMEOUT_CR	ConsecutiveFrame or Last Frame N-PDU not received (lost, overwritten) on the receiver side.[35]	Abort message reception and issue PduR_FrTpRxIndication with the corresponding FrTpRxSduId and <NotifResultType> = NTFRSLT_E_TIMEOUT_CR.

Table 3: Timeout behaviour for FrTp module

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

32 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.

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

34 See 32

35 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.

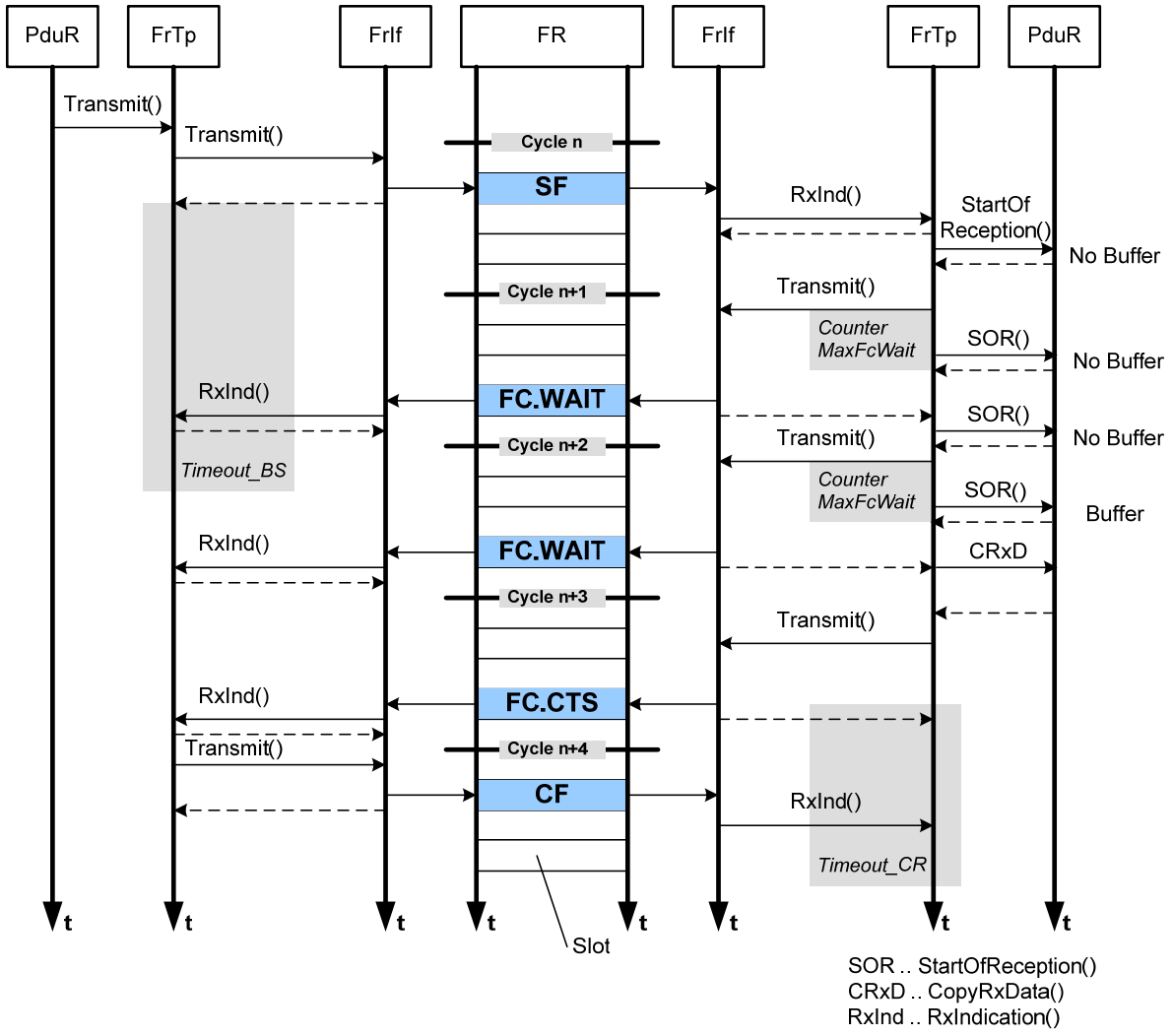


Figure 24: Counter, timer and function call interaction

[FRTP1105] ⌈The FrTp module shall support the counters and corresponding limits as defined in Table 4: FrTp Counter.⌋()

[FRTP1114] ⌈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[36]	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

[FRTP1113] ⌈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 transmission
COUNTER_AS	Abort transmission
COUNTER_RX_RN	**Case a) Segmented – Acknowledged transmission** 1) Abort reception by calling service primitive PduR_FrTpRxIndication with NotifResultType = NTFRSLT_E_WRONG_SN 2) Send FlowControl.ABT (if aFlowControl is possible)

Table 5: FrTp module reaction if counters reached

[36] 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

[FRTP217] ⌈The detection of development errors shall be pre-compile time configurable by the configuration parameter: *FRTP_DEV_ERROR_DETECT*.⌋()

[FRTP1163] ⌈The detection of development errors shall be (de)activated (ON / OFF) by the configuration parameter FRTP_DEV_ERROR_DETECT. ⌋()

[FRTP205] ⌈If the *FRTP_DEV_ERROR_DETECT* switch is enabled API parameter checking shall be enabled. ⌋()

[FRTP218] ⌈It shall be not possible to switch off production code errors detection.⌋ ()

Note: If no production error is currently specified the requirement FRTP218 could be disregarded.

[FRTP1106] ⌈The *FrTpState* shall be checked to detect whether FrTp module is initialized or not.⌋()

7.7.2 Error Notification

[FRTP206] ⌈Detected development errors shall be reported to the Development Error Tracer (DET) if the pre-processor switch *FrTpDevErrorDetect* is set. ⌋()

[FRTP1110] ⌈Production errors shall be reported to Diagnostic Event Manager (DEM) [12].⌋()

Note: If no production error is currently specified the requirement FRTP1110 could be disregarded.

[FRTP1107] ⌈The FrTp module shall use the Development Error Tracer [13] service *Det_ReportError* to report development errors.⌋()

[FRTP1108] ⌈The header file of the FrTp module, `FrTp.h`, shall provide a module Identifier `FRTP_MODULE_ID`.⌋()

[FRTP1109] ⌈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.

[FRTP1111] ⌈The FrTp module shall support the error codes for development errors (Dev.) and production errors (Prod.) as defined in Table 6.⌋()

[FRTP1112] ⌈Event Ids values for production code are assigned externally by the configuration of the DEM module. They are published in the file Dem_IntErrId.h and included via Dem.h.⌋()

[FRTP179] ⌈Development error values are of type uint8.⌋()

[FRTP1132] ⌈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[37]	Dev.	FRTP_E_NO_CHANNEL	0x07
Dev. .. Development Prod. .. Production			

Table 6: Module error classification

[37] 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.

[FRTP141] ⌈*Std_ReturnType* shall be imported from *Std_Types.h.*⌋()

[FRTP1164] ⌈Std_VersionInfoType shall be imported from Std_Types.h.⌋()

[FRTP1165] ⌈BufReq_ReturnType shall be imported from ComStack_Types.h.⌋()

[FRTP1166] ⌈NotifResultType shall be imported from ComStack_Types.h.⌋()

[FRTP1167] ⌈PduIdType shall be imported from ComStack_Types.h.⌋()

[FRTP1168] ⌈PduInfoType shall be imported from ComStack_Types.h.⌋()

[FRTP1169] ⌈PduLengthType shall be imported from ComStack_Types.h.⌋()

[FRTP1170] ⌈RetryInfoType shall be imported from ComStack_Types.h.⌋()

[FRTP1178] ⌈TPParameterType shall be imported from ComStack_Types.h.⌋()

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

8.2 Type definitions

[FRTP1133] ⌈The following FrTp specific types shall be defined in FrTp_Types.h.⌋ (BSW00305)

8.2.1 AUTOSAR Notification Result Types (NotifResultType)

ISO10681-2 [15] and the FrTp module are using several result types to store the result status of a notification (indication or confirmation). For the AUTOSAR Communication Stack a subset of notification result types is defined in specification [7]. This chapter specifies the mapping of ISO10681-2 notification result types to AUTOSAR notification result types.

8.2.1.1 General Return Codes

[FRTP555] ⌈The FrTp module shall support the AUTOSAR Notification Result Types as defined in specification [7].⌋()

[FRTP557] ⌈The general AUTOSAR Notification Result Types as defined in specification [7] shall map to the ISO 10681-2 Notification Result Types as defined in Table 7.⌋()

Table 7: Mapping of AUTOSAR general notification result types to ISO-10681 result types

AUTOSAR - NotifResultCode	ISO10681-2 - C_Result	Value
NTFRSLT_E_ABORT	C_ABORT	Refer to specification [7]
NTFRSLT_E_CANCELATION_NOT_OK	---[38]	Refer to specification [7]
NTFRSLT_E_CANCELATION_OK	C_CANCEL	Refer to specification [7]
NTFRSLT_E_INVALID_FS	C_INVALID_FS	Refer to specification [7]
NTFRSLT_E_NO_BUFFER	C_BUFFER_OVFLW	Refer to specification [7]
NTFRSLT_E_NOT_OK	C_ERROR	Refer to specification [7]
NTFRSLT_E_TIMEOUT_A	C_TIMEOUT_A	Refer to specification [7]
NTFRSLT_E_TIMEOUT_BS	C_TIMEOUT_Bs	Refer to specification [7]
NTFRSLT_E_TIMEOUT_CR	C_TIMEOUT_Cr	Refer to specification [7]
NTFRSLT_E_UNEXP_PDU	C_UNEXP_PDU	Refer to specification [7]
NTFRSLT_E_WFT_OVRN	C_WFT_OVRN	Refer to specification [7]
NTFRSLT_E_WRONG_SN	C_WRONG_SN	Refer to specification [7]
NTFRSLT_OK	C_OK	Refer to specification [7]

[38] Not specified within ISO 10681-2

8.2.1.2 FlexRay Transport Protocol Specific Return Codes

This chapter defines the FlexRay communication stack specific notification result types.

[FRTP142] 「The FrTp module shall support FlexRay specific Notification Result Types as defined in Table 8:」()

Note: Notification Result Types are used to signal results of a data transmission to the PDU Router [6]. A detailed description when a special type has to be used is described within ISO 10681-2 [15].

Table 8: Mapping of AUTOSAR FlexRay specific notification result code to ISO10681-2 result code

Name:	NotifResultType		
Type:	uint8		
Range:	**AUTOSAR – FrTp - NotifResultCode**	**ISO10681-2 - C_Result**	**Value**
	NTFRSLT_E_FR_ML_MISMATCH	C_ML_MISMATCH	0x5B
	NTFRSLT_E_FR_WRONG_BP	C_WRONG_BP	0x5C
	NTFRSLT_E_FR_TX_ON	C_TX_ON	0x5D
Description:	FrTp_ChangeResultType according to ISO10681-2		
	NTFRSLT_E _RX_ON	C_RX_ON	0x0F

[FRTP1121] 「The FrTp module shall not support the notification result types of the ISO document as listed below:
- C_MOREDATA[39]
- C_WRONG_PARAMETER[40]
- C_WRONG_VALUE[41] 」()

8.2.2 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.

Name:	`FrTp_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
Description:	This is the base type for the configuration of the FlexRay Transport Protocol	

[39] ISO functionality for “C_MOREDATA” is supported by the AUTOSAR API “PduR_CopyTxData” or “PduR_CopyRxData”.
[40] When the upper layer calls FrTp_ChangeParameter for change request and if the parameter are invalid, then the API return E_NOT_OK indicating the upper layer about the undefined or invalid parameters, so C_WRONG_PARAMETER and C_WRONG_VALUE are nor used.
[41] See 40

	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

FRTP1137: 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

[FRTP215「
]

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.

」(BSW00407)

[FRTP202] 「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 .

[FRTP498] 「The function FrTp_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: *FrTpVersionInfoApi*」()

[FRTP1150] ⌈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

[FRTP147] ⌈

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).	

⌋(BSW101)

[FRTP1151] ⌈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

[FRTP148] ⌈

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

[FRTP149].⌈

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.

[FRTP1139] ⌈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.⌋()

[FRTP1140] ⌈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

[FRTP150] ⌈

Service name:	FrTp_CancelTransmit	
Syntax:	`Std_ReturnType FrTp_CancelTransmit( PduIdType FrTpTxPduId )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTpTxPduId	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.	

⌋()

[FRTP1141] ⌈ If development error detection for the FrTp_CancelTransmit is enabled: the function FrTp_CancelTransmit shall check the parameter FrTpTxPduId for being valid. If the check for FrTpTxPduId 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:

[FRTP151] ⌈

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.
	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.	

」()

[FRTP1143] 「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.」()

[FRTP1144] 「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

[FRTP1172] 「

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 the function returns, no reception is in progress anymore with the given N-SDU identifier. If a cancellation was performed, the user will be informed by FrTpRxIndication.	

⌋()

8.4 Call-back notifications

8.4.1 FrTp_TriggerTransmit

[FRTP154] ⌈

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 shall be copied to. 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 has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

⌋()

[FRTP1145] ⌈ 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.⌋()

[FRTP1146] ⌈ 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

[FRTP152] ⌈

Service name:	FrTp_RxIndication
Syntax:	`void FrTp_RxIndication( PduIdType RxPduId,`

	`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 module.

」()

[FRTP1147] 「If development error detection for the FrTp_RxIndication is enabled: the function FrTp_RxIndication shall check the parameter FrTpRxPduId for being valid. If the check for FrTpRxPduId fails, the function FrTp_RxIndication shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.」()

[FRTP1148] 「If development error detection for the FrTp_RxIndication is enabled: the function FrTp_RxIndication shall check the parameter FrTpRxPduInfoPtr for being valid. If the check for FrTpRxPduInfoPtr fails, the function FrTpRxPduInfoPtr shall raise the development error FRTP_E_NULL_PTR and return E_NOT_OK.」()

8.4.3 FrTp_TxConfirmation

[FRTP153] 「

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 module confirms the transmission of an I-PDU.

」()

[FRTP1149] 「If development error detection for the FrTp_TxConfirmation is enabled: the function FrTp_TxConfirmation shall check the parameter FrTxPduId for being valid. If the check for FrTxPduId fails, the function

FrTp_TxConfirmation shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.」()

8.5 Scheduled functions

Basic Software Scheduler directly calls these functions.

8.5.1 FrTp_MainFunction

[FRTP203] 「The `FrTp_MainFunction` shall be used to schedule the FrTp module.」()

[FRTP580] 「The `FrTp_MainFunction` shall be the entry point for FrTp processing tasks.」()

[FRTP1152] 「The `FrTp_MainFunction` shall be called at least one time per FlexRay cycle.[42]」()

[FRTP162] 「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
Timing:	FIXED_CYCLIC
Description:	Schedules the FlexRay TP. (Entry point for scheduling)

」()

Timing parameter definitions:

[42] 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.

Fixed cyclic: Fixed cyclic means that one cycle time is defined at configuration and shall not be changed because functionality is requiring that fixed timing (e.g. filters).

Variable cyclic: Variable cyclic means that the cycle times are defined at configuration, but might be mode dependent and therefore vary during runtime.

On pre condition: On pre-condition means that no cycle time could be defined. The function is if conditions are fulfilled. Alternatively, the function may be called cyclically however the cycle time will be assigned dynamically during runtime by other modules.

8.6 Expected Interfaces

This chapter describes all expected APIs from other modules.

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

[FRTP577] 「The FrTp module expects service primitives from the PDU Router as listed in Table 9.」()

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 9: Required PDU Router service primitives

8.6.1.2 FlexRay Interface

[FRTP578] 「The FrTp module expects service primitives from the FlexRay Interface as listed in Table 10.」()

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 10: Required FlexRay Interface service primitives

8.6.2 Optional Interfaces

8.6.2.1 Debug Error Tracer Interface

[FRTP579] 「Depending on the configuration parameter `FrTpDevErrorDetect`, the FrTp module expects service primitives from Debug Error Tracer module as listed in Table 10.」()

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

Table 11: Required Debug Error Tracer service primitives

8.6.2.2 PduRouter Interface

[FRTP1188] 「Depending on the configuration parameter `FRTP_CHANGE_PARAMETER_API`, the FrTp module expects service primitives from PDURouter module as listed in Table 12.」()

API service primitive	Description
PduR_FrTpChangeParameterConfirmation	By this API service primitive, the FrTp module confirms the (un)successful change of module parameter(s).

Table 12: Required PDU Router service primitives

8.6.3 Configurable interfaces

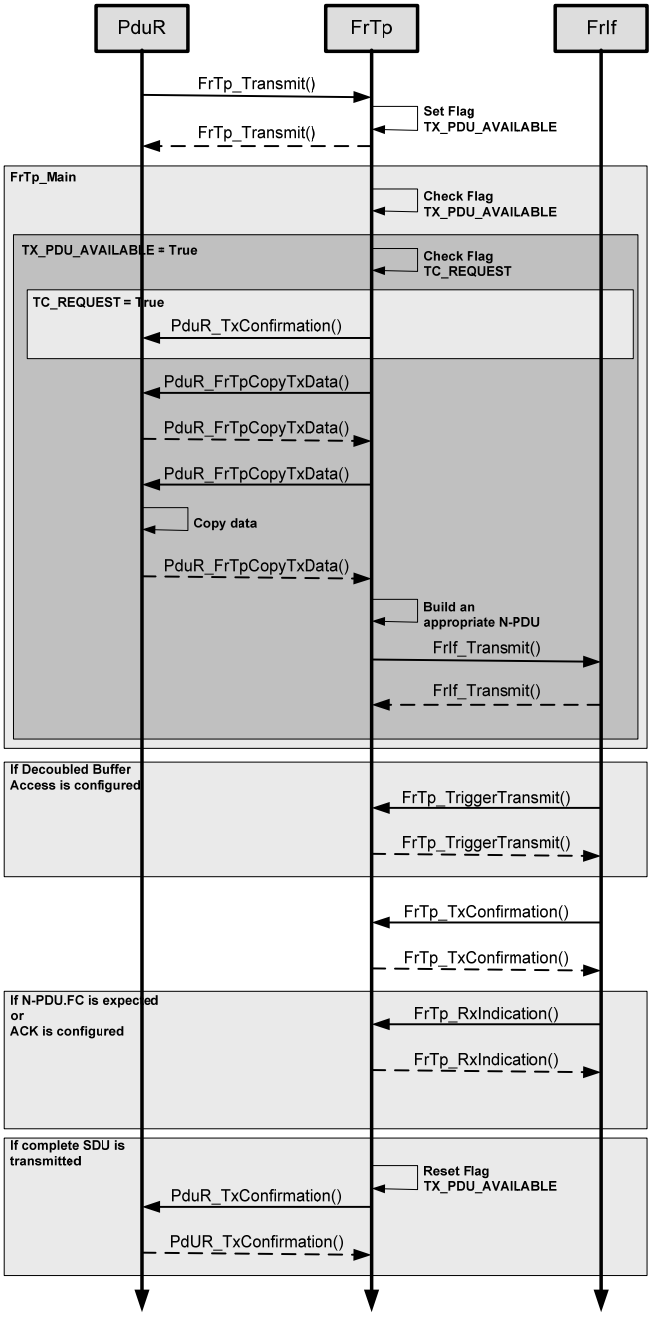
No interfaces are defined.

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.

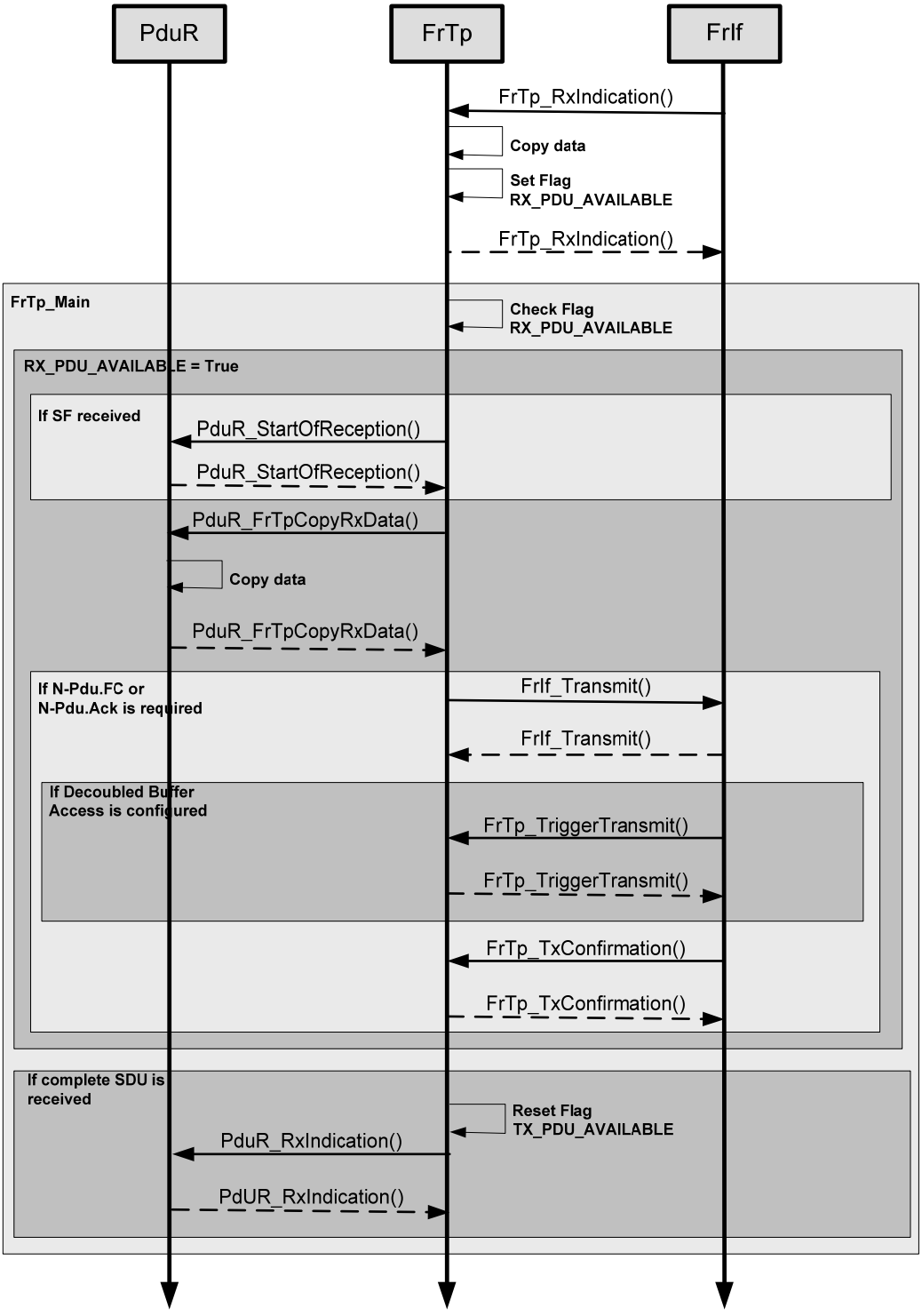
9.1 Sending of N-Pdus

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.



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.



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.

[FRTP569] ⌈The configuration tool should extract all information to configure the FlexRay Transport Protocol.⌋()

[FRTP570] ⌈The configuration tool shall check the configuration consistency at configuration time. Configuration rules and constraints for plausibility checks shall performed where ever possible, during configuration time.⌋()

10.1 How to read this chapter

In addition to this section, it is highly recommended to read the documents:

- AUTOSAR Layered Software Architecture [2].
  This document describes the AUTOSAR layered software architecture and shows the involved software modules.

- AUTOSAR Specification of Communication Stack Types [7].
  This document describes the basic data types for the communication stack. This data types are used in the configuration parameter containers.

- AUTOSAR ECU Configuration Specification [11].
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration meta model in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of module implementation. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times within the software generation process:

- before compile time,
- before link time,
- after build time.

In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

10.1.2 Variants

Variants describe sets of configuration parameters. E.g. variant 1: only pre-compile time configuration parameters, variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant, a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The configuration parameter containers consist of three sections:

- the general section
  includes general information of a container
- the configuration parameter section
  includes the configuration information of the specific parameter of a container
- the section of included containers
  describes the sub-containers which are also part of that (parent) container.

Table 13 depicts a detailed description of the different container items.

General Section			
SWS Item			
Container Name	Identifies the container by name		
Description	Explains the intention and content of the container.		
Configuration Parameters			
Name	Identifies the parameter by name.		
Description	Explains the intention of the configuration parameter.		
Type or Unit	Specifies the type of parameter (e.g., `uint8..uint32`) or specifies the unit of the parameter (e.g., `ms`)		
Range	Specifies the range (or possible values) of the parameter (e.g., `1..15`, `ON,OFF`)		Description(s) of the value(s) or range(s).
Configuration Class	***Pre-compile***	See description below.	Refer here to (a) variant(s).
	Link time	See description below.	Refer here to (a) variant(s).
	Post Build	See description below.	Refer here to (a) variant(s).
Scope	The scope describes the impact of the configuration parameter: Does the setting affect only one instance of the module (instance), all instances of this module (module), the ECU or a network? *Possible values of scope: instance, module, ECU, network*		
Dependency	*Describes the dependencies to other parameters, containers or global configurations.*		

Included Container(s)		
Container Name	***Multiplicity***	***Scope / Dependency***
Reference a valid (sub)container by its name.	Specifies the number of possible instances of the referenced container.	The scope describes the impact of the configuration parameter: Does the setting affect only one instance of the module (instance), all instances of this module (module), the ECU or a network?

Table 13: Description of the container items

Pre-compile time:
This item specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not.

Label	***Description***
x	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time:
specifies whether the configuration parameter shall be of configuration class Link time or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build:
specifies whether the configuration parameter shall be of configuration class Post Build or not

Label	*Description*
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

10.2 Containers and configuration parameters

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

10.2.1 Variants

[FRTP1131]「The FrTp module shall support configuration variants as listed below:

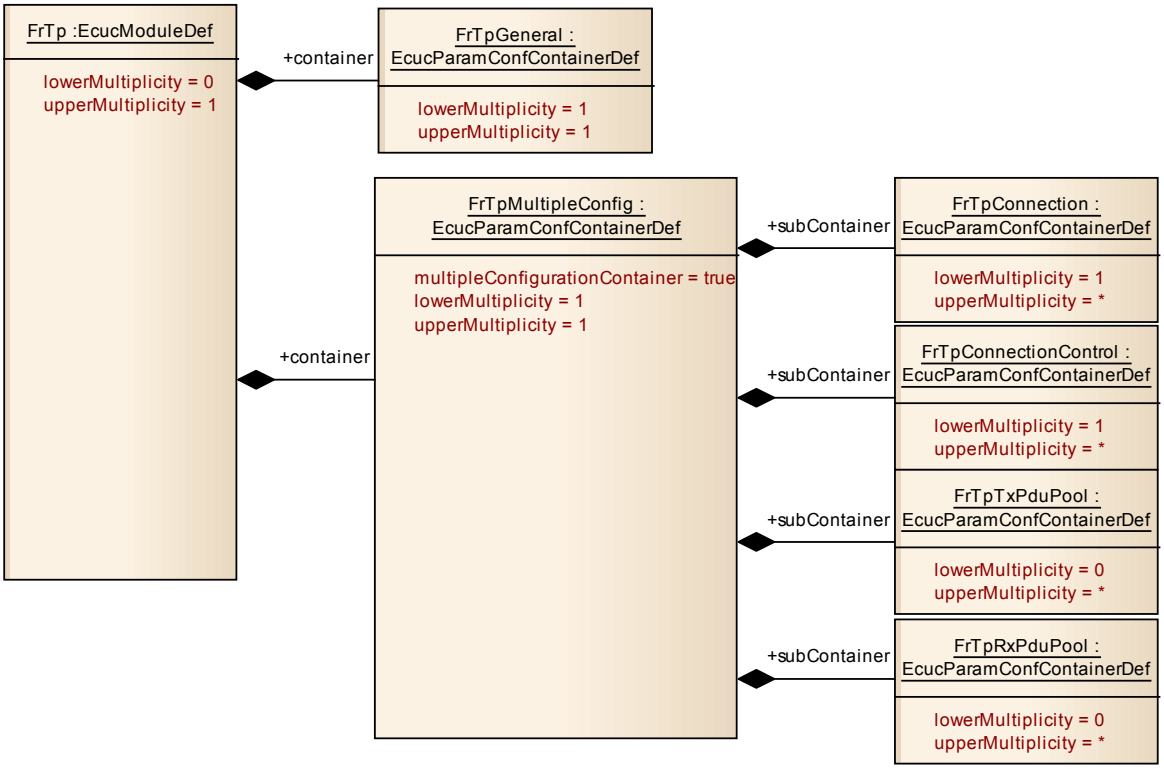
VARIANT-PRE-COMPILE	Only parameters with "Pre-compile time" configuration are allowed in this variant.
VARIANT-POST-BUILD	Parameters with "Pre-compile time" and "Post-build time" are allowed in this variant.

」()

10.2.2 FrTp

SWS Item	**FRTP001_Conf :**
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	**FRTP009_Conf :**
Container Name	FrTpGeneral
Description	This container contains the general configuration parameters of the FlexRay Transport Protocol module.
Configuration Parameters	

SWS Item	**FRTP002_Conf :**		
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: Module		

SWS Item	**FRTP004_Conf :**		
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: Module		

SWS Item	**FRTP052_Conf :**		
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: Module		

SWS Item	**FRTP008_Conf :**		
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: Module		

SWS Item	**FRTP051_Conf :**		
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: Module		

SWS Item	**FRTP011_Conf :**		
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: Module		

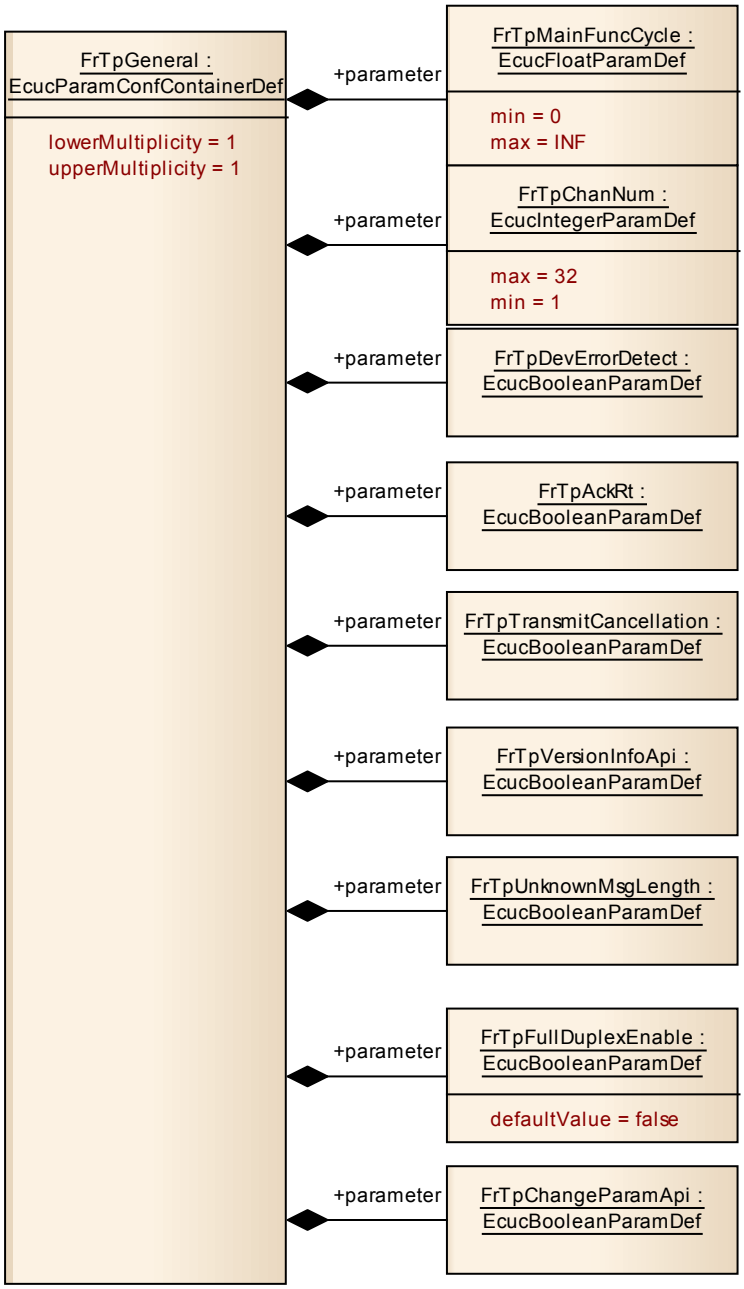
SWS Item	**FRTP036_Conf :**

Name	FrTpTransmitCancellation {FRTP_HAVE_TC}		
Description	Preprocessor switch for enabling Transmit Cancellation. True: Transmit Cancellation is enabled False: Transmit Cancellation is disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FRTP044_Conf :**		
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: Module		

SWS Item	**FRTP045_Conf :**		
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: Module		

No Included Containers



10.2.4 FrTpMultipleConfig

SWS Item	**FRTP018_Conf :**
Container Name	FrTpMultipleConfig [Multi Config Container]
Description	This container holds one or several multiple configuration sets.
Configuration Parameters	

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	**FRTP006_Conf :**
Container Name	FrTpConnection{FrTpConnectionConfiguration}
Description	This container contains the connection specific parameters to transfer N-PDUs via FlexRay TP.
Configuration Parameters	

SWS Item	**FRTP050_Conf :**		
Name	FrTpBandwidthLimitation {FRTP_BW_LIMIT}		
Description	This parameter indicates wheather 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: Module		

SWS Item	**FRTP010_Conf :**		
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: Module		

SWS Item	**FRTP019_Conf :**		
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: Module		

SWS Item	**FRTP021_Conf :**

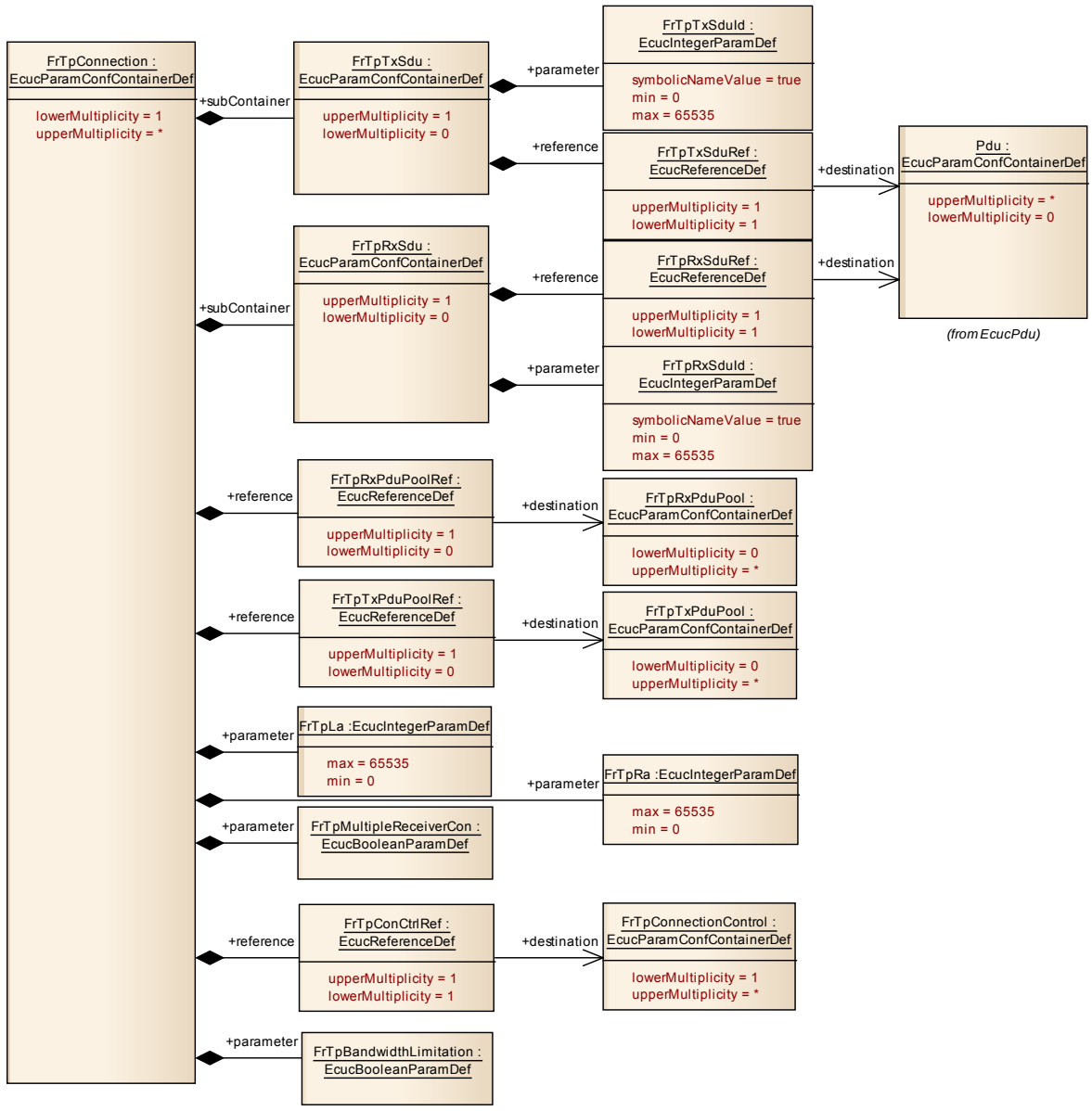
Name	FrTpRa {FRTP_RA}		
Description	This parameter defines the Remote Address for the respective connection. When the local instance is the sender, this is the Target Address within the TP frame. When the local instance is the receiver, this is the Source Address within the TP frame.		
Multiplicity	1		
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: Module		

SWS Item	**FRTP005_Conf :**		
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: Module		

SWS Item	**FRTP025_Conf :**		
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: Module		

SWS Item	**FRTP039_Conf :**		
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: Module		

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	**FRTP041_Conf :**
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	**FRTP042_Conf :**	
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	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FRTP043_Conf :**		
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	**FRTP027_Conf :**
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	**FRTP053_Conf :**		
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	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FRTP028_Conf :**		
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	**FRTP007_Conf :**
Container Name	FrTpConnectionControl{FrTpConnectionControl}
Description	This container contains the configuration parameters to control a FlexRay TP connection.
Configuration Parameters	

SWS Item	**FRTP003_Conf :**		
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: Module		

SWS Item	**FRTP012_Conf :**		
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: Module		

SWS Item	**FRTP013_Conf :**		
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: Module		

SWS Item	**FRTP015_Conf :**		
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: Module		

SWS Item	**FRTP014_Conf :**		
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: Module		

SWS Item	**FRTP016_Conf :**		
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: Module		

SWS Item	**FRTP029_Conf :**		
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: Module		

SWS Item	**FRTP017_Conf :**		
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: Module

SWS Item	**FRTP020_Conf :**		
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: Module		

SWS Item	**FRTP047_Conf :**		
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: Module		

SWS Item	**FRTP030_Conf :**		
Name	FrTpTimeBuffer {FRTP_TIME_BUFFER}		
Description	This parameter defines the time in seconds of waiting for the next try to get a Tx or Rx buffer.		
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: Module		

SWS Item	**FRTP031_Conf :**		
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: Module		

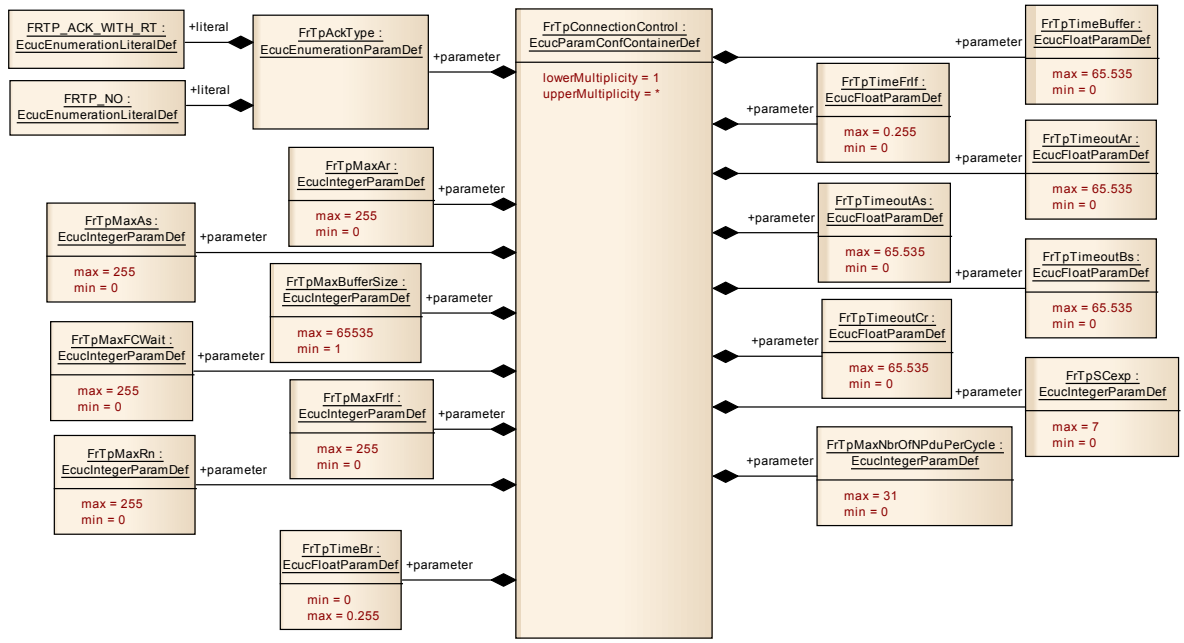
SWS Item	**FRTP032_Conf :**		
Name	FrTpTimeoutAr {FRTP_TIMEOUT_AR}		
Description	This parameter states the timeout in seconds between the PDU transmit request of the Transport Layer to the FlexRay Interface and the corresponding confirmation of the FlexRay Interface on the receiver side (for FC or AF).		
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: Module 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	**FRTP033_Conf :**		
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: Module 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	**FRTP034_Conf :**		
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: Module 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	**FRTP035_Conf :**		
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: Module 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

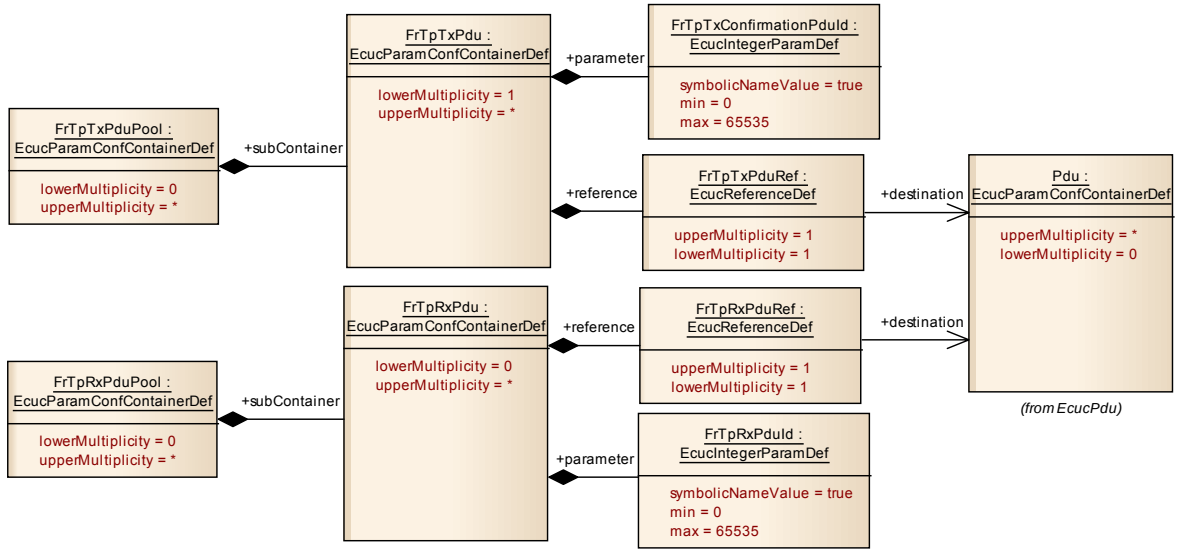
SWS Item	**FRTP038_Conf :**
Container Name	FrTpTxPduPool{PduPoolContainer}
Description	This container contains all Pdus that are assigned to that Pdu Pool.
Configuration Parameters	

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

10.2.10 FrTpRxPduPool

SWS Item	**FRTP024_Conf :**
Container Name	FrTpRxPduPool{PduPoolContainer}
Description	This container contains all Pdus that are assigned to that Pdu Pool.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrTpRxPdu	0..*	Container to hold the PDU parameters. ImplementationType: PduInfoType



10.2.11 FrTpTxPdu

SWS Item	**FRTP037_Conf :**
Container Name	FrTpTxPdu
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType
Configuration Parameters	

SWS Item	**FRTP049_Conf :**		
Name	FrTpTxConfirmationPduId		
Description	Handle Id to be used by the FrIf to confirm the transmission of the FrTpTxPdu to the FrIf module.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
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: Module		

SWS Item	**FRTP040_Conf :**
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			

No Included Containers

10.2.12 FrTpRxPdu

SWS Item	**FRTP022_Conf :**
Container Name	FrTpRxPdu
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType
Configuration Parameters	

SWS Item	**FRTP023_Conf :**		
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	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FRTP026_Conf :**		
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

[FRTP1192] ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules [3] shall be published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1].⌋()

Additional module-specific published parameters are listed below if applicable.

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 25 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.

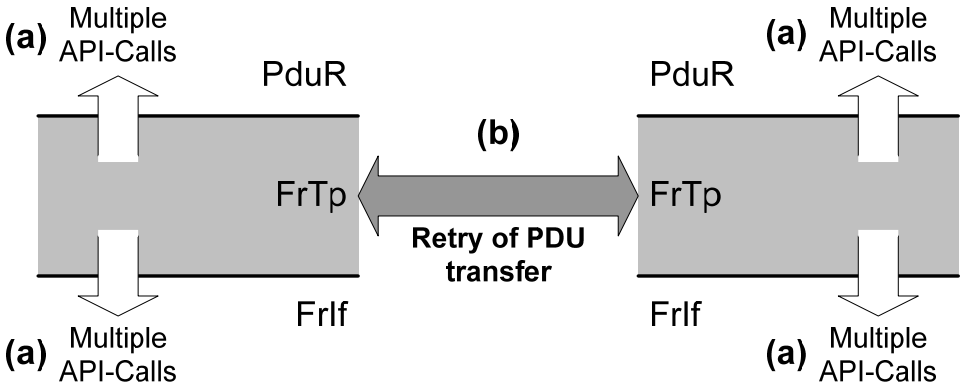


Figure 25: FrTp Retry Scenarios

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.

[FRTP598] ⌈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*

[FRTP1154] ⌈For timeout As configuration it shall be considered that

FRTP_TIMEOUT_AS > FRTP_TIME_AS ⌋()

[FRTP1155] ⌈For timeout Ar configuration it shall be considered that

FRTP_TIMEOUT_AR > FRTP_TIME_AR ⌋()

[FRTP599] ⌈For timeout Bs configuration it shall be considered that

FRTP_TIMEOUT_BS > FRTP_TIME_BR + FRTP_TIME_AR ⌋()

[FRTP1153] ⌈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 [15]

[FRTP180] ⌈ The FrTp configuration shall ensure that $2^{SeparationCycleExponent}-1 \times t_{CycleTime} \leq$ FRTP_TIMEOUT_CR .[43] ⌋()

10.4.4 Bandwidth Control Configuration

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

It could occur that an ECU is not able to receive as much PDUs as defined within the PDU-Pool referenced by a connection[44]. In that case the bandwidth has to be limited by the receiver. There are three possibilities to limit the bandwidth for a connection link:

a) Limit Bandwidth by Parameter FlowControl.BandwidthControl.MaxPduPerCycle in combination with Hardware FIFO buffer mechanisms.[45]
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.

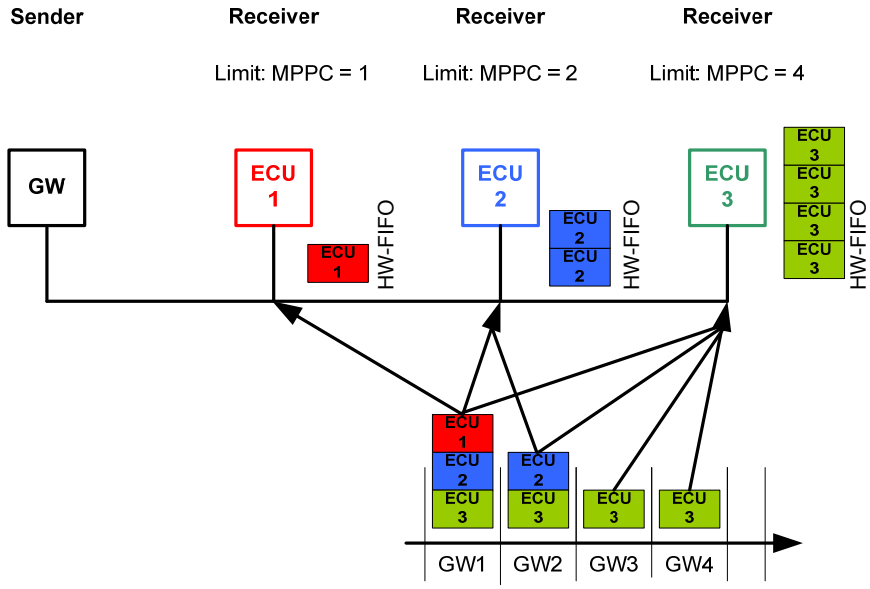


Figure 26: BandwidthControl by FlowControl Parameters in combination with HW FIFO buffer

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.

[44] 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.
[45] This is an esential mechanism of FlexRay 3.0 protocol.

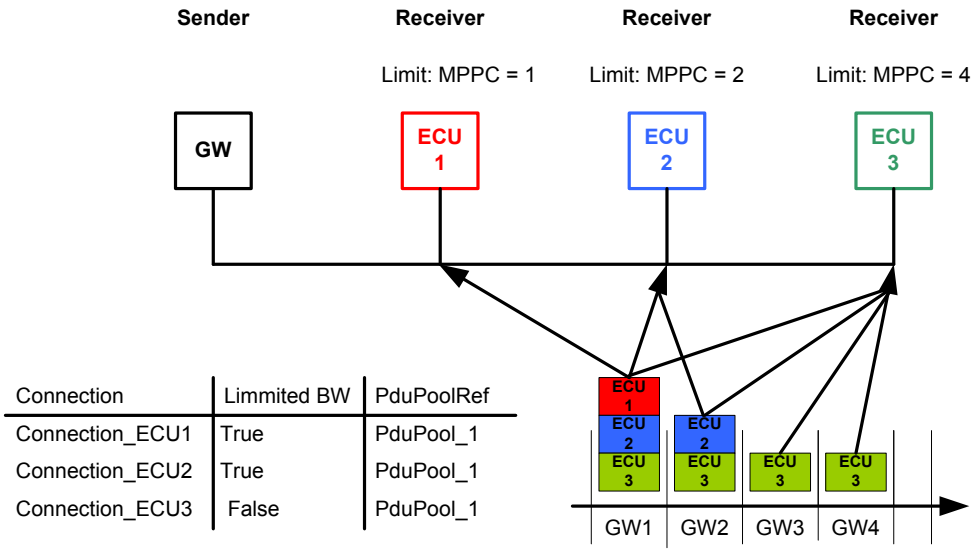


Figure 27: BandwidthControl by FlowControl Parameters

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:

[FRTP1173] ⌈ 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“.⌋()

[FRTP1174] ⌈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.⌋()

[FRTP1175] ⌈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.⌋()

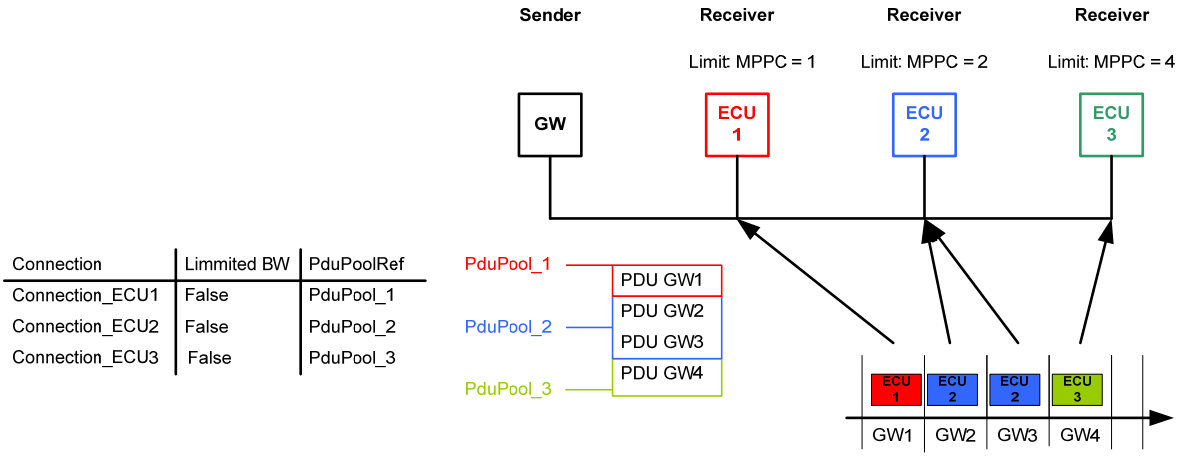
[FRTP1177] ⌈If the attribute “FrTpBandwidthLimitation“ is set to "TRUE", a Rx-connection shall use the first Pdu of the referenced Tx-Pdu-Pool for sending the required FlowControl frame to continue a communication link.⌋()

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:

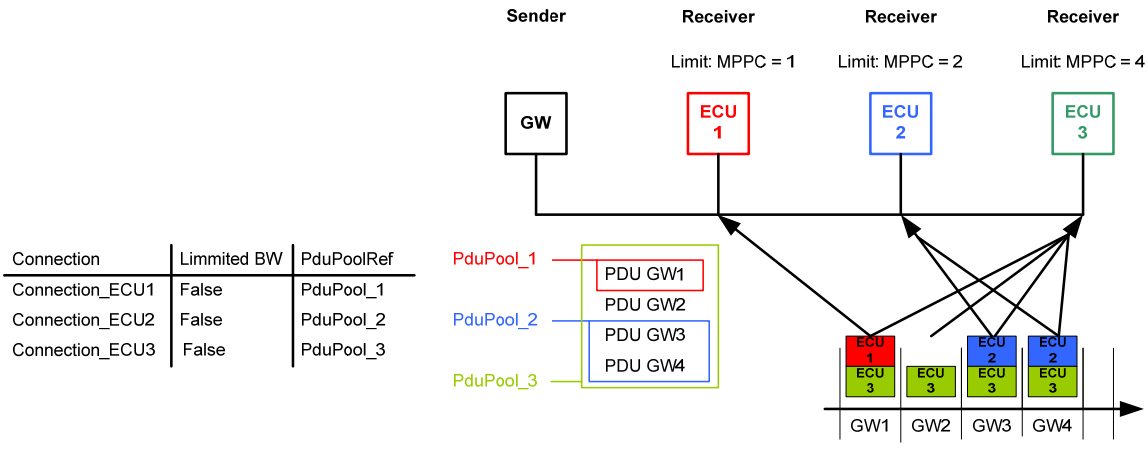


Connection	Limmited BW	PduPoolRef
Connection_ECU1	False	PduPool_1
Connection_ECU2	False	PduPool_2
Connection_ECU3	False	PduPool_3

Figure 28: 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 29: 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[46].

[FRTP1176] 「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.

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

11 Changes from Release 3.x to 4.0 (Support ISO 10681-2)

11.1 Deleted SWS Items

SWS Item	*Rationale*
FRTP010	Requirement is specified within ISO 10681-2
FRTP012	Requirement is specified within ISO 10681-2
FRTP014	Requirement is specified within ISO 10681-2
FRTP016	Requirement is specified within ISO 10681-2
FRTP017	Requirement is specified within ISO 10681-2
FRTP019	Requirement is specified within ISO 10681-2
FRTP020	Requirement is specified within ISO 10681-2
FRTP023	Requirement is specified within ISO 10681-2
FRTP024	Requirement is specified within ISO 10681-2
FRTP025	Requirement is specified within ISO 10681-2
FRTP026	Requirement is specified within ISO 10681-2
FRTP029	Requirement is specified within ISO 10681-2
FRTP030	Requirement is specified within ISO 10681-2
FRTP031	Requirement is specified within ISO 10681-2
FRTP036	Requirement is specified within ISO 10681-2
FRTP037	Requirement is specified within ISO 10681-2
FRTP038	Requirement is specified within ISO 10681-2
FRTP055	Requirement is specified within ISO 10681-2
FRTP056	Requirement is specified within ISO 10681-2
FRTP057	Requirement is specified within ISO 10681-2
FRTP058	Requirement is specified within ISO 10681-2
FRTP060	Requirement is specified within ISO 10681-2
FRTP061	Requirement is specified within ISO 10681-2
FRTP064	Requirement is specified within ISO 10681-2
FRTP065	Requirement is specified within ISO 10681-2
FRTP066	Requirement is specified within ISO 10681-2
FRTP067	Requirement is specified within ISO 10681-2
FRTP070	Requirement is specified within ISO 10681-2
FRTP075	Requirement is specified within ISO 10681-2
FRTP079	Requirement is specified within ISO 10681-2
FRTP082	Requirement is specified within ISO 10681-2
FRTP083	Requirement is specified within ISO 10681-2
FRTP085	Requirement is specified within ISO 10681-2
FRTP086	Requirement is specified within ISO 10681-2
FRTP087	Requirement is specified within ISO 10681-2
FRTP092	Requirement ID from Type deleted
FRTP093	Requirement ID from Type deleted
FRTP094	Requirement ID from Type deleted
FRTP102	Requirement ID from Type deleted
FRTP103	Requirement ID from Type deleted
FRTP105	Requirement ID from Type deleted
FRTP106	Requirement is specified within ISO 10681
FRTP108	Requirement is specified within ISO 10681
FRTP110	Requirement is specified within ISO 10681
FRTP112	Requirement is specified within ISO 10681
FRTP113	Requirement ID from Type deleted
FRTP114	Requirement ID from Type deleted
FRTP116	Requirement ID from Type deleted
FRTP117	Requirement ID from Type deleted
FRTP120	Requirement ID from Type deleted

FRTP121	Requirement ID from Type deleted
FRTP123	Requirement ID from Type deleted
FRTP124	Requirement ID from Type deleted
FRTP125	Requirement ID from Type deleted
FRTP126	Requirement ID from Type deleted
FRTP127	Requirement ID from Type deleted
FRTP128	Requirement ID from Type deleted
FRTP129	Requirement ID from Type deleted
FRTP130	Requirement ID from Type deleted
FRTP138	Requirement ID from Type deleted
FRTP139	Requirement ID from Type deleted
FRTP140	Requirement ID from Type deleted
FRTP143	Requirement ID deleted after MS-2 review
FRTP145	Requirement ID deleted after MS-4 review
FRTP169	Requirement ID from Type deleted
FRTP170	Requirement ID from Type deleted
FRTP181	Requirement ID from Type deleted
FRTP189	Requirement ID from Type deleted
FRTP190	Requirement ID from Type deleted
FRTP194	Requirement ID deleted after MS-2 review
FRTP207	Requirement removed
FRTP213	Requirement was specified twice
FRTP225	Requirement is specified within ISO 10681
FRTP226	Requirement is specified within ISO 10681
FRTP227	Requirement is specified within ISO 10681
FRTP229	Requirement is specified within ISO 10681
FRTP230	Requirement is specified within ISO 10681
FRTP237	Requirement is specified within ISO 10681
FRTP238	Requirement is specified within ISO 10681
FRTP239	Requirement is specified within ISO 10681
FRTP241	Requirement is specified within ISO 10681
FRTP245	Requirement is specified within ISO 10681
FRTP246	Requirement is specified within ISO 10681
FRTP247	Requirement is specified within ISO 10681
FRTP248	Requirement is specified within ISO 10681
FRTP250	Requirement is specified within ISO 10681
FRTP252	Requirement is specified within ISO 10681
FRTP253	Requirement is specified within ISO 10681
FRTP254	Requirement is specified within ISO 10681
FRTP255	Requirement is specified within ISO 10681
FRTP256	Requirement is specified within ISO 10681
FRTP257	Requirement is specified within ISO 10681
FRTP258	Requirement is specified within ISO 10681
FRTP259	Requirement is specified within ISO 10681
FRTP261	Requirement is specified within ISO 10681
FRTP262	Requirement is specified within ISO 10681
FRTP263	Requirement is specified within ISO 10681
FRTP264	Requirement is specified within ISO 10681
FRTP265	Requirement is specified within ISO 10681
FRTP266	Requirement is specified within ISO 10681
FRTP267	Requirement is specified within ISO 10681
FRTP268	Requirement is specified within ISO 10681
FRTP269	Requirement is specified within ISO 10681
FRTP270	Requirement is specified within ISO 10681
FRTP271	Requirement is specified within ISO 10681
FRTP272	Requirement is specified within ISO 10681
FRTP273	Requirement is specified within ISO 10681
FRTP274	Requirement is specified within ISO 10681

FRTP275	Requirement is specified within ISO 10681
FRTP276	Requirement is specified within ISO 10681
FRTP277	Requirement is specified within ISO 10681
FRTP278	Requirement is specified within ISO 10681
FRTP279	Requirement is specified within ISO 10681
FRTP280	Requirement is specified within ISO 10681
FRTP281	Requirement is specified within ISO 10681
FRTP282	Requirement is specified within ISO 10681
FRTP283	Requirement is specified within ISO 10681
FRTP284	Requirement is specified within ISO 10681
FRTP285	Requirement is specified within ISO 10681
FRTP286	Requirement is specified within ISO 10681
FRTP287	Requirement is specified within ISO 10681
FRTP288	Requirement is specified within ISO 10681
FRTP289	Requirement is specified within ISO 10681
FRTP296	Requirement is specified within ISO 10681
FRTP297	Requirement is specified within ISO 10681
FRTP298	Requirement is specified within ISO 10681
FRTP299	Requirement is specified within ISO 10681
FRTP300	Requirement is specified within ISO 10681
FRTP301	Requirement is specified within ISO 10681
FRTP302	Requirement is specified within ISO 10681
FRTP303	Requirement is specified within ISO 10681
FRTP304	Requirement is specified within ISO 10681
FRTP305	Requirement is specified within ISO 10681
FRTP306	Requirement is specified within ISO 10681
FRTP307	Requirement is specified within ISO 10681
FRTP308	Requirement is specified within ISO 10681
FRTP312	Requirement is specified within ISO 10681
FRTP313	Requirement is specified within ISO 10681
FRTP314	Requirement is specified within ISO 10681
FRTP315	Requirement is specified within ISO 10681
FRTP316	Requirement is specified within ISO 10681
FRTP317	Requirement is specified within ISO 10681
FRTP318	Requirement is specified within ISO 10681
FRTP319	Requirement is specified within ISO 10681
FRTP320	Requirement is specified within ISO 10681
FRTP321	Requirement is specified within ISO 10681
FRTP322	Requirement is specified within ISO 10681
FRTP323	Requirement is specified within ISO 10681
FRTP325	Requirement is specified within ISO 10681
FRTP326	Requirement is specified within ISO 10681
FRTP327	Requirement is specified within ISO 10681
FRTP328	Requirement is specified within ISO 10681
FRTP329	Requirement is specified within ISO 10681
FRTP330	Requirement is specified within ISO 10681
FRTP331	Requirement is specified within ISO 10681
FRTP332	Requirement is specified within ISO 10681
FRTP333	Requirement is specified within ISO 10681
FRTP334	Requirement is specified within ISO 10681
FRTP335	Requirement is specified within ISO 10681
FRTP336	Requirement is specified within ISO 10681
FRTP337	Requirement is specified within ISO 10681
FRTP338	Requirement is specified within ISO 10681
FRTP339	Requirement is specified within ISO 10681
FRTP342	Requirement is specified within ISO 10681
FRTP343	Requirement is specified within ISO 10681
FRTP344	Requirement is specified within ISO 10681

FRTP345	Requirement is specified within ISO 10681
FRTP353	Requirement is specified within ISO 10681
FRTP354	Requirement is specified within ISO 10681
FRTP356	Requirement is specified within ISO 10681
FRTP360	Requirement is specified within ISO 10681
FRTP361	Requirement is specified within ISO 10681
FRTP363	Requirement is specified within ISO 10681
FRTP364	Requirement is specified within ISO 10681
FRTP370	Requirement is specified within ISO 10681
FRTP371	Requirement is specified within ISO 10681
FRTP372	Requirement is specified within ISO 10681
FRTP373	Requirement is specified within ISO 10681
FRTP374	Requirement is specified within ISO 10681
FRTP375	Requirement is specified within ISO 10681
FRTP376	Requirement is specified within ISO 10681
FRTP378	Requirement is specified within ISO 10681
FRTP377	Requirement is specified within ISO 10681
FRTP379	Requirement is specified within ISO 10681
FRTP387	Requirement is specified within ISO 10681
FRTP388	Requirement ID from Type deleted
FRTP389	Requirement ID from Type deleted
FRTP390	Requirement is specified within ISO 10681
FRTP391	Requirement is specified within ISO 10681
FRTP392	Requirement is specified within ISO 10681
FRTP393	Requirement is specified within ISO 10681
FRTP394	Requirement is specified within ISO 10681
FRTP395	Requirement is specified within ISO 10681
FRTP396	Requirement is specified twice ([FRTP242)
FRTP397	Removed according to
FRTP401	Requirement is specified within ISO 10681
FRTP403	Requirement ID from Type deleted
FRTP404	Requirement ID from Type deleted
FRTP406	Requirement is specified within ISO 10681
FRTP407	Requirement is specified within ISO 10681
FRTP408	Requirement is specified within ISO 10681
FRTP409	Requirement is specified within ISO 10681
FRTP410	Requirement is specified within ISO 10681
FRTP413	Requirement ID from Type deleted
FRTP414	Requirement ID from Type deleted
FRTP417	Requirement ID deleted
FRTP419	Requirement ID deleted
FRTP425	Requirement ID from Type deleted
FRTP426	Requirement ID from Type deleted
FRTP427	Requirement ID from Type deleted
FRTP446	Requirement is specified within ISO 10681
FRTP447	Requirement is specified within ISO 10681
FRTP600	Requirement deleted
FRTP1016	Requirement deleted
FRTP1027	Requirement deleted after MS-2 review
FRTP1031	Requirement deleted
FRTP1082	Requirement deleted after MS-2 review
FRTP1085	Requirement deleted
FRTP1098	Requirement deleted after MS-2 review
FRTP1103	Requirement deleted
FRTP1119	Requirement deleted
FRTP1122	Requirement deleted after MS-2 review
[FRTP111525	Requirement deleted after MS-2 review
[FRTP111526	Requirement deleted after MS-2 review

FRTP216	Requirement deleted after MS-3 review
FRTP1127	Requirement deleted after MS-3 review
FRTP1174	Requirement deleted after MS-4 review

11.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
FRTP214	[FRTP1001, [FRTP1002	Requirement split up in two requirements and explanation.
[FRTP195	[FRTP195; [FRTP1003; [FRTP1004	Requirement split up in three requirements and explanation
FRTP240	[FRTP1113	Requirement is now part of Table 5
FRTP249	[FRTP1113	Requirement is now part of Table 5
FRTP251	[FRTP1113	Requirement is now part of Table 5
FRTP352	[FRTP1015	Requirement changed because of new channel / connection behaviour.
FRTP355	[FRTP1018	Requirement changed because of new channel / connection behaviour.
FRTP341	[FRTP228	[FRTP228 specifies now the same requirement as FRTP341
FRTP380	[FRTP1100	Old Requirement is part of the new requirement
FRTP381	[FRTP1100	Old Requirement is part of the new requirement
FRTP382	[FRTP1100	Old Requirement is part of the new requirement
FRTP383	[FRTP1097	Editorial change
FRTP431	[FRTP1112	Editorial change
FRTP434	[FRTP1110	Old Requirement is part of the new requirement

11.3 Changed SWS Items

SWS Item	*Rationale*
[FRTP088	Max number of concurrent channels skipped.
[FRTP136	simplified
[FRTP137	simplified
FRTP142	Notification Result Types changed
FRTP145	ChangeParameterType modified according to ISO 10681-2
[FRTP208	Reference to source document included.
[FRTP228	modified
FRTP399	modified
FRTP416	modified

11.4 Added SWS Items

SWS Item	*Rationale*
[FRTP1000	New requirement (in chapter 5)
[FRTP1005	New requirement (in chapter 7)

[FRTP1006	New requirement (in chapter 7)
[FRTP1007	New requirement (in chapter 7)
[FRTP1008	New requirement (in chapter 7)
[FRTP1009	New requirement (in chapter 7)
[FRTP1010	New requirement (in chapter 7)
[FRTP1011	New requirement (in chapter 7)
[FRTP1012	New requirement (in chapter 7)
[FRTP1013	New requirement (in chapter 7)
[FRTP1014	New requirement (in chapter 7)
[FRTP1017	New requirement (in chapter 7)
[FRTP1019	New requirement (in chapter 7)
[FRTP1020	New requirement (in chapter 7)
[FRTP1021	New requirement (in chapter 7)
FRTP1022	New requirement (in chapter 7)
FRTP1023	New requirement (in chapter 7)
FRTP1024	New requirement (in chapter 7)
[FRTP1025	New requirement (in chapter 7)
[FRTP1026	New requirement (in chapter 7)
[FRTP1028	New requirement (in chapter 7)
[FRTP1029	New requirement (in chapter 7)
[FRTP1030	New requirement (in chapter 7)
[FRTP1032	New requirement (in chapter 7)
[FRTP1033	New requirement (in chapter 7)
[FRTP1034	New requirement (in chapter 7)
[FRTP1035	New requirement (in chapter 7)
[FRTP1036	New requirement (in chapter 7)
[FRTP1037	New requirement (in chapter 7)
[FRTP1038	New requirement (in chapter 7)
[FRTP1039	New requirement (in chapter 7)
[FRTP1040	New requirement (in chapter 7)
[FRTP1041	New requirement (in chapter 7)
[FRTP1042	New requirement (in chapter 7)
[FRTP1043	New requirement (in chapter 7)
[FRTP1044	New requirement (in chapter 7)
[FRTP1045	New requirement (in chapter 7)
[FRTP1046	New requirement (in chapter 7)
[FRTP1047	New requirement (in chapter 7)
[FRTP1048	New requirement (in chapter 7)
[FRTP1049	New requirement (in chapter 7)
[FRTP1050	New requirement (in chapter 7)
[FRTP1051	New requirement (in chapter 7)
[FRTP1052	New requirement (in chapter 7)
[FRTP1053	New requirement (in chapter 7)
[FRTP1054	New requirement (in chapter 7)
[FRTP1055	New requirement (in chapter 7)
[FRTP1056	New requirement (in chapter 7)
[FRTP1057	New requirement (in chapter 7)
[FRTP1058	New requirement (in chapter 7)
[FRTP1059	New requirement (in chapter 7)
[FRTP1060	New requirement (in chapter 7)
[FRTP1061	New requirement (in chapter 7)
[FRTP1062	New requirement (in chapter 7)
[FRTP1063	New requirement (in chapter 7)
[FRTP1064	New requirement (in chapter 7)
[FRTP1065	New requirement (in chapter 7)
[FRTP1066	New requirement (in chapter 7)
[FRTP1067	New requirement (in chapter 7)
[FRTP1068	New requirement (in chapter 7)

[FRTP1069	New requirement (in chapter 7)
[FRTP1070	New requirement (in chapter 7)
[FRTP1071	New requirement (in chapter 7)
[FRTP1072	New requirement (in chapter 7)
	New requirement (in chapter 7)
[FRTP1074	New requirement (in chapter 7)
[FRTP1075	New requirement (in chapter 7)
[FRTP1076	New requirement (in chapter 7)
[FRTP1077	New requirement (in chapter 7)
[FRTP1078	New requirement (in chapter 7)
[FRTP1079	New requirement (in chapter 7)
[FRTP1080	New requirement (in chapter 7)
[FRTP1081	New requirement (in chapter 7)
[FRTP1083	New requirement (in chapter 7)
[FRTP1084	New requirement (in chapter 7)
[FRTP1088	New requirement (in chapter 7)
[FRTP1089	New requirement (in chapter 7)
[FRTP1090	New requirement (in chapter 7)
[FRTP1091	New requirement (in chapter 7)
[FRTP1092	New requirement (in chapter 7)
[FRTP1093	New requirement (in chapter 7)
[FRTP1094	New requirement (in chapter 7)
[FRTP1095	New requirement (in chapter 7)
[FRTP1096	New requirement (in chapter 7)
[FRTP1099	New requirement (in chapter 7)
[FRTP1100	New requirement (in chapter 7)
[FRTP1101	New requirement (in chapter 7)
[FRTP1102	New requirement (in chapter 7)
FRTP1103	New requirement (in chapter 7)
[FRTP1104	New requirement (in chapter 7)
[FRTP1105	New requirement (in chapter 7)
[FRTP1106	New requirement (in chapter 7)
[FRTP1107	New requirement (in chapter 7)
[FRTP1108	New requirement (in chapter 7)
[FRTP1109	New requirement (in chapter 7)
[FRTP1111	New requirement (in chapter 7)
[FRTP1113	New requirement (in chapter 7)
[FRTP1114	New requirement (in chapter 7)
[FRTP1115	New requirement (in chapter 7)
[FRTP1116	New requirement (in chapter 7)
[FRTP1117	New requirement (in chapter 7)
[FRTP1118	New requirement (in chapter 7)
FRTP1119	New requirement (in chapter 7)
[FRTP1120	New requirement (in chapter 7)
[FRTP1121	New requirement (in chapter 8)
[FRTP1123	New requirement (in chapter 7)
[FRTP1124	New requirement (in chapter 7)
[FRTP1129	New requirement (in chapter 5)
[FRTP1131	New requirement (in chapter 10.2)
[FRTP1131	New requirement (in chapter 8.2)
FRTP1134	New requirement (in chapter 7.5.1)
FRTP1135	New requirement (in chapter 5.6.2
FRTP1136	New requirement (in chapter 7.5.3.2.1)
FRTP1136	New requirement (in chapter 8.2)
FRTP1138	New requirement
FRTP1139 – FRTP1150	New requirement
FRTP1152	New requirement

FRTP1152	New requirement (in chapter 10)
FRTP1153	New requirement (in chapter 10)
FRTP1154	New requirement (in chapter 10)
FRTP1178	New requirement (in chapter 8)
FRTP1179	New requirement (in chapter 7)
FRTP1180	New requirement (in chapter 7)
FRTP1181	New requirement (in chapter 7)
FRTP1182	New requirement (in chapter 7)
FRTP1183	New requirement (in chapter 7)
FRTP1184	New requirement (in chapter 7)
FRTP1185	New requirement (in chapter 7)
FRTP1186	New requirement (in chapter 7)
FRTP1187	New requirement (in chapter 7)
FRTP001_PI	Rework of Published Information

12 Changes from Release 4.0 Rev 1

12.1 Deleted SWS Items

SWS Item	*Rationale*
FRTP1086	
FRTP1087	

12.2 Added SWS Items

SWS Item	*Rationale*
[FRTP1000051_Conf	New requirement (in chapter 10)

13 Changes from Release 4.0 Rev 3

13.1 Deleted SWS Items

SWS Item	*Rationale*
FRTP201	

13.2 Added SWS Items

SWS Item	*Rationale*
FRTP1189	
FRTP1190	
FRTP1191	

14 Not applicable requirements

[FRTP9999]「These requirements are not applicable to this specification.」
(BSW00306, BSW00312, BSW00314, BSW00323, BSW00325, BSW00326, BSW00328, BSW00329, BSW00330, BSW00331, BSW00333, BSW00335, BSW00341, BSW00343, BSW00345, BSW00347, BSW00350, BSW00358, BSW00370, BSW00371, BSW00373, BSW00375, BSW00376, BSW00377, BSW00386, BSW00387, BSW00401, BSW00405, BSW00409, BSW00410, BSW00413, BSW00414, BSW00415, BSW00417, BSW00423, BSW00424, BSW00425, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW005, BSW006, BSW009, BSW010, BSW159, BSW160, BSW161, BSW162, BSW164, BSW167, BSW168, BSW172, BSW05082)