Document Title	Specification of FlexRay AUTOSAR Transport Layer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	601
Document Classification	Standard

Document Version	2.6.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	2.6.0	AUTOSAR Release Management	• Avoided the retry mechanism to get a buffer from PduR • Removed BSW05129 and fixed the linking to BSW05129 • Removed FRARTP prefix for fields of FrTp frames and used camel case notation consistently for EcuC parameters • Updated the description of configuration parameters FrTpHaveTc/FrTpTc, FrTpPduId • Editorial changes • Removed chapter(s) on change documentation
17.05.2012	2.5.0	AUTOSAR Administration	• New Document ID and Title • Organization of PDUs in PDU pools • Dynamic assignment of Tx N-PDUs to connections at runtime • Reserved Tx N-PDUs for high priority connections • Cleanup of Document (incl. SWS IDs)
19.04.2011	2.4.0	AUTOSAR Administration	• Added new TP layer status to Table 3 • Corrected inconsistencies of the attributes Synchronicity and Reentrancy for FrTp_CancelTransmitRequest, FrTp_CancelReceiveRequest and FrTp_ChangeParameterRequest APIs • Added information about selection of FlexRay TP Protocol Engine • Added support for TP receive cancelation • Updated FrTp_ChangeParameter API syntax

15.09.2010	2.3.0	AUTOSAR Administration	• Added FRTP222, FRTP223 • Modified FRTP195 • Use parameter PduInfoType in callback RxIndication • Legal disclaimer revised
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
17.12.2007	2.2.0	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
24.01.2007	2.1.1	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added
05.12.2006	2.1.0	AUTOSAR Administration	• Correction in Interaction Diagram • Various descriptions adapted in Chapter 10 • Added BSW00435 due to WP112 decision • Changing API FrTp_Transmit • Several wording corrections • Adaptation of chapter 5.4.2 to new SRS Requirement • Legal disclaimer revised
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 configuration of the AUTOSAR basic software module FlexRay AUTOSAR Transport Layer (FrTp).

The FrTp Layer is between the PDU Router [6] and the FlexRay Interface module [5] (see Figure 1, according to [2]). This module's main purpose is segmentation and reassembly of messages that do not fit in a single FlexRay L-SDU.

The PDU Router deploys I-PDUs of AUTOSAR COM or DCM to different communication protocols. The routing through a network system type (e.g. CAN, LIN and FlexRay) depends on the I-PDU identifier. The PDU-Router is also in charge of determining whether a transport protocol has to be used or not.
The FlexRay Interface (FrIf) provides equal mechanisms to access a FlexRay bus channel regardless of its location (µC internal/external). It abstracts from the location of FlexRay controllers (on chip / onboard), the ECU hardware layout and the number of FlexRay drivers. The FrIf is in charge to route received PDUs to the FrTp, the PDU Router, the FlexRay NM and the XCP (only defined in AUTOSAR release 4 and beyond).

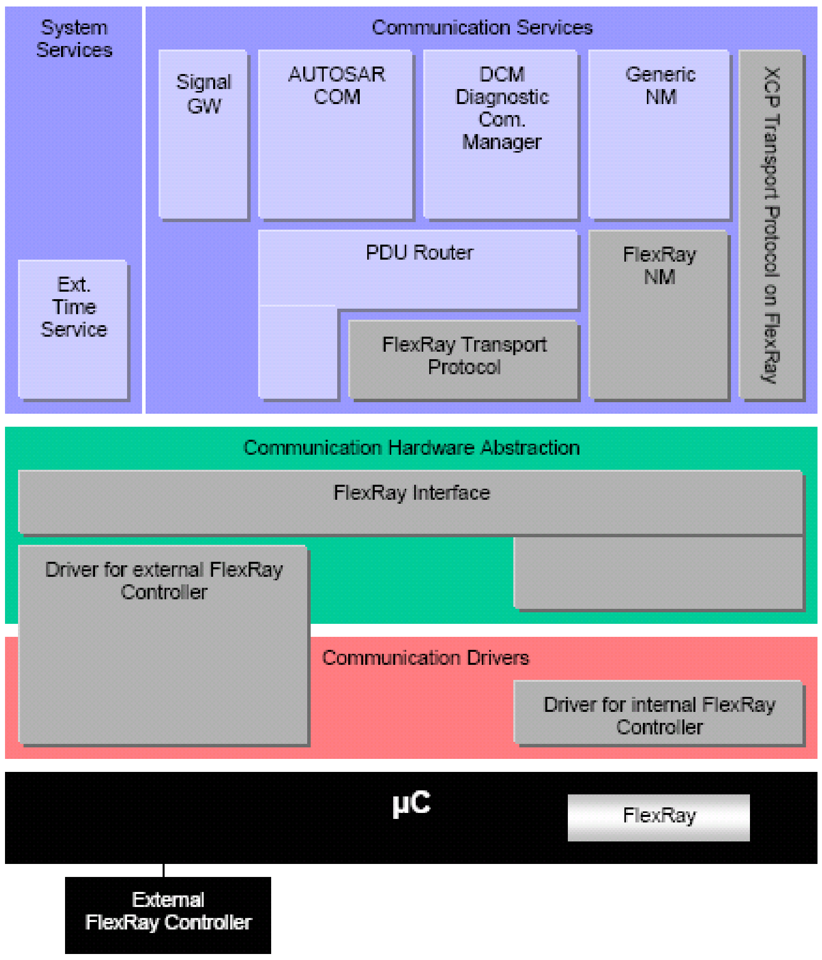

System Services
Communication Services
Signal GW
AUTOSAR COM
DCM Diagnostic Com. Manager
Generic NM
XCP Transport Protocol on FlexRay
PDU Router
FlexRay NM
Ext. Time Service
FlexRay Transport Protocol
Communication Hardware Abstraction
FlexRay Interface
Driver for external FlexRay Controller
Communication Drivers
Driver for internal FlexRay Controller
µC
FlexRay
External FlexRay Controller

Figure 1: AUTOSAR FlexRay Layered Architecture

Among others, the FlexRay AUTOSAR Transport Layer includes the following features:

- Segmentation of data in send direction
- Collection of data in receive direction
- Control of data flow
- Detection of errors
- Acknowledgement (and Retry)
- 1:1 and 1:n connections
- 2 or 4 Bytes address information
- Transfer of up to 2^{32}-1 Bytes payload
- Configurable to be compliant to ISO 15765-2 regarding frame layout and sequences

This specification supports only the AUTOSAR FlexRay Transport Protocol derived from ISO 15765-2, which is used as standard in AUTOAR release 3.x and below. Starting with AUTOSAR release 4.0, the standard FlexRay transport layer is compatible to ISO 10681-2.For AUTOSAR release 3.2, a back port of the ISO 10681-2 compliant FlexRay transport layer has been created as a separate document named FlexRay ISO Transport Layer [11]. Thus, both in AUTOSAR release 3.2 and 4.0, users must be cautious in choosing which specification to use for FlexRay Transport Layer.

It is an AUTOSAR decision to base on existing standards the specification of basic software module. So the FlexRay AUTOSAR Transport Layer specification is based on the international standard ISO 15765 (Diagnostics on CAN), which is the most used in automotive area.

The basic idea is to have an ISO 15765-2 compliant Transport Layer, which allows by the means of static configuration to add one or more optional features (like acknowledgement) per channel independently of each other. Of course, by adding such a feature the compliance to ISO 15765-2 gets lost for this particular channel.
Additionally, the features are deactivateable at compile time. Even if they are compiled in, they are still deactivateable by static configuration.
The rationale behind some of the provided features is the usage of this transport layer not only for diagnostic purposes but also for Inter-ECU communication.

Since addressing within ISO 15765-2 is specific for the CAN bus system (CAN identifier), it is obvious that another approach is taken within FlexRay AUTOSAR Transport Layer.

Although FlexRay transport protocol is at first set to vehicle diagnostic systems, it has been developed to also deal with requirements from other FlexRay based systems needing a transport layer protocol.

2 Acronyms and abbreviations

Following acronyms and abbreviations have a local scope only and therefore are not contained in the AUTOSAR glossary.

Acronym:	***Description:***
Channel	A channel hosts a group of connections sharing the properties configurable by the parameters in section 10.2.
Connection	Communication path between two nodes (1:1) or one node and the network (1:n), characterized by the parameters in section 10.2.
Frame	Synonymous for N-PDU or L-SDU.
I-PDU	PDU of the AUTOSAR COM module; corresponds to an N-SDU of the FlexRay Transport Layer.
L-SDU	This is the SDU of the FlexRay Interface module. It represents the same entity as the N-PDU, but from the FlexRay Interface’s point of view.
L-SDU ID	Unique identifier of an L-SDU; used by upper layers such as the FrTp to interact with the FlexRay Interface.
Message	Synonymous for N-SDU or I-PDU.
N-PDU	This is a PDU of the FlexRay Transport Layer, which is given to the FlexRay Interface for sending. It consists of address information, protocol control information and the payload (N-SDU).
N-SDU	This is the SDU of the FlexRay Transport Layer. In the AUTOSAR architecture, it is a set of data exchanged with the PDU Router.
N-SDU ID	Unique identifier of an SDU; used by upper layers such as the PduR to interact with the FlexRay Transport Layer.
PDU pool	Set of FlexRay N-PDUs that share the same size and same addressing type.

Abbreviation:	***Description:***
AF	Acknowledgement Frame
CF	Consecutive Frame
COM	AUTOSAR COM module
FC	Flow Control
FF	First Frame
Fr	FlexRay
PCI	Protocol Control Information
FrIf	FlexRay Interface
FrTp	FlexRay Transport Layer
NM	Network Management
PDU	Protocol Data Unit
PduR	PDU Router
SDU	Service Data Unit
SF	Single Frame
XCP	Universal Calibration Protocol

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements of Basic Software Modules
AUTOSAR_SRS_General.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_PDU_Router.pdf

[7] Specification of Communication Stack Types
AUTOSAR_SWS_ComStackTypes.pdf

[8] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[9] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

[10] AUTOSAR Basic Software Module Description Template
AUTOSAR_BSWMDTemplate.pdf

[11] Specification of FlexRay ISO Transport Layer
AUTOSAR_SWS_FlexRay_ISO_TP.pdf

3.2 Related standards and norms

[12] ISO 15765-2 (2003-11-11), Road vehicles — Diagnostics on Controller Area Networks (CAN) — Part2: Network layer services

[13] ISO 10681-2, Road vehicles — Communication on FlexRay — Part2: Communication Layer Services

[14] FlexRay Communications System Protocol Specification Version 2.1

4 Constraints and assumptions

4.1 Limitations

AUTOSAR architecture defines protocol specific transport layer (CanTp, LinTp, FrTp, etc.). Thus, the FlexRay AUTOSAR Transport Layer covers only FlexRay transport protocol specifics.

The FlexRay AUTOSAR Transport Layer has an interface to a single underlying FlexRay Interface Layer and a single upper PDU Router module.

AUTOSAR release 3 does not support transport protocol facilities for AUTOSAR COM. Therefore, I-PDUs of COM are limited to the configured payload size of the FlexRay bus.

4.2 Applicability to car domains

The FlexRay AUTOSAR Transport Layer can always be used for applications if the FlexRay protocol was used.

5 Dependencies to other modules

This section sets out relations between the FrTp and other AUTOSAR basic software modules. It contains brief descriptions of the services, which are required by the FrTp from other modules or other modules can call at the FrTp. The following picture gives a brief overview of the interactions.

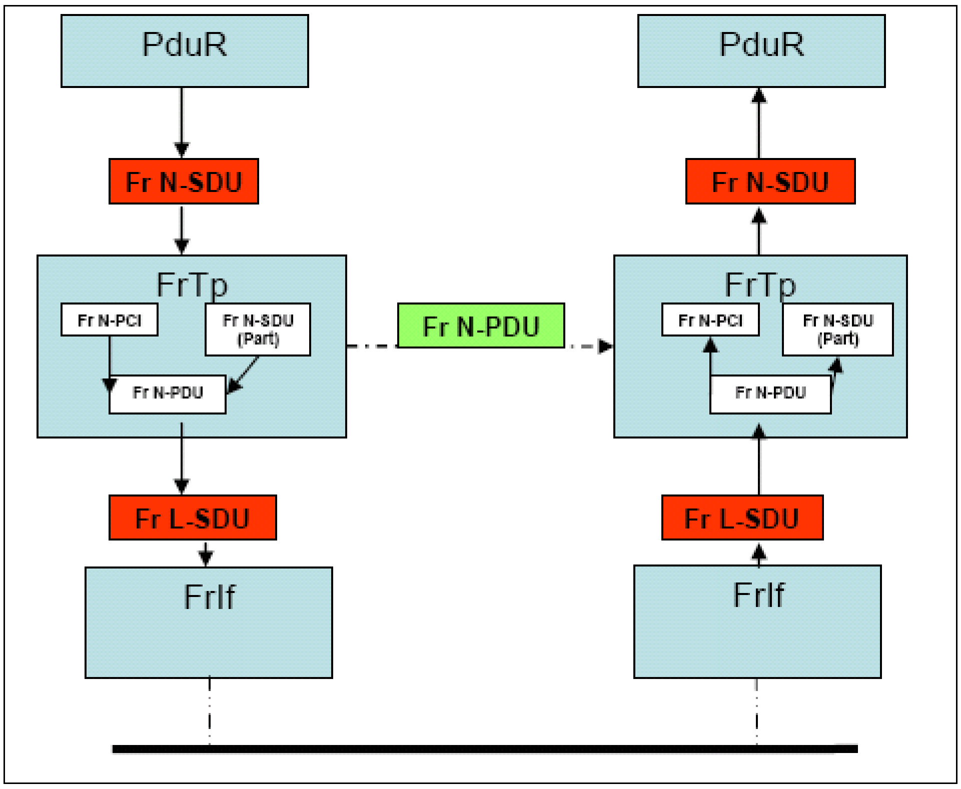


Figure 2: FrTp interactions

5.1 PduRouter

The following services of the PduRouter are called by the FrTp:

- *PduR_FrTpRxIndication*
  By this API service, the FrTp indicates the completed (un)successful reception of a message
- *PduR_FrTpProvideRxBuffer*
  By this API service, the FrTp asks the actual receiver (e.g. DCM) of the message to provide a receive buffer. It is not necessary for the buffer to have at least the same size as the whole N-SDU length (there will be another call in this case).

- *PduR_FrTpProvideTxBuffer*
  By this API service, the FrTp asks the actual sender (e.g. DCM) of the message to provide a transmit data. It is not necessary for the transmit data to have at least the same size as the whole N-SDU length (there will be another call in this case).
- *PduR_FrTpTxConfirmation*
  By this API service, the FrTp confirms the (un)successful sending of the complete message (N-SDU) to the actual sender (e.g. DCM).

The following services of the FrTp are called by the PduRouter:

- *FrTp_Transmit*
  By this API service, the sending of a message (N-SDU) is triggered. The FrTp will then ask for a transmit buffer and start sending.
- *FrTp_CancelTransmit*
  By this API service, the sending of a message (N-SDU) is cancelled. This service is optional (per channel).
- *FrTp_CancelReceive*
  By this API service, the receiving of a message (N-SDU) is cancelled. This service is optional (per channel).
- *FrTp_ChangeParameter*
  By this API service, the STmin value of a connection can be changed.

5.2 FlexRay Interface

The following services of the FlexRay Interface are called by the FrTp:

- *FrIf_Transmit*
  By this API service, the sending of a frame (N-PDU) is triggered. Depending on configuration on the FlexRay Interface, the N-PDU is sent immediately or after the call of FrTp_TriggerTransmit*.*

The following services of the FrTp are called by the FlexRay Interface:

- *FrTp_RxIndication*
  By this API service, the FlexRay Interface indicates the reception of an FrTp frame (N-PDU, please do not mistake this with a FlexRay frame) to the FrTp. The FrTp then processes this frame.
- *FrTp_TxConfirmation*
  By this API service, the FlexRay Interface confirms the sending of the frame containing the N-PDU over the FlexRay network.
- *FrTp_TriggerTransmit*
  By this API service, the FlexRay Interface makes the FrTp to copy the N-PDU into the buffer provided by the FlexRay Interface. The FlexRay interface then can start sending the FlexRay frame containing the N-PDU.

5.3 ECU State Manager

The following services of the FrTp are called by the ECU State Manager:

- *FrTp_Init*
  By this API service, all global variables are initialized and each connection is set into the Idle state.
- *FrTp_Shutdown*
  By this API service, all pending transport connections are closed, resources are freed and the module is stopped.

5.4 File structure

5.4.1 Code file structure

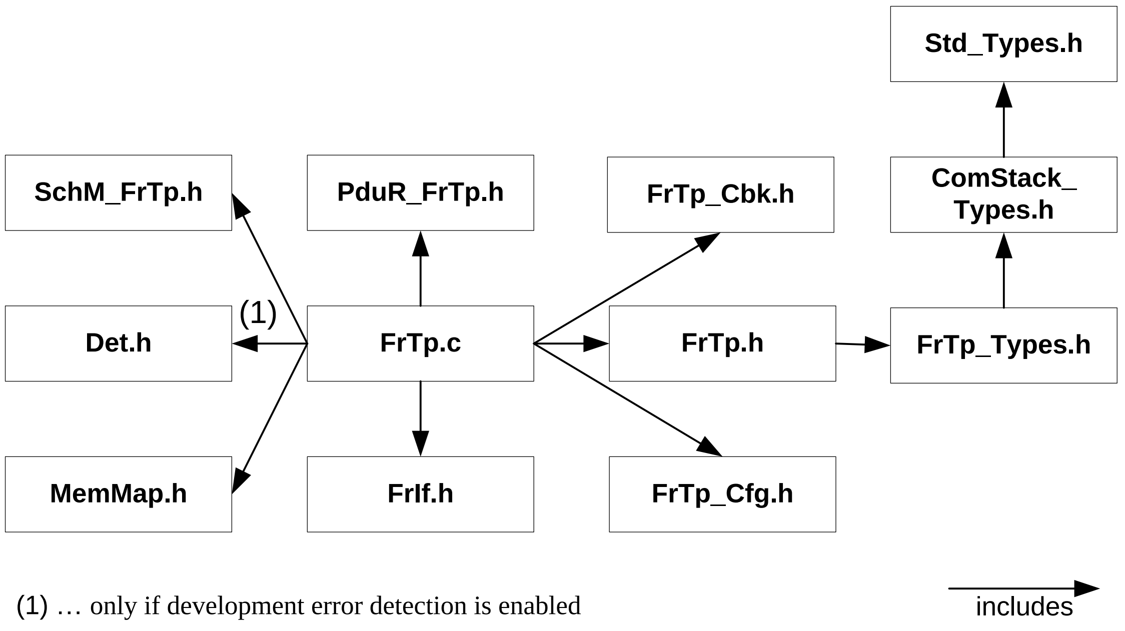
FRTP214: The Code file structure shall include the following files named:
- FrTp.c – the source code
- FrTp_Lcfg.c – for link time configurable parameters
- FrTp_PBcfg.c – for post build time configurable parameters

The latter files shall contain all link time and post-build time configurable parameters.

5.4.2 Header file structure

FRTP195: The Header file structure shall include the following files named:
- FrTp.h - general header file
- FrTp_Cfg.h - pre-compile time configuration parameters
- Det.h – header file of DET
- PduR_FrTp.h – header file of PduR
- FrIf.h – header file of FrIf
- SchM_FrTp.h – header file of TP related SchM declarations
- MemMap.h – header file for Memory Mapping
- 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

FRTP222: The FrTp.h file shall include FrTp_Types.h.



5.4.3 Design Rules

FRTP208: The code of the FrTp shall conform to the HIS subset of the MISRA C Standard.

FRTP209: Direct use of compiler and platform specific keywords shall be avoided.

FRTP210: Indicate all global data with read-only purposes by explicitly assigning the `const` keyword.

FRTP211: It is allowed to use macros instead of functions where source code is used and runtime is critical.

FRTP212: No global data shall be defined in the header files. If global variables have to be used, the definition shall take place in the C file.

FRTP213: The source code of the FrTp module shall not be processor and compiler dependent.

6 Requirements traceability

Document: General Requirements of Basic Software Modules

Requirement	***Satisfied by***
[BSW00160] Human-readable configuration data	Fulfilled by chapter 10
[BSW00161] Microcontroller abstraction	not applicable
[BSW00162] ECU layout abstraction	not applicable
[BSW00172] Compatibility and documentation of scheduling strategy	not applicable
[BSW003] Version identification	**FRTP200:**
[BSW003] Version identification	**FRTP178:**
[BSW00300] Module naming convention	Fulfilled by API definitions in chapter 8
[BSW00301] Limit imported information	not applicable
[BSW00302] Limit exported information	not applicable
[BSW00304] AUTOSAR integer data types	Fulfilled by API definitions in chapter 8
[BSW00305] Self-defined data types naming convention	Fulfilled by type definitions in chapter 8
[BSW00306] Avoid direct use of compiler and platform specific keywords	**FRTP209:**
[BSW00307] Global variables naming convention	not applicable
[BSW00308] Definition of global data	**FRTP212:**
[BSW00309] Global data with read-only constraint	**FRTP210:**
[BSW00310] API naming convention	chapter 8
[BSW00312] Shared code shall be reentrant	Here the means are described so this is a requirement to implementation
[BSW00314] Separation of interrupt frames and service routines	not applicable
[BSW00318] Format of module version numbers	**FRTP178:**
[BSW00321] Enumeration of module version numbers	not applicable
[BSW00323] API parameter checking	**FRTP205:**
[BSW00325] Runtime of interrupt service routines	not applicable
[BSW00326] Transition from ISRs to OS tasks	not applicable
[BSW00327] Error values naming convention	Fulfilled by section 7.6
[BSW00328] Avoid duplication of code	This is a requirement to implementation
[BSW00329] Avoidance of generic interfaces	not applicable
[BSW00330] Usage of macros / inline functions instead of functions	**FRTP211:**
[BSW00331] Separation of error and status values	Fulfilled by the different types
[BSW00333] Documentation of callback function context	Fulfilled by API definitions in chapter 8
[BSW00334] Provision of XML file	not applicable
[BSW00335] Status values naming convention	not applicable
[BSW00336] Shutdown interface	**FRTP148:**
[BSW00337] Classification of errors	**FRTP179:**
[BSW00338] Detection and Reporting of development errors	**FRTP217:**
[BSW00339] Reporting of production relevant errors and exceptions	not applicable (No productions are available)
[BSW00341] Microcontroller compatibility documentation	not applicable
[BSW00342] Usage of source code and object code	not applicable

[BSW00343] Specification and configuration of time	Fulfilled by chapter 10
[BSW00344] Reference to link-time configuration	not applicable (no link-time only parameters)
[BSW00345] Configuration at Compile time	**FRTP178:**
[BSW00346] Basic set of module files	**FRTP195:**
[BSW00347] Naming separation of different instances of BSW drivers	not applicable For driver only.
[BSW00348] Standard type header	not applicable
[BSW00350] Development error detection keyword	**FRTP217:**
[BSW00353] Platform specific type header	not applicable
[BSW00355] Do not redefine AUTOSAR integer data types	Fulfilled by API definitions in chapter 8
[BSW00357] Standard API return type	Fulfilled by API definitions in chapter 8
[BSW00358] Return type of `init()` functions	Fulfilled by API definitions in chapter 8
[BSW00359] Return type of callback functions	Fulfilled by API definitions in chapter 8
[BSW00360] Parameters of callback functions	Fulfilled by API definitions in chapter 8
[BSW00361] Compiler specific language extension header	not applicable
[BSW00369] Do not return development error codes via API	**6.1.1** FRTP149: **-** **FRTP154:**
[BSW00370] Separation of callback interface from API	Fulfilled by chapter 8
[BSW00371] Do not pass function pointers via API	Fulfilled by API definitions in chapter 8
[BSW00373] Main processing function naming convention	Fulfilled by API definitions in chapter 8
[BSW00374] Module vendor identification	**FRTP178:**
[BSW00375] Notification of wake-up reason	not applicable
[BSW00376] Return type and parameters of main processing functions	Fulfilled by API definitions in chapter 8
[BSW00377] Module specific API return types	Fulfilled by API definitions in chapter 8
[BSW00378] AUTOSAR boolean type	Fulfilled by API definitions in chapter 8
[BSW00379] Module identification	**FRTP178:**
[BSW00380] Separate C-Files for configuration parameters	**FRTP214:**
[BSW00381] Separate configuration header file for pre-compile time parameters	**FRTP195: FRTP214:**
[BSW00383] List dependencies of configuration files	**FRTP195: FRTP214:**
[BSW00384] List dependencies to other modules	Fulfilled by chapter 5
[BSW00385] List possible error notifications	Fulfilled by section 7.6
[BSW00386] Configuration for detecting an error	Fulfilled by chapter 10
[BSW00387] Specify the configuration class of callback function	Fulfilled by chapter 8
[BSW00388] Introduce containers	Fulfilled by section 10.2
[BSW00389] Containers shall have names	Template requests names, so the requirement is fulfilled
[BSW00390] Parameter content shall be unique within the module	Parameters are unique
[BSW00391] Parameter shall have unique names	Parameters have unique names
[BSW00392] Parameters shall have a type	Template requests type, so the requirement is fulfilled
[BSW00393] Parameters shall have a range	Template requests range, so the requirement is fulfilled
[BSW00394] Specify the scope of the parameters	Template requests scope, so the requirement is fulfilled

[BSW00395] List the required parameters (per parameter)	not applicable
[BSW00396] Configuration classes	Parameter-template requests configuration classes
[BSW00397] Pre-compile-time parameters	This is not a requirement, it is a description
[BSW00398] Link-time parameters	This is not a requirement, it is a description
[BSW00399] Loadable Post-build time parameters	Done by configuration description
[BSW004] Version check	**FRTP201:**
[BSW00400] Selectable Post-build time parameters	not applicable
[BSW00401] Documentation of multiple instances of configuration parameters	Fulfilled by chapter 10
[BSW00402] Published information	**FRTP178:**
[BSW00404] Reference to post build time configuration	**FRTP195:**
[BSW00405] Reference to multiple configuration sets	not applicable
[BSW00406] Check module initialization	To perform this check the start up code of the microcontroller has to initialize the status variables. Furthermore E_UNINIT is not defined in the Std_ReturnType
[BSW00407] Function to read out published parameters	**FRTP202:**
[BSW00408] Configuration parameter naming convention	Fulfilled by chapter 10
[BSW00409] Header file for production error code IDs	not applicable
[BSW00410] Compiler switch shall have defined values	Template requests compiler switches with defined values, so the requirement is fulfilled
[BSW00411] Get version info keyword	**FRTP215:**
[BSW00412] Separate H-File for configuration parameters	not applicable, post build time configuration is referenced in the init-function
[BSW00413] Accessing instances of BSW modules	not applicable Only 1 instance of FrTp allowed.
[BSW00414] Parameter of init function	Fulfilled by **FRTP147:**
[BSW00415] User dependent include files	not applicable
[BSW00416] Sequence of initialization	not applicable
[BSW00417] Reporting of Error Events by Non-Basic Software	not applicable
[BSW00419] Separate C-Files for pre-compile time configuration parameters	not applicable
[BSW00422] Pre-Debouncing of production relevant error status	not applicable DEM requirement
[BSW00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Template used
[BSW00424] BSW main processing function task allocation	**FRTP162:**
[BSW00425] Trigger conditions for schedulable objects	not applicable
[BSW00426] Exclusive areas in BSW modules	not applicable
[BSW00427] ISR description for BSW modules	not applicable No ISR function
[BSW00428] Execution order dependencies of main processing functions	not applicable FlexRay TP has only one MainFunction

[BSW00429] Restricted BSW OS functionality access	not applicable
[BSW00431] The BSW Scheduler module implements task bodies	not applicable
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Fulfilled by chapter 8
[BSW00433] Calling of main processing functions	not applicable
[BSW00434] The Schedule module shall provide an API for exclusive areas	not applicable
[BSW00435] Module Header File Structure for the Basic Software Scheduler	**FRTP195:**
[BSW00436] Module Header File Structure for Memory Mapping	**FRTP195:**
[BSW005] No hard coded horizontal interfaces within MCAL	not applicable
[BSW006] Platform independency	**FRTP213:**
[BSW007] HIS MISRA C	**FRTP208:**
[BSW009] Module User Documentation	Fulfilled by the whole document
[BSW010] Memory resource documentation	not applicable
[BSW101] Initialization interface	**FRTP147:**
[BSW158] Separation of configuration from implementation	Redundant to BSW00346
[BSW159] Automatic configuration	**FRTP178:** , **FRTP180:** , **FRTP181:**
[BSW164] Implementation of interrupt service routines	not applicable
[BSW167] Static configuration checking	**FRTP171: FRTP174:** , **FRTP180:** , **FRTP181:**
[BSW168] Diagnostic interface	not applicable
[BSW170] Data for reconfiguration of AUTOSAR SW-components	not applicable
[BSW171] Configurability of optional functionality	section 10.2

Document: AUTOSAR requirements on Basic Software, cluster FlexRay

Requirement	***Satisfied by***
BSW05073 (Usage of ISO 15765-2 and ISO 15765-4 specifications)	**FRTP009:**
BSW05074 (FlexRay Transport Interfaces)	Fulfilled by chapter 8
BSW05075 (Configuration Independence)	**FRTP149:**
BSW05123 (Configuration Modifiable by a Flashing Process)	not supported
BSW05076 (Multiple Logical FlexRay Transport Layer Channels)	Section 7.1.3
BSW05077 (Unique Identifier of N-SDU)	section 10.2
BSW05079 (Transport Connection Properties)	section 10.2
BSW05082 (Acknowledgement without Retry)	section 7.2.1.2
BSW05083 (Acknowledgement with Retry)	section 7.2.1.3
BSW05085 (Segmented 1:n Connections without Flow Control)	section 7.2.2
BSW05104 (Default Separation Time)	section 10.2.4
BSW05088 (FlexRay Transport Layer Initialization)	**FRTP147:**
BSW05089 (FlexRay Transport Layer Availability)	**FRTP179:**
BSW05090 (Support of Optional ISO 15765-2	**FRTP104:**

Service)	
BSW05093 (Transmit Cancellation)	**FRTP099:- FRTP103:**
BSW05095 (Bandwidth Control)	not applicable

7 Functional specification

The FrTp offers services for segmentation, transmission with flow control, and reassembly of messages (N-SDUs). Its main purpose is to transfer messages that may or may not fit in a single FlexRay frame.

FRTP192: The FlexRay AUTOSAR Transport Layer provides full duplex capabilities for PDU pools and connections.

FRTP201: The FlexRay TP shall perform a preprocessor-check if its source and header files belong to the same version.

FRTP200: A readable software version number shall be included in the header file.

7.1 Overview

7.1.1 Extensions of ISO 15765-2

FRTP009: Beside the features according to ISO 15765-2 (7 byte data per frame, 4 kByte message length, unsegmented 1:n connections, multiple logical channels concurrently, flow control, service request confirmation) it allows to configure independently of each other the following features for a specific channel at both pre- and post-compile time:

- Acknowledgement (with or without Retry) for 1:1 connections
- Segmented 1:n connections (without flow control)
- Transmission cancellation
- Up to 2^{32}-1 Byte message length

For the rest of this document, sections or features that are not compliant to ISO 15765-2 will be marked as “**Not compliant to ISO 15765-2**”.

7.1.2 Connections

Connections are used to transfer data from one sender to one (1:1) or more (1:n) receivers. Connections with one sender and one receiver are bi-directional; data can be transferred in both directions. To transport parts of a possibly much larger message, connections use FlexRay PDUs that are grouped into PDU pools. Connections may specify a number of prioritized PDUs that must be reserved for exclusive use as long as the connection is active.

7.1.3 Channels

A Channel is used to group several connections with similar properties and to manage access to the transport PDUs. Consequently, the channel itself carries all relevant properties, like addressing type, timing and handshake parameters, and acknowledgement and retry capability, and the transport PDUs of at most one received and one transmitted PDU pool.

7.1.4 PDU Pools

A PDU pool is a conceptional element, which groups the transport PDUs of several channels. The PDUs in a PDU pool need to have identical PDU sizes and addressing type. For a specific ECU, a PDU pool is either received or transmitted. The PDUs in a PDU pool may be used by one or more channels. To ensure the pool semantics in a configuration, channels must reference either all PDUs of a Pool, or none. It must also be ensured that no two connections assigned to the same PDU pool have identical addresses. The PDUs of a PDU pool are evenly distributed to all open connections at runtime, but only after all PDU required by open connections with prioritized PDUs have been assigned.

7.1.5 Active Connections

The maximum number of concurrently active connections is configurable separately for each channel via FrTpConcurrentConnections. This number can range from one connection at a time to all connections of the channel. For each active connection, two separate state machines are required, because connections can be bi-directional. These state machines belong to the channel, and are therefore associated with the PDU pool used by this channel.
State machines are assigned to connections at runtime when a new data-transmission is requested, or when frames of an incoming connection are received. When the maximum number of state machines is reached, further transmission requests or new incoming connections must be rejected or ignored, respectively.
For each state machine, a RAM buffer will be needed to store the content of the last received frame(s) until enough buffer is available during reception, or to store the content of the next to-be-transmitted frame(s) until enough data is available.

7.2 Protocol Processes

There are, as will be shown later on, different types of First Frames and Single Frames. So in the sequence diagrams, always FF or SF will be used, regardless of the concrete subtype.

7.2.1 1:1 Connections

This type of connections exists between two nodes and is bidirectional. Within the FlexRay AUTOSAR Transport Layer, the following subtypes are possible.

7.2.1.1 1:1 Connection in a channel without Acknowledgement

Unsegmented Transfer
When a message does not exceed the possible amount of payload for a SF (which can be derived from the N-PDU length, FrTpAdrType and FrTpLm）, there is no need to segment this message.

The transfer takes place as illustrated in Figure 3:

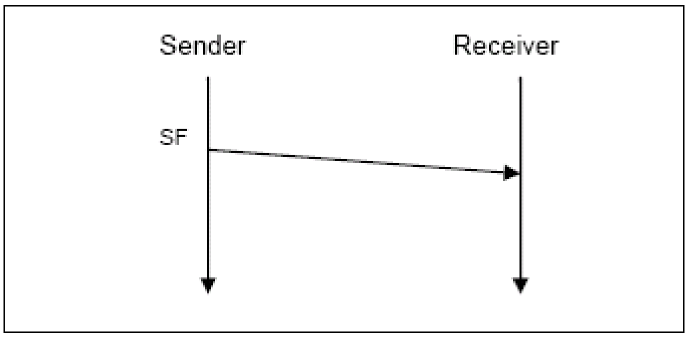


Figure 3: Unsegmented 1:1 transfer without acknowledgement

The sending transport layer packs the payload (N-SDU) into an N-PDU and sends it to the receiving transport layer. This is done via a Single Frame (SF).

Segmented Transfer
In case a message does not fit into an SF, it needs to be split up into several parts and flow control is applied to control the data flow taking into account the needs of the receiver.

In this case, the transfer takes place as shown in Figure 4:

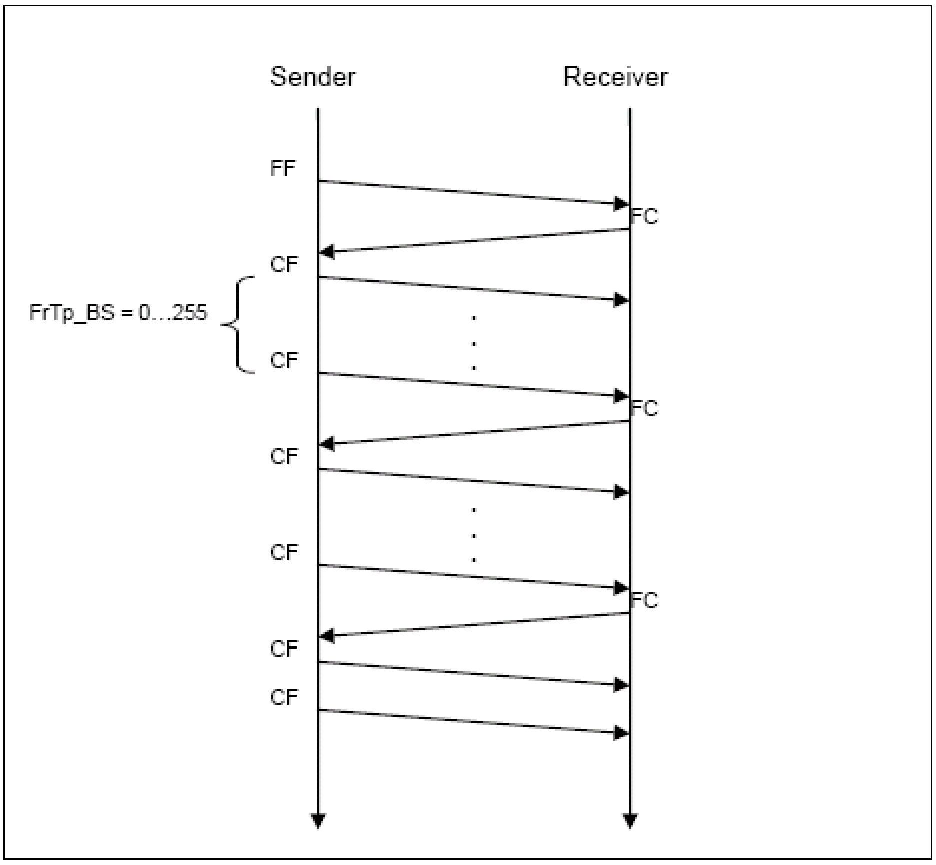


Figure 4: Segmented 1:1 transfer without acknowledgement

The transfer starts with sending a First Frame (FF) from the sender to the receiver. This frame contains the length of the whole message (e.g. 1000 Byte) and even the first data bytes.
The receiving peer reacts to the reception of a FF with sending of a Flow Control frame (FC) back to the sender. This FC frame contains the value of three parameters: FS, BS and STmin.

FS states the flow status. The possible values are:

• CTS:	Clear To Send The sender can continue transmitting the message
• WT:	Wait The sender shall wait for another FC frame.
• OVFLW:	Overflow The sender shall abort the transfer, because the receiver has not enough buffer for the whole message available.

There shall be a statically defined upper limit (FrTpMaxWft) for the number of allowed WT's. If this number has been reached, the transmission shall be aborted and within *PduR_FrTpTxConfirmation,* the result NTFRSLT_E_WFT_OVRN shall be returned.

BS specifies the block size. This is the number of Consecutive Frames (CF) the sender is allowed to send between two FC Frames. The possible range is from 0x00 to 0xFF, whereas 0x00 states that no more FC Frames will be transmitted by the receiver, i.e. the whole message shall be sent in one big block.

STmin defines the minimum gap between two CFs in milliseconds or microseconds. The valid values range from 0x00 to 0x7F and from 0xF1 to 0xF9. The range from 0x00 to 0x7F specifies the minimum gap in milliseconds (0ms .. 127ms), the one from 0xF1 to 0xF9 defines the gap in microseconds (100 µs, 200 µs, .. 900µs). The supported values of STmin are restricted by the placement of N-PDUs in FlexRay cycles, and are subject to the jitter created by the placement of N-PDUs in the slots of a cycle.

The alternating transmission of CF blocks and a FC frame lasts, until the whole message is sent.

The STMin parameter can be changed during runtime by using the respective API call.

7.2.1.2 1:1 Connection in a channel with Acknowledgement without Retry

This section is **Not compliant to ISO 15765-2** and describes how a simple acknowledgement mechanism looks like.

Unsegmented Transfer
This is mostly done like in section Unsegmented Transfer of section 7.2.1.1, except that there is an additional Acknowledge Frame (AF) which is sent from the receiver to the sender. This is illustrated in Figure 5:

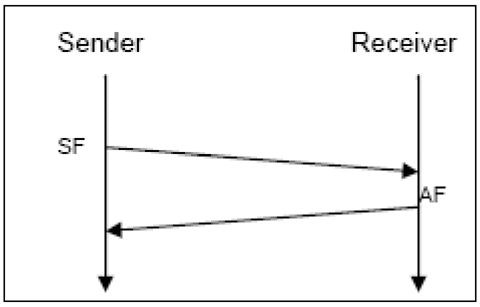


Figure 5: Unsegmented 1:1 transfer with Acknowledgement without Retry

The AF contains among others the field ACK, which has two possible values, Positive Acknowledgement (POS_ACK) or Negative Acknowledgement (NEG_ACK). Thus, the sender is informed about the (un)successful reception of a message by the receiving peer. If the FS field of an AF frame (see section 7.3.6) contains the value WT, another AF, up to FrTpMaxRn, will arrive.

Segmented Transfer
This is done very similar to section Segmented Transfer of section 7.2.1.1. There are only three differences:
The first difference is the transmission of an AF after the last block, because this one has to be acknowledged as well. This frame is similar to an ordinary Flow Control frame but contains additionally the ACK parameter (for positive or negative acknowledgement) and the sequence number of the first faulty frame of the transmitted block.
The second difference is the transmission of an AF with a negative acknowledgement after a block in which an error occurred. This AF also contains the sequence number of the first faulty or missing frame.
The third difference is, that the block size shall be in the range from 1 to 16 (due to the 4 bit sequence number, see section 7.3.4)

The procedure can be seen in Figure 6:

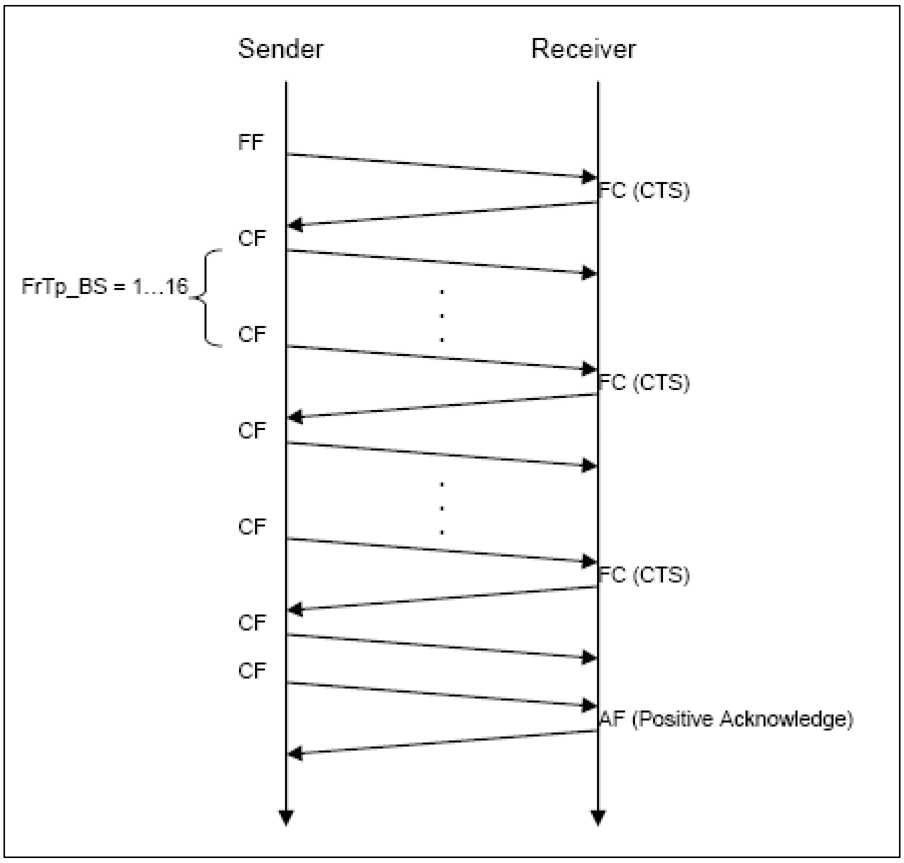


Figure 6: Segmented 1:1 transfer with Acknowledgement without Retry

Obviously, the acknowledgement is done on a “per block” basis, depending on the current block size.
In case of a negative acknowledgement after a block (in that case instead of an FC frame an AF with a negative acknowledgement is sent to the sender and the receiver aborts the reception and indicates an appropriate result to its upper layer

(*PduR_FrTpRxIndication*) the sender aborts the transmission and informs its upper layer (*PduR_FrTpTxConfirmation*).

7.2.1.3 1:1 Connection in a channel with Acknowledgement with Retry

This section is **Not compliant to ISO 15765-2**

Unsegmented Transfer
This section is quite similar to the corresponding one in section 7.2.1.2. The only difference is that in case of a negative acknowledgement the frame is retransmitted.

This behavior is depicted in Figure 7 and Figure 8:

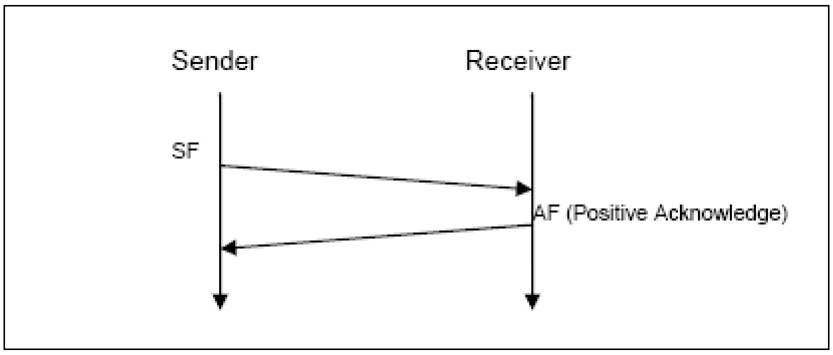


Figure 7: Unsegmented 1:1 transfer, Acknowledgement with Retry configured, Positive Acknowledgement

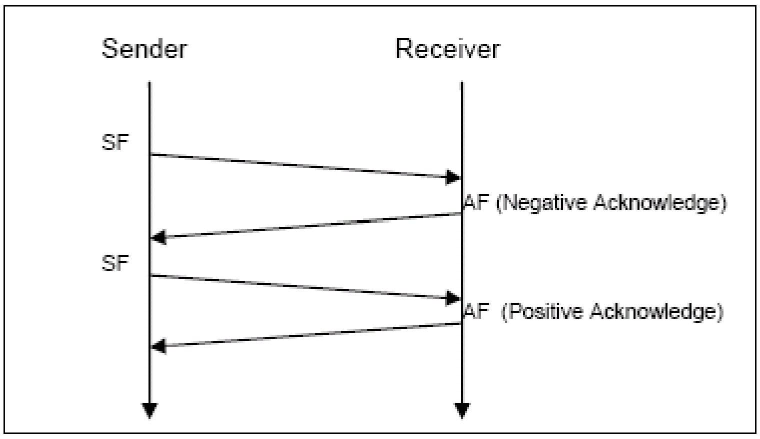


Figure 8: Unsegmented 1:1 transfer, Acknowledgement with Retry configured, Negative Acknowledgement

If in Figure 8 the second try of sending the message also failed, there would be a third one and so on.

In order to prevent infinite retransmissions in the case of a permanent failure, an upper limit (FrTpMaxRn) has to be defined. If the number of retries has reached this value, the transmission of the corresponding message shall be stopped and within *PduR_FrTpTxConfirmation* and *PduR_FrTpRxIndication,* an adequate result (see section 8.2.1) shall be returned.

Segmented Transfer

Compared to the segmented transfer in section 7.2.1.2, the difference is the Retry mechanism and, coming with it, the alternating block mechanism.

The Retry mechanism works as follows:

In the case a negative acknowledge arrives at the sender, this also contains the sequence number of the first faulty frame in the currently transmitted block. Now the sender transmits, starting with the stated sequence number, all remaining frames of the just transmitted block again.
In order to prevent infinite retransmissions in case of a permanent failure, the parameter FrTpMaxRn limits the retry attempts.
The Retry mechanism is shown in Figure 9 for the case of a block size of 4:

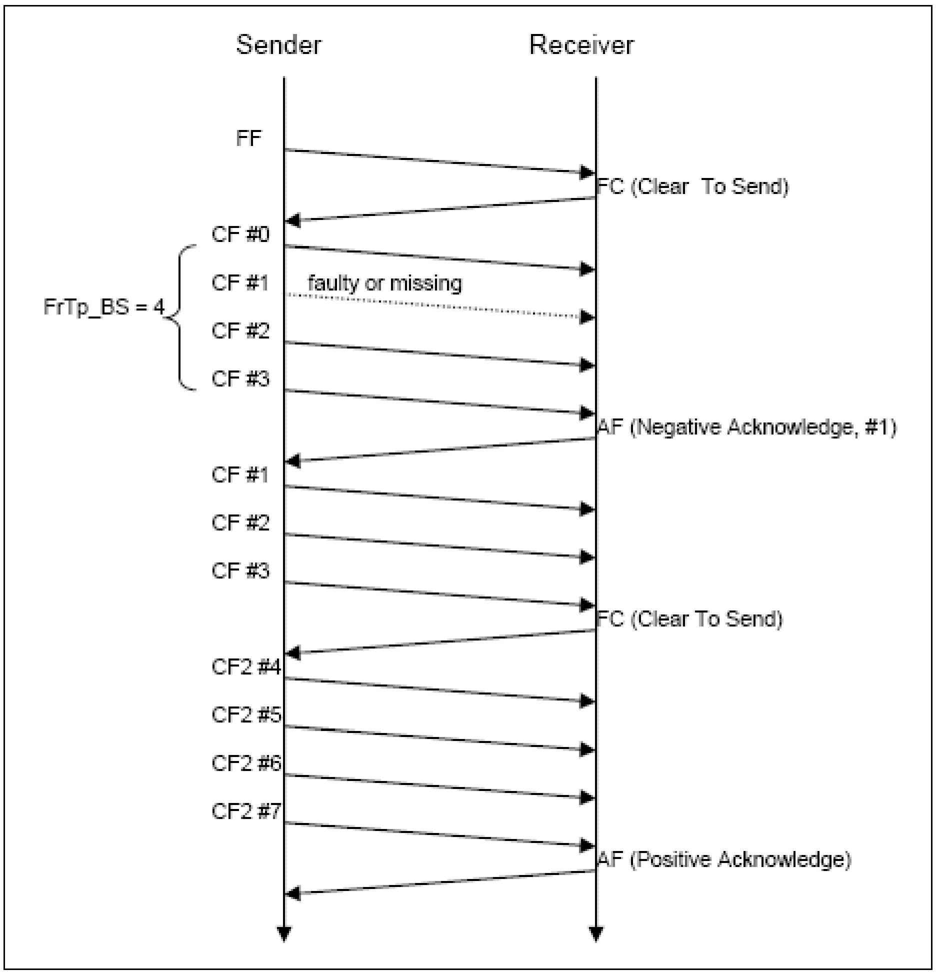


Figure 9: Segmented 1:1 transfer with Acknowledgement with Retry

If the retry starts with a lower sequence number than requested, this shall be tolerated, i.e. all frames until the requested shall be ignored and errors within the ignored frames shall be ignored, too. If it starts with a higher number than requested, this shall lead to another negative acknowledgement after the block end.

Alternating Block Mechanism

When using the Retry mechanism the FlexRay TP transfers blocks using the Alternating Block Mechanism. This works as follows:
The first block is transferred using normal CF frames. The second block is transferred using CF2 frames, the third one with CF frames and so on. When a retry occurs, a CF block is again transferred with CF frames and, of course, a CF2 block is retried with CF2 frames.

This mechanism ensures correct behavior in case at the block end an FC frame is lost, especially if it is an FC with flow status CTS, by allowing the detection of the unnecessary retries.

7.2.2 1:n Connections

In the case of 1:n connections (1 sender, multiple receivers) there is no further distinction in subtypes (with or without acknowledgement). The reason for this is that the size of the receiving group is often not known a priori, so it is not possible to apply flow control or acknowledgement mechanisms to 1:n connections.
Therefore, the only distinction made is between unsegmented and segmented transfer.
1:n connections are unidirectional by nature.

Unsegmented Transfer
This is exactly the same like in the section Unsegmented Transfer of section 7.2.1.1.
The only difference is the multiple receivers instead of one. Thus, the procedure looks like the following:

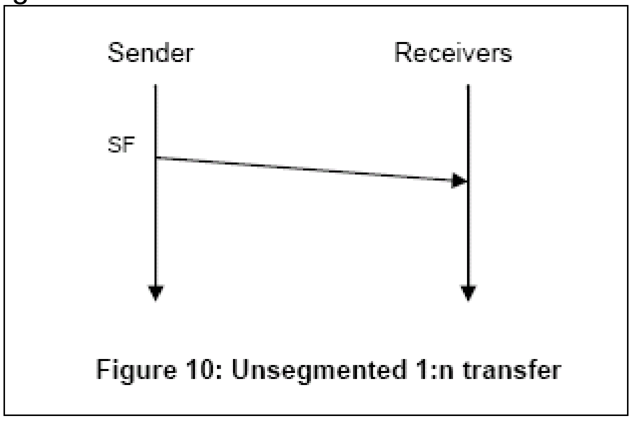


Figure 10: Unsegmented 1:n transfer

One sender sends its message to a group of receivers.

Segmented Transfer **Not compliant to ISO 15765-2**
Since no flow control or acknowledgement is possible in this case, a segmented 1:n transfer only consists of a FF and the number of necessary CFs

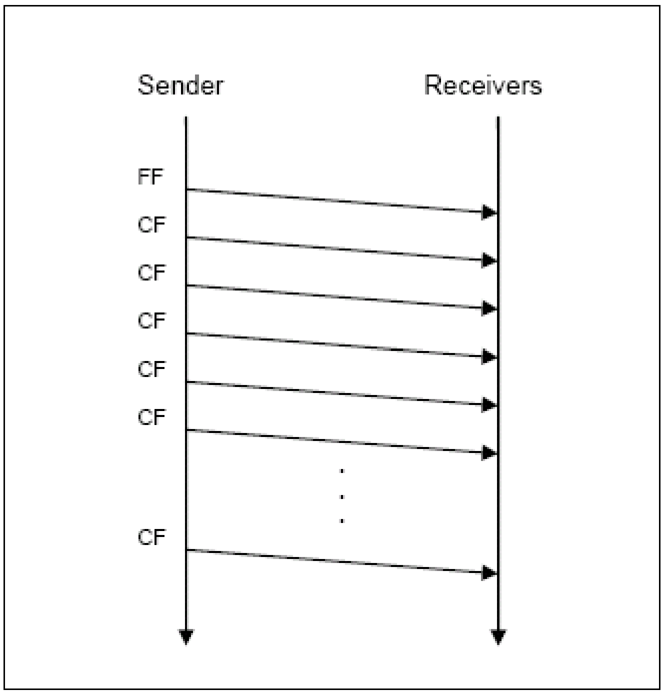


Figure 11: Segmented 1:n transfer

When an error occurs, the reception will be terminated and the appropriate result will be given within *PduR_FrTpRxIndication()*.

The distance of consecutive CFs is defined by the configuration parameter FrTpStMinGrpSeg, similar to FrTpStMin for segmented 1:1 connections.

7.3 Frame Layout

As seen in section 7.2 there are different types of frames. A detailed explanation of all the types follows below.

7.3.1 General

The general structure of a frame is shown in Figure 12:

Address Information	Protocol Control Information	Data (Payload)

Figure 12: Structure of a FlexRay AUTOSAR Transport Layer frame

It is common to all frames that they are headed by address information. Depending on static configuration (per channel), in a way whether 1 Byte or 2 Byte addressing is used, this address information consists of 1 Byte for Target Address and 1 Byte for Source Address or 2 Bytes for Target address and 2 Bytes for Source Address. Since it depends on the interpretation of the address information, it is not further specified whether this address information is utilized for the in automotive area so called "Physical" or for "Functional" addressing.

FRTP255: When the FrTp frame does not require the whole length of its N-PDU (in a Single Frame, a First Frame, or the last Consecutive Frame in a transfer), the remaining space (bits) in the N-PDU shall be set to 0.

1 Byte Addressing Not compliant to ISO 15765-2

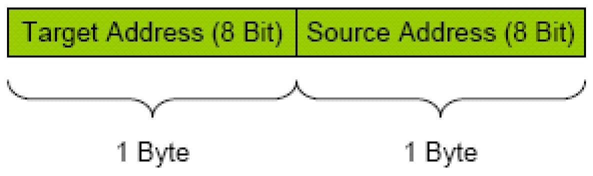


Figure 13: Address header for 1 Byte addressing

For both target and source address 1 Byte is provided, so up to 256 receivers are addressable.

2 Byte Addressing Not compliant to ISO 15765-2

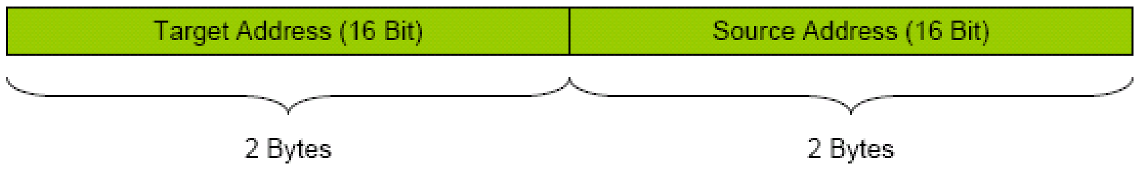


Figure 14: Address header for 2 Byte addressing

Looking at this scheme it is possible to address up to 65536 different receivers.

As seen in Figure 12, frames generally consist of the address information, protocol control information and the data. The length and content of the protocol control information (PCI) varies from frame type to frame type.

Before explaining the details of each frame, a short overview is given by the following table (the mentioned bytes and nibbles regard to the PCI):

FRTP021:

ISO 15765-2	***Name***	***1st Nibble***	***2nd Nibble***	***2nd Byte***	***3rd Byte***	***4th Byte***	***5th Byte***	***Description***
YES	SF-I	0x0	DL	data				ISO 15765-2 Single Frame
NO	SF-E	0x4	Res (0x0)	DL	data			Extended Single Frame
YES	FF-I	0x1	DL		data			ISO 15765-2 First Frame
NO	FF-E	0x5	Res (0x0)	DL				Extended First Frame
YES	CF	0x2	SN	data				ISO 15765-2 Consecutive Frame
NO	CF2	0x6	SN	data				Consecutive Frame used in Retry Channels
YES / NO	FC	0x3	FS	BS	STmin	--	--	(ISO 15765-2) Flow Control Frame
NO	AF	0x7	FS	BS	STmin	ACK (4 Bit) / SN (4 Bit)	--	Acknowledgement Frame

Table 1: Overview of the different frames format

Note: Unused bytes in this table shall be set to 0x00.

Endianness
In case a protocol value transmitted over the bus consists of more than 1 Byte (e.g. Source Address and Target Address when using 2-Byte addressing), the endianness shall be Most Significant Byte first, Least Significant Byte last.

7.3.2 Single Frames (SF-x)

FRTP022: A SF is sent when a message does not exceed the available amount of payload of this frame type or if ISO 15765-2 compliance is required. To be compliant with ISO 15765-2 on the one hand and to allow using the possibilities of FlexRay on the other hand, there are two types of Single Frames. In ISO 15765-2 compliant channels only SF-I is allowed, in non ISO 15765-2 compliant channels (i.e. FrTpLm = FRTP_L4G) only SF-E is allowed.

7.3.2.1 ISO 15765-2 Single Frame (SF-I)

FRTP023: A SF-I looks as follows (address information header is not depicted):

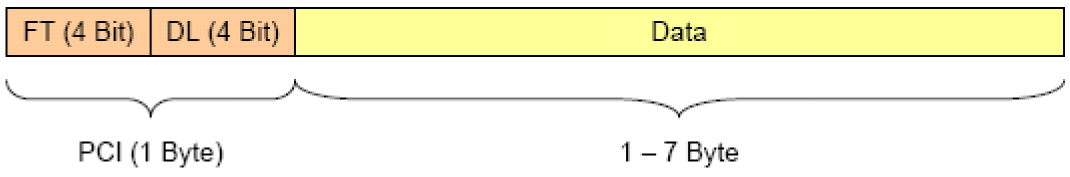


Figure 15: Single Frame ISO 15765-2

In a SF-I the PCI consists of only one byte. This byte is divided in two parts, called FT (Frame Type) and DL (Data Length). Both parts are 4 Bit long.

The FT field is common to every frame type because it identifies the respective type.

FRTP024: For ISO 15765-2 Single Frames the FT field shall be set to 0x0.

FRTP025: The DL field states the amount of the actual data bytes, according to ISO 15765-2 the values 0x1 – 0x7 (0x6 in FRTP_ISO6 mode) are valid, so in an ISO 15765-2 compliant connection the size of the associated N-PDU has to be, depending on the addressing mode, 10 (9) or 12 (11) Bytes long, since the SF has this length.

FRTP026: The actual frame length can be derived considering the addressing mode and looking in the length statement of the corresponding PduInfoType struct in the configuration.

FRTP027: Including address information the length of an SF-I reaches from 4 Byte (1 Byte payload, 1 Byte addressing) to 12 Bytes.

Error Handling

FRTP028: DL field:
Incoming SF-I frames with an invalid DL value of 0x0 or higher than 0x7 (0x6 in FRTP_ISO6 mode) shall be ignored. This shall also be done if a value arrives which is higher than the amount of payload that can be derived from the length statement of the corresponding PduInfoType struct in the configuration and the addressing mode.
If acknowledgement is configured, additionally an AF with a negative acknowledgement shall be sent back to the sender in the cases above.

7.3.2.2 Extended Single Frame (SF-E)

This section is **Not compliant to ISO 15765-2**.

FRTP029: An SF-E allows using the whole possible FlexRay payload of 254 Bytes for an un-segmented transfer. It looks as depicted in Figure 16:

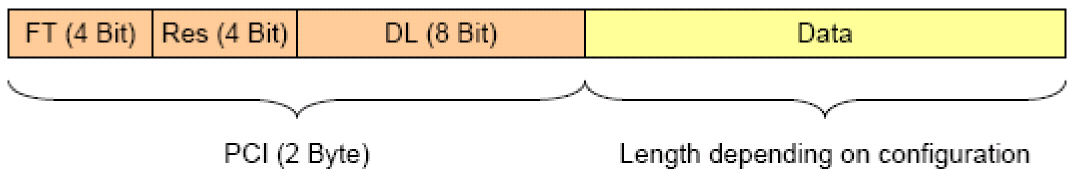


Figure 16: Single Frame Extended

FRTP030: The PCI of an SF-E consists of two bytes. The FT field is 4 Bit long; for an SF-E, it shall be set to 0x4. The following nibble is reserved; it shall be set to 0x0.

FRTP031: The next byte is the DL field and states the amount of payload contained in the SF-E. Depending on the configuration of the addressing mode (1 Byte or 2 Byte) and the length of the associated N-PDU, all values except 0x00 and above 0xFA (1 Byte addressing) or above 0xF8 (2 Byte addressing) are valid here.

FRTP032: The minimum length of such a frame is 5 Byte (1 Byte addressing, 1 Byte payload), the maximum is 254 Byte (FlexRay limit according to [14]). The actual frame length can be derived considering the addressing mode and looking in the length statement of the corresponding PduInfoType struct.

Error Handling

FRTP286: DL field:
If this field contains the value 0x00 or, depending on the addressing mode, a value higher than 0xFA or higher than 0xF8, the SF-E shall be ignored.
If acknowledgement is configured, additionally an AF with a negative acknowledgement shall be sent back to the sender.

FRTP287: General:
If messages longer than allowed by ISO 15765-2 are not configured (FrTpLm) for the corresponding channel, this frame shall be ignored. This shall also be done if a value arrives which is higher than the amount of payload that can be derived from the length statement of the corresponding PduInfoType struct and the addressing mode or if a value different from 0x0 arrives in the reserved nibble.
If acknowledgement is configured, additionally an AF with a negative acknowledgement shall be sent back to the sender in the cases above.

7.3.3 First Frames (FF-x)

If a message does not fit into a SF, it has to be segmented.

FRTP034: The FlexRay AUTOSAR Transport Layer takes the decision whether a message has to be segmented based on the message length, the possibility (depending on per channel configuration) to use SF-E frames and the size of the assigned N-PDU (see also section 7.4.1). Therefore, to start the transfer of such a long message, a First Frame is used.

FRTP035: To enable compliance with ISO 15765-2 on the one hand and to allow messages longer than 2^{12}-1 Byte on the other hand, there are several types of First Frames.

FRTP036: **Not compliant to ISO 15765-2**
It can be statically per channel configured, whether a First Frame can also start a segmented message in a 1:n connection.

7.3.3.1 First Frame ISO 15765-2 (FF-I)

FRTP237: The figure below shows the layout of a FF-I:

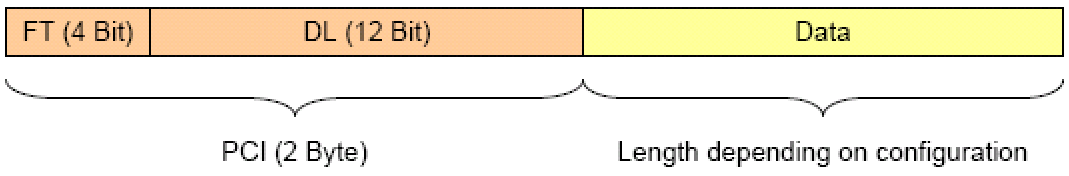


Figure 17: First Frame ISO 15765-2

In an FF-I the PCI consists of 2 Bytes. As in an SF, the FT field is 4 Bit long, the DL field 12 Bit.

FRTP037: For a FF-I, the FT field shall be set to 0x1.

The DL field contains the length of the whole message. Due to the 12 bit length of this field, messages up to 2^{12}-1 Bytes can be transferred.

FRTP038: The overall length of a First Frame including address information lasts (depending on the per channel configuration) from 4 Byte to a connection specific maximum.
This maximum on its part depends on the use case (e.g. for communication with CAN for which full ISO 15765-2 compliance is necessary, it will be 10 or 12 (9 or 11 in FRTP_ISO6 mode)) as well as on the size of the associated N-PDU. The actual amount of payload of an FF-I can be derived by considering the addressing type (1 or 2 Byte) and e.g. looking in the length designation of the corresponding PduInfoType struct.

FRTP039: Error Handling

DL field:
Incoming FF-I frames with DL = 0x000 shall be ignored. Moreover, if the DL value is lower than the possible (from the PDU size, the addressing type and the channel specific Long Messages switch derivable) payload of a SF, the frame shall also be ignored.
If acknowledgment is configured, in all the cases above additionally an AF with a negative acknowledgement shall be sent back to the sender.

7.3.3.2 First Frame Extended (FF-E)

This section is **Not compliant to ISO 15765-2**.

FRTP238: The layout of FF-E is shown below:

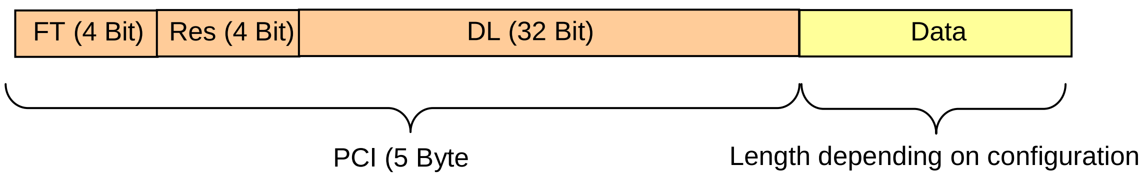


Figure 18: First Frame Extended

In an FF-E, the PCI consists of 5 Bytes. The FT field is 4 Bit long, 4 Bits are reserved, the DL field 32 Bit.

FRTP054: The DL field is 4 Byte long, so it allows transporting up to 2^{32}-1 bytes.

FRTP055: The FT field is set to 0x5.

FRTP056: The Res field (reserved) is set to 0x0.

The overall length of an FF-E reaches from 7 Byte to a connection specific maximum, which depends on the size of the associated N-PDU.

FRTP057: Error Handling

DL:
If the FR_DL value is lower than the possible (from the PDU size and the addressing type derivable) payload of an SF, the frame shall be ignored.
If acknowledgement is configured for the corresponding channel, an AF with a negative acknowledgement shall be sent back to the sender.

7.3.4 Consecutive Frames

FRTP058: If no error occurred, an FF-x is followed by Consecutive Frames until the whole message is transmitted.

FRTP059: **Not compliant to ISO 15765-2**
If configured for the specific channel, a Consecutive Frame can also appear in a 1:n connection.

FRTP239: As shown below, Consecutive Frames consist of one byte PCI and the payload.

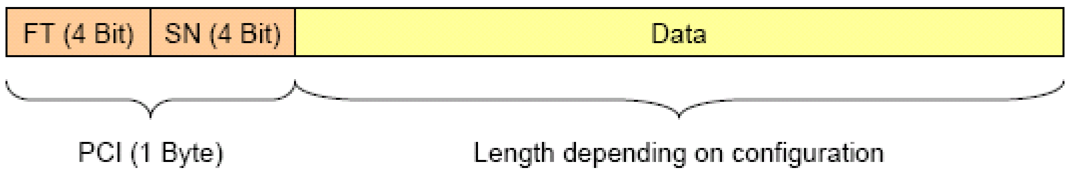


Figure 19: Consecutive Frame

The PCI of a Consecutive Frame consists of one byte, which is divided in two 4-bit parts.

FRTP060: The FT field again states the frame type, for a CF it shall be set to 0x2, for a CF2 it shall be 0x6 (CF2 frames are **Not compliant to ISO 15765-2**).

FRTP061: The SN (Sequence Number) field gives the current sequence number of the Consecutive Frame. **Please note that the SN of the CF that immediately follows the FF-x is set to 1** and then incremented with each frame until it wraps around to 0 and so on.

FRTP062: The overall length of a Consecutive Frame including address information ranges (depending on the per connection configuration) from 4 Byte to a connection specific maximum. This maximum depends on the use case (e.g. for communication with CAN for which full ISO 15765-2 compliance is necessary it will be 10 or 12 (9 or 11 in FRTP_ISO6 mode)) as well as on the size of the associated PDU.

The receiving peer can derive the actual data length by looking in the associated PduInfoType struct und considering the addressing mode.

Error Handling

FRTP063: SN field:
If no acknowledgement is configured, then in case of a wrong SN, i.e. after SN x does not follow SN x+1, the transfer shall be aborted and within *PduR_FrTpRxIndication* the result NTFRSLT_E_WRONG_SN shall be returned.
If acknowledgment is configured, after the block end, a negative acknowledgement shall take place and then the transfer shall be aborted as described above.

If Retry is configured too, then the transfer shall not be aborted but the Retry shall take place (up to FrTpMaxRn times).

7.3.5 Flow Control (FC)

FRTP064: A Flow Control frame is used in segmented 1:1 connections (see section 7.2.1.1). Thus, it cannot appear in a 1:n connection. It allows the receiver to send information to the sender. It is sent after reception of an FF-x and after the last CF of a block if no error occurred.

FRTP240: The layout of FC is shown below:

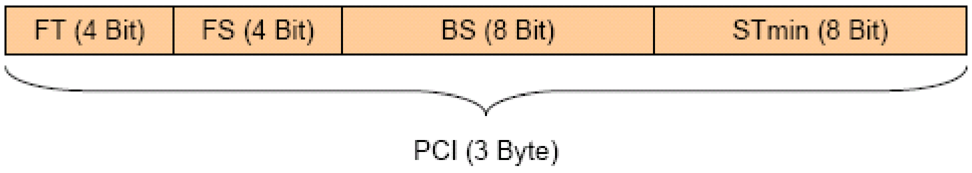


Figure 20: Flow Control frame

FRTP065: A Flow Control frame only consists of Protocol Control Information.

FRTP066: As usual, the FT field states the frame type, thus for Flow control frames, it shall be set to 0x3.

FRTP067: The FS field may contain the following three flow status values (see also section 7.2.1.1):

• CTS (value 0x0):	Clear To Send The sender can continue transmitting the message.
• WT (value 0x1):	Wait The sender shall wait for another FC frame (and therefore restart its timer FrTpTimeoutBs). If the number of consecutive Flow Control frames with FS = WT reaches a per channel defined maximum, the transfer shall be aborted.
• OVFLW (value 0x2):	Overflow The transfer shall be aborted, because the receiver has not enough buffer for the whole message available (according to the value of the DL field of the FF-x)

FRTP068: BS states the block size (the number of CFs between the Flow Control frames). If no acknowledgement is configured, all values from 0x00 to 0xFF are valid whereas 0x00 indicates that no more flow control shall take place and the rest of the

pending message will be transmitted within one big block. Otherwise, only the values 0x01 – 0x10 are valid. FrTp module shall use the buffer size provided by first call of the function PduR_FrTpProvideRxBuffer to calculate the BS.

FRTP069: The last byte contains STmin, which defines the minimum gap between two CFs. The valid values range from 0x00 to 0x7F and from 0xF1 to 0xF9. The range from 0x00 to 0x7F specifies the minimum gap in milliseconds (0ms .. 127ms), the one from 0xF1 to 0xF9 defines the gap in microseconds (100 µs, 200 µs, .. 900µs). The supported values of STmin are restricted by the placement of N-PDUs in FlexRay cycles, and are subject to the jitter created by the placement of N-PDUs in the slots of a cycle.

FRTP070: Depending on addressing configuration, a Flow Control frame is 5 or 7 byte long.

Error Handling
FRTP285: FS
If acknowledgment with Retry is configured, instead of abortion of the transfer, the frame shall be ignored.

FRTP244: BS
All values are valid if no acknowledgement is configured. Otherwise, only the values from 0x1 to 0x10 are valid. If no Retry is configured in the latter case, the transfer shall be aborted and *PduR_FrTpTxConfirmation* shall be called with NTFRSLT_E_NOT_OK, otherwise the frame shall be ignored.

FRTP245: STmin
The invalid values of this parameter range from 0x80 to 0xF0 and from 0xFA to 0xFF. If such a value is received, the value 0x7F shall be taken instead.

7.3.6 Acknowledgement Frame (AF)

This section is **Not compliant to ISO 15765-2**.

FRTP072: If acknowledgement is configured, every block of CFs is in the case of a positive acknowledgement acknowledged by an FC frame (as it is in unacknowledged connections). Additionally an SF-x, the last block of CFs is acknowledged by an AF in 1:1 connections, and, in the case of a negative acknowledgement, also other blocks. This frame type cannot appear in a 1:n connection.
This type of frame looks similar to an FC frame (section 7.3.5) but it has an additional byte.

FRTP241: The layout of FC is shown below:

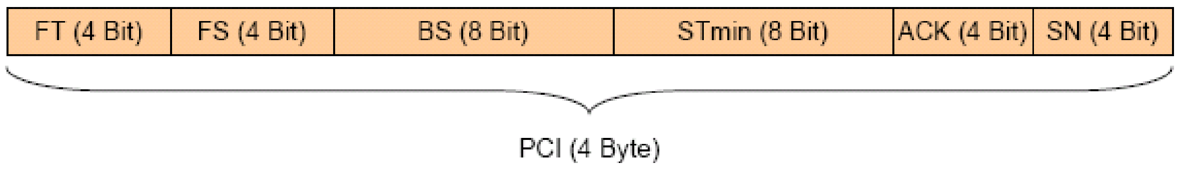


Figure 21: Acknowledgement Frame

FRTP073: This frame is identified by the value 0x7 of the FT field.

FRTP074: FS field is the same as in an FC frame.

FRTP075: BS field can only be set to the values 0x01 to 0x10 due to the 4 Bit Sequence Number counter in a CF (section 7.3.4).

FRTP076: STmin is the same as in FC frames.

FRTP077:ACK field gives the type of the acknowledgement, Positive (0x0) or Negative (0x1). All other values are reserved.

FRTP078: SN field contains the number of the first faulty CF within the last block. All values are valid.

FRTP079: Depending on addressing type, this frame is 6 or 8 Byte long.

Error Handling:
The following only holds if an AF arrives when it is expected. Otherwise, see section 7.3.7.

FRTP284: FS field:
In a segmented transfer, all values higher than 0x2 shall lead to the abortion of the transfer and *PduR_FrTpTxConfirmation* shall be called with the result NTFRSLT_E_INVALID_FS.

If additionally Retry is configured, such values shall not lead to the abortion of the transfer. Instead, the frame shall be ignored.

FRTP246: BS field:
The value 0x00 and all values higher than 0x10 shall cause the abortion of the transfer and *PduR_FrTpTxConfirmation* shall be called with the result NTFRSLT_E_NOT_OK.
If additionally Retry is configured, such values shall not lead to the abortion of the transfer. Instead, the frame shall be ignored.

FRTP247: STmin field:
The invalid values of this parameter range from 0x80 to 0xF0 and from 0xFA to 0xFF. If such a value is received, the value 0x7F shall be taken instead.

FRTP248: ACK field:
Values higher than 0x1 are invalid and shall cause the abortion of the transfer and *PduR_FrTpTxConfirmation* shall be called with the result NTFRSLT_E_NOT_OK.
If additionally Retry is configured, such values shall not lead to the abortion of the transfer. Instead, the frame shall be ignored.

FRTP249: SN field:
If a frame arrives that contains an SN of a CF that has not been transmitted within the block, e.g. block size is 10 and this field has value 12, the transfer shall be aborted and *PduR_FrTpTxConfirmation* shall be called with the result NTFRSLT_E_WRONG_SN.
If additionally Retry is configured, the transfer shall not be aborted but the frame shall be ignored.

FRTP250: General:
If for the channel no acknowledgement is configured, this frame type shall be ignored.
In an unsegmented acknowledged transfer, the expected value for the fields BS, STmin, and SN is 0x0. Other values shall be tolerated.

FRTP251: For the FS field there is an exception: In case an AF with negative acknowledgement and FS = OVFLW arrives in an **unsegmented** acknowledged transfer or at the end of an segmented acknowledged transfer at the sender, regardless of Retry being configured or not, the transfer shall be aborted and *PduR_FrTpTxConfirmation* shall be called with the result NTFRSLT_E_INVALID_FS.

For a better understanding, the following table depicts the possible combinations (and their meaning) of the FS and ACK field in an Acknowledgment Frame:

Possible combinations of FS and ACK field in Acknowledgement Frames	**ACK = 0x0**	**Meaning / Appearance**	**Leads to Retry (if configured)**	**ACK = 0x1**	**Meaning / Appearance**	**Leads to Retry (if configured)**
FS = CTS	X	Positive Acknowledge after SF or after end of Segmented Transfer	NO	X	Negative Acknowledge after SF or after block end in Segmented Transfer	YES
FS = WT	--	--	NO	X	Negative Acknowledge after SF (if currently no Receive buffer is available)	NO
FS = OVFLW	--	--	NO	X	Negative Acknowledge after SF (if no Receive buffer is available)	NO

Table 2: Possible combinations of FS and ACK field

7.3.7 Error Handling of the FT Field

Not every frame type is accepted at any point in time and in any configuration of a channel/connection. Thus, a detailed description is given below.

FRTP082: A value of the FR_FT field higher than 0x7 shall always be ignored.

FRTP083: If the corresponding channel and connection is set to be ISO 15765-2 compliant, then the following table holds:

TP Layer Status	***SF-I***	***FF-I (1:1)***	***CF (1:1)***	***FC***	***Other***
Segmented Transmit within this connection in progress	If reception is in progress within the connection, see corresponding cell below. Otherwise, process the SF-I as start of a new reception.	If reception is in progress within the connection, see corresponding cell below. Otherwise, process the FF-I as start of a new reception.	If reception is in progress within the connection, see corresponding cell below. Otherwise, ignore it.	If awaited then process, otherwise ignore it.	Ignore
Segmented Receive within this connection in progress	Terminate the current reception, report a *PduR_FrTpRxIndication* with the result NTFRSLT_E_UNEXP_PDU and process the SF-I as the start of a new reception.	Terminate the current reception, report a *PduR_FrTpRxIndication* with the result NTFRSLT_E_UNEXP_PDU and process the FF-I as the start of a new reception.	If awaited then process, otherwise ignore	If transmission is in progress within the connection, see corresponding cell above. Otherwise ignore it	Ignore
Idle	Process the SF-I as the start of a new reception	Process the FF-I as the start of a new reception	Ignore	Ignore	Ignore

Table 3: FT Error Handling in ISO 15765-2 compliant channels/connections

Otherwise, the behavior is explained below:

FRTP283: SF-x, FF-x, CF/CF2 and FC:
The behavior shall be as depicted in Table 3 (also for 1:n connections).

The ignoring of an FF-E shall be according to the value of FrTpLm.

Regarding CF and CF2 frames there is a special error handling in case Retry is configured (otherwise CF2 frames are ignored):
If the sender starts a block with another frame than expected, i.e. CF instead of CF2 or CF2 instead of CF, then the sender is doing a Retry that has not been requested by the receiver (maybe because of losing the FC-CTS frame on the bus). Therefore, the receiver always has to remember the old block size and send another FC-CTS at the end of this retransmitted block. Errors in the unnecessarily retransmitted block shall be ignored.

FRTP252: AF
If no acknowledgement is activated, this frame shall be ignored. Otherwise, on the receiver side or in idle state, these frames shall be ignored, too.

On the sender side, the behavior in case of an incoming AF shall be the following:

- **FRTP269:** If an AF arrives when it is expected, the action is as described in section 7.2.1.3 and in section error handling of section 7.3.6.
- **FRTP270:** If a non-faulty AF with positive acknowledgement arrives during a block, it shall be ignored.
- **FRTP271:** If a non-faulty AF with negative acknowledgement arrives during a block, it shall be processed depending on the activation of the Retry mechanism. If no Retry is configured the transfer shall be aborted. Otherwise, the AF shall be processed, i.e. starting with the stated sequence number the Retry shall take place.
- **FRTP272:** If a faulty AF arrives during a block, it shall be ignored.

7.3.8 Addressing Errors

SF-x:
No restrictions.

FRTP086: FF-x and CF
If not explicitly configured by the parameter FrTpGrpSeg for the particular channel, a FF-x or CF in a 1:n connection shall be ignored.

FRTP087: FC and AF
These frame types are not allowed to appear in 1:n connections; thus, they shall be ignored in that case.

7.4 Further Principles of Working

7.4.1 Decision of Segmentation

As mentioned earlier in this specification, there are several factors influencing the decision of the FrTp to segment a message (N-SDU) or not.
The values of the following parameters play a role hereby:
N-PDU length, FrTpLm, FrTpAdrType, FrTpMultRec, FrTpGrpSeg, and the length of the to-be-transmitted message (N-SDU).

FRTP091: The amount of bytes of an N-PDU that is usable for payload, i.e. for the N-SDU, depends on the length of the PCI of the used frames, i.e. if two or four bytes (FrTpAdrType) are needed to state to address information. The frames, which are allowed to be utilized, and the payload they can carry depend on the value of FrTpLm (e.g. SF-E is allowed or not, SF-I can carry 7 or 6 bytes etc.). In case the connection

is a 1:n connection (FrTpMultRec), the parameter FrTpGrpSeg states whether segmentation is allowed or not.
With all this information and the length of the to-be-transmitted N-SDU, the FrTp can decide whether it has to segment the N-SDU or not.

7.4.2 Scheduling of PDUs during Transmission

PDUs of a PDU pool must be assigned to all active connections in a way that no connection freezes, while connections with prioritized PDUs are served first.

FRTP256: To achieve an even distribution of PDUs to all currently transmitting and/or receiving connections associated with a PDU pool, the PDUs of this pool shall be assigned to active connections using round robin scheduling. A connection that is currently receiving and transmitting may claim two PDUs in one round of the assignment: one for the FC, and one for an SF/FF/CF.

FRTP257: The scheduling shall be executed in the context of the main function, and shall start with the connection where the scheduling stopped in the previous cycle.

FRTP258: Each PDU assignment cycle shall start with the prioritized PDUs. In this phase, PDUs are only assigned to active connections with prioritized PDUs, until their needs are satisfied. Afterwards, PDU assignment continues for all active connections.

FRTP259: It must be ensured that the last PDU of a PDU pool within a FlexRay cycle is always used by the scheduling.

FRTP260: If not all PDUs of the pool are used, the positions of the unused PDUs are not relevant; gaps are allowed in any place.

7.4.3 Detection of Receiving Connection

When an SF-x or FF-x frame is received, the N-PDU-ID is used to identify the relevant pool. Because a PDU pool may only be used by channels with identical addressing type, the address information can be extracted from the N-PDU, by which the receiving connection can be identified, because no two connections using the same PDU pool have identical addresses.

FRTP261: If the receiving connection is not active, and all state machines of the associated channel are in use, the incoming message shall be ignored.

7.4.4 Single Frame Handling during Reception

FRTP262: No state machine shall be used for the reception of an SF-x. When the corresponding connection is free, the single frame shall be forwarded immediately by calling PduR_FrTpProvideRxBuffer and, upon successful return of this function, PduR_FrTpRxIndication.

7.4.5 Sending and Receiving within the same connection

FRTP094: The FlexRay AUTOSAR Transport Layer shall be implemented to support both sending and receiving within one connection. So the same connection can be utilized for sending and receiving.
To explain it more in detail, imagine a connection being in idle state. If now the call *FrTp_Transmit()* occurs, the local peer becomes the sender in this connection (Source Address of TP frame = FrTpLa, Target Address of TP frame = FrTpRa). Otherwise, if an *FrTp_RxIndication()* occurred for an N-PDU which is mapped on the N-SDU of this connection, it would become the receiver (Source Address of TP frame = FrTpRa, Target Address of TP frame = FrTpLa).

This feature is intended for connections in which sometimes one peer has to send data and sometimes the other in order not to need two connections in this case.

7.4.6 Behavior on Timeouts and Errors

FRTP095: The behavior in case a timeout occurs depends on the value of FrTpAckType, i.e. what kind of acknowledgement is configured for the corresponding channel.

FRTP291: The FrArTp shall abort the connection when FrIf_Transmit returns E_NOT_OK.

7.4.6.1 No Acknowledgement configured for the Channel

In this case, the behavior shall be as described in [12], i.e.:

- **FRTP282:** If the AS timer (FrTpTimeoutAs) expires, depending on the value of FrTpMaxAs, sending shall be retried (because of still remaining attempts) or the transmission shall be aborted and within *PduR_TxConfirmation* the result NTFRSLT_E_TIMEOUT_A shall be returned.
- **FRTP263:** If the AR timer (FrTpTimeoutAr) expires, depending on the value of FrTpMaxAr, sending shall be retried (because of still remaining attempts) or the transmission shall be aborted, and within *PduR_RxIndication* the result NTFRSLT_E_TIMEOUT_A shall be returned.
- **FRTP264:** If the BS timer (FrTpTimeoutBs) expires, the transmission shall be aborted, and within *PduR_TxConfirmation* the result NTFRSLT_E_TIMEOUT_BS shall be returned.
- **FRTP265:** If the CR timer (FrTpTimeoutCr) expires, the transmission shall be aborted, and within *PduR_RxIndication* the result NTFRSLT_E_TIMEOUT_CR shall be returned. If previously in the current block a sequence error occurred, this error will be reported at the block end in *PduR_RxIndication*.

7.4.6.2 Acknowledgement without Retry configured for the Channel

This section is **Not compliant to ISO 15765-2**.

In this case, the behavior is the following:

- **FRTP281:** In case of a timeout of timer AS, AR or BS, the behavior shall be as mentioned in section 7.4.6.1.
- **FRTP266:** If the CR timer (FrTpTimeoutCr) expires, an AF with negative acknowledgement shall be sent, the transmission shall be aborted, and within *PduR_RxIndication* the result NTFRSLT_E_TIMEOUT_CR shall be returned. If previously in the current block a sequence error occurred, this error will be reported at the block end in *PduR_RxIndication*.

7.4.6.3 Acknowledgement with Retry configured for the Channel

This section is **Not compliant to ISO 15765-2**.

In this case, the behavior shall be the following:

- **FRTP280:** In case of a timeout of timer AS or AR, the behavior shall be as mentioned in section 7.4.6.1.
- **FRTP267:** If the BS timer (FrTpTimeoutBs) expires, the sender shall retransmit the whole block up to FrTpMaxRn times. After that, the transmission shall be aborted, and within *PduR_TxConfirmation* the result NTFRSLT_E_TIMEOUT_BS shall be returned.
- **FRTP268:** If the CR timer (FrTpTimeoutCr) expires, the receiver shall send an AF with negative acknowledgement and the sequence number of the missed CF. This shall be done up to FrTpMaxRn times. After that, the transmission shall be aborted, and within *PduR_RxIndication* the result NTFRSLT_E_TIMEOUT_CR shall be returned. If previously in the current block a sequence error occurred, at the block end this error will be reported in *PduR_RxIndication*.

7.4.7 Transmit Cancellation

FRTP099: This feature can be (de)activated by static configuration (parameter FrTpTc). Transmit Cancellation is triggered by the call of *FrTp_CancelTransmit*.

FRTP236: When a transmission is still in progress, FrTp_CancelTransmit shall return E_OK, and the transmission shall be stopped. When a connection is not active, or when the last N-PDU of a transmission without acknowledgement has already been forwarded to the FrIf, FrTp_CancelTransmit shall return E_NOT_OK.

The service works at the sender side of a connection as follows:

- **FRTP279:** If no transmit request is pending for the corresponding connection, there is nothing to do.
- **FRTP273:** If a request is pending but the transmission has not been started, the FrTp shall immediately call PduR_FrTpTxConfirmation and free the connection.
- **FRTP274:** If the transmission already has been started, the FrTp shall immediately call PduR_FrTpTxConfirmation, and remember that the N-PDUs that have already been allocated for this connection cannot be used again before they have been confirmed. When requested via TriggerTransmit, the pending N-PDUs shall be transferred to the FrIf as if they had not been canceled, or E_NOT_OK shall be returned.

FRTP103: If a transfer was cancelled by the call of *FrTp_CancelTransmit*, *PduR_FrTpTxConfirmation* shall be called with NTFRSLT_E_NOT_OK.

7.4.8 Receive Cancellation

FRTP224: If development error detection is enabled the function
FrTp_CancelReceive shall check the validity of FrTpRxSduId parameter.

FRTP225: If the FrTpRxSduId parameter value is invalid, the FrTp_CancelReceive function shall raise the development error FRTP_E_INVALID_PDU_SDU_ID and return E_NOT_OK.

FRTP226: The FrTp shall abort the reception of the current N-SDU if the
service FrTp_CancelReceive provides a valid FrTpRxSduId.

FRTP227: The FrTp shall reject the request for receive cancellation by returning E_NOT_OK when

a) the cancelled connection is not active, or when
b) the FrTp has already received the last frame of an unacknowledged connection, or when
c) the FrTp has already provided the final AF of an acknowledged connection.

FRTP228: If the FrTp_CancelReceive service has been successfully executed, the FrTp shall call the PduR_FrTpRxIndication with NTFRSLT_E_NOT_OK.

7.4.9 Parameter Changing

FRTP104: The FlexRay AUTOSAR Transport Layer also supports the optional service for changing the parameter FrTpStMin mentioned in [12] via the API call *FrTp_ChangeParameter*. A change is not possible during an ongoing reception and shall lead to the return value E_NOT_OK.
Please note that against [12] only the value of STmin is changeable. This comes from the buffer-requesting concept of AUTOSAR, which requires an automatic choosing of the block size by the FrTp software module.

7.4.10 Buffer Requests, Block Size and WAIT-Frames

FRTP105: Depending on the message length and configuration of the FrTp, a segmented or an unsegmented transfer will take place.

7.4.10.1 Unsegmented Transfer

FRTP106: At the sender side, this principle works as follows:

1. PduR calls the service *FrTp_Transmit*.
2. The FrTp shall then call *PduR_FrTpProvideTxBuffer* (value 0 for Length which means transmit data can be of arbitrary size) in order to get all the data bytes of the SF-x.

FRTP107: If not all data bytes are contained within the first Tx buffer, the service *PduR_FrTpProvideTxBuffer* will be called repeatedly (always with value 0 for the Length parameter) until all data bytes are put into the SF. If calling the service does not provide a valid data length, it is tried until FrTpTimeCs expires to get one (if the call returned BUFREQ_E_BUSY) or the transfer shall be aborted (*PduR_FrTpTxConfirmation* shall be called with NTFRSLT_E_NO_BUFFER).

FRTP108: At the receiver side, the principle works as follows:

1. FrIf calls the service *FrTp_RxIndication*.
2. The FrTp shall then call *PduR_FrTpProvideRxBuffer* in order to get a buffer for the to-be-received SF-x.

FRTP289: If *PduR_FrTpProvideRxBuffer* provides a valid buffer, it will be filled with data. If the buffer is too small another one will not be requested and the transfer shall be aborted and PduR_FrTpRxIndication shall be called with NTFRSLT_E_NO_BUFFER.

FRTP253: If acknowledgement is configured in case of failing to get a receive buffer (either BUFREQ_E_OVFL was returned or BUFREQ_E_NOT_OK), an AF with a negative acknowledgement and FS = OVFLW is sent back to the sender.

7.4.10.2 Segmented Transfer

FRTP110: At the sender side, this principle works as follows:

1. PduR calls the service *FrTp_Transmit*.
2. The FrTp shall then call *PduR_FrTpProvideTxBuffer* in order to get the data bytes of the FF and following CFs.

FRTP111: When calling *PduR_FrTpProvideTxBuffer*, the value of the API parameter Length is important.

If no Retry is configured, the value 0 shall be used, since it does not matter what size the transmit data has because no data have to be kept in store for a retry.
If Retry is configured, things behave differently. Before requesting a transmit data, the sender sends an FF-x (without data bytes) to the receiver and waits for an FC frame to get knowledge about the block size (and the STmin value). The sender then can request a transmit data with an appropriate value for Length, i.e. the amount of data bytes which will be transferred in this block. This is necessary to have all the data available in case a Retry takes place. Since the sender will get a transmit data with the requested size, there will be no additional data in the FrTp necessary.
Since the requested transmit data can theoretically go up to 4016 bytes (when the receiver has that much receive buffer and the N-PDUs are configured to maximum FlexRay payload length of 254 bytes), the parameter FrTpMaxBs allows limiting the block size to a defined maximum per channel.

FRTP112: After transmitting the block, the sender waits for the next FC and can then request a data according to the new block size. The FrTp shall request for transmission of data as long as there are data bytes to send.

FRTP113: If calling the service does not provide a valid data length, it is tried until FrTpTimeCs expires to get one (if the call returned BUFREQ_E_BUSY) or the transfer shall be aborted if the call returns BUFREQ_E_NOT_OK or BUFREQ_E_OVLW (*PduR_FrTpTxConfirmation* shall be called with NTFRSLT_E_NO_BUFFER).

FRTP114: At the receiver side, this principle works as follows:

1. FrIf calls the service *FrTp_RxIndication*.
2. The FrTp shall then call *PduR_FrTpProvideRxBuffer* in order to get a buffer for the FF and the first block of CFs.

FRTP115: Depending on the size of the buffer returned by *PduR_FrTpProvideRxBuffer*, the FrTp uses a block size which makes the transferred data bytes within the upcoming block to fit into the provided receive buffer. The provided buffer will have a size which is greater or equal to x * Payload_in_CF + Payload_in_FF (where X is the number of CFs sent in the block). The block size, which is derived from the length of the provided buffer, is then sent to the sender within an FC frame.

FRTP290: When *ProvideRxBuffer* returns BUFREQ_E_NOT_OK or BUFREQ_E_OVFLW, FrTp shall terminate the session by calling *PduR_FrTpRxIndication* with NTFRSLT_E_WFT_OVRN, but only after at least one call to *ProvideRxBuffer* succeeded

FRTP292: When the buffer returned by the first ProvideRxBuffer is too small for the FF, the connection shall be aborted and PduR_FrTpRxIndication should be called with NTFRSLT_E_NO_BUFFER. An AF with a negative acknowledgement and FS = OVFLW is sent back to the sender.

FRTP293: The FrTp module shall call the function PduR_FrTpProvideRxBuffer after copying the FF to get the next block of data for the following CFs and in each call (except first call) the function PduR_FrTpProvideRxBuffer shall return the available receiver buffer.

FRTP117: In case *PduR_FrTpProvideRxBuffer* fails, the remaining CFs of the current block shall be discarded. When the failure occurred in the last block, and acknowledgement is enabled, an AF with a negative acknowledgement and FS = OVFLW shall be sent back to the sender. Otherwise, an FC with FS = OVFLW shall be sent back, but only if the first call to *PduR_FrTpProvideRxBuffer* returned BUFREQ_E_OVFL.

7.4.10.3 Buffer Locking

At the sender side the provider of the transmit buffer (e.g. DCM or AUTOSAR COM) shall not access the buffer, until the next one has been requested (by *PduR_FrTpProvideTxBuffer*) or transmission has been completed (i.e. calling of *PduR_FrTpTxConfirmation*).

At the receiver side the provider of the receive buffer (e.g. DCM or AUTOSAR COM) shall not access a buffer, until the next one has been requested (by *PduR_FrTpProvideRxBuffer*) or reception has been completed (i.e. calling of *PduR_FrTpRxIndication*).

7.4.10.4 Data Bytes in First Frames

FRTP120: **Not compliant to ISO 15765-2**
As stated in 7.4.10.2, if acknowledgement with Retry is configured for the corresponding channel, no payload is sent within an FF-x if a segmented transfer takes place. This is necessary because the FF-x is sent before the request for a Tx buffer (because the FrTp needs the block size from the receiving peer in order to call *PduR_FrTpProvideTxBuffer* with the correct value for Length).

FRTP121: If acknowledgment without Retry (or no acknowledgement) is configured, there are data bytes within an FF-x.

7.4.11 Ignored Frames

FRTP139: Throughout this specification, many times the ignoring of frames is mentioned. Please note that an ignored frame does never affect a timer, i.e. never causes the restarting of a timer.

FRTP140: The only exception is at the receiver side when retry is configured and due to an erroneous frame an AF with negative acknowledgement is sent and therefore it is waited for the retry frame(s). In this case, the timer CR will be reset by the erroneous frame.

7.5 Buffer Access Modes in the FlexRay Interface

FRTP187: The FlexRay AUTOSAR Transport Layer software module shall be implemented being able to work both with N-PDUs configured (in the FlexRay Interface) for Immediate Buffer Access and for Decoupled Buffer Access, i.e. it shall reuse its channel specific temporary buffers, in case the local peer is the sender, not before the TxConfirmation for the respective PDU pool has arrived.
In the receiving case, from the FlexRay AUTOSAR Transport Layers point of view, there is no difference between an N-PDU being configured for Decoupled Buffer Access or Immediate Buffer Access.

7.6 Error classification

This section lists and classifies all the errors that can be detected within this software module.

FRTP179: Error classification table

Type or error	***Relevance***	***Related error code***	***Value***
API service called while module is not initialized	Development	FRTP_E_UNINIT	0x1
API service called with invalid pointer	Development	FRTP_E_PARAM_POINTER	0x2
API service called with invalid SDU or PDU ID	Development	FRTP_E_INVALID_PDU_SDU_ID	0x3

Table 4: Error Classification

7.7 Error detection

FRTP217: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time.
The switch *FRTP_DEV_ERROR_DETECT* (see chapter 10) shall activate or deactivate the detection of all development errors.

FRTP205: If the *FRTP_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in section 7.6 and chapter 8.

7.8 Error notification

FRTP206: Detected development errors shall be reported to the *Det_ReportError* service of the Development Error Tracer (DET) if the pre-processor switch *FRTP_DEV_ERROR_DETECT* is set (see chapter 10).

8 API specification

8.1 Imported types

FRTP141: The following types are defined within AUTOSAR and used for the FlexRay AUTOSAR Transport Layer:

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

8.2 Type definitions

FRTP288: The following FrTp specific types shall be defined in FrTp_Types.h:

Name:	`FrTp_ConfigType`	
Type:	`Structure`	
Range:	`implementation specific`	--
Description:	This is the base type for the configuration of the FlexRay Transport Protocol. A pointer to an instance of this structure will be used in the initialization of the FlexRay Transport Protocol. The outline of the structure is defined in chapter 10 Configuration Specification.	

8.3 Function definitions

This is a list of functions provided for upper layer modules.

Here is the API Naming convention for the FrTp services:

- The service name format is FrTp_<ServiceName>(…)
- <ServiceName>: is the name of the service primitive with first letter of each word upper case and consecutive letters lower case

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.	

FRTP202: This service returns the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers (BSW00407).

This function shall be pre compile time configurable `On/Off` by the configuration parameter: `FRTP_VERSION_INFO_API`

Configuration: `FRTP_VERSION_INFO_API`

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 the FlexRay Transport Layer and sets all states to idle.	

Please note: The call of this service is mandatory before using the FrTp for further processing.

8.3.2.2 FrTp_Shutdown

FRTP148:

Service name:	FrTp_Shutdown	

Syntax:	`void FrTp_Shutdown(`
	
	`)`	

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 FrTpTxPduId,`
	`    const PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrTpTxPduId	This parameter contains the unique identifier of the Fr N-SDU to be transmitted.
	PduInfoPtr	A pointer to a structure with Fr N-SDU related data: data length and pointer to an Fr N-SDU buffer.
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. because the parameter check has failed (invalid N-SDU ID or size), the corresponding connection is still occupied, or no state machine is free (FrTpConcurrentConnections).
Description:	This service is utilized to request the transfer of data. This function has to be called with the PDU-Id of the FrTp, i.e. the upper layer has to translate its own PDU-Id into the one of the TP for the corresponding message. Within the provided PduInfoPtr only SduLength is valid (no data)! If this function returns E_OK then there will arise an call of PduR_FrTpProvideTxBuffer in order to get data for sending.	

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 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: Cancellation request of the transfer of the specified Fr N-SDU is accepted. E_NOT_OK: Cancellation request of the transfer of the specified Fr N-SDU is rejected.
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. This function has to be called with the PDU-Id of the FrTp, i.e. the upper layer has the same PDU-Id as for the FrTp_Transmit() call.	

Please note: When a transfer is successfully cancelled, the function PduR_FrTpTxConfirmation will be called with NTFRSLT_E_NOT_OK.

8.3.3.3 FrTp_CancelReceive

FRTP229:

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.	

Please note: This function has to be called with the PduId defined the receiving upper layer. The FrTp has to reverse map this to the corresponding connection.

8.3.3.4 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	Identifier of the received N-SDU on which the reception parameter has to be changed.
	parameter	Specify the parameter which shall be changed (currently only STmin is supported).
	value	The new value of the parameter. 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:	This service is used to request the change of reception parameter STmin for a specified N-SDU.	

Caveats: According to ISO 15765-2, it is not possible to change a parameter value during an ongoing reception.

8.4 Call-back notifications

8.4.1 FrTp_TriggerTransmit

FRTP154:

Service name:	FrTp_TriggerTransmit	

Syntax:	`Std_ReturnType FrTp_TriggerTransmit(`
	`    PduIdType FrTxPduId,`
	`    PduInfoType* PduInfoPtr`
	`)`	

Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTxPduId	ID of FlexRay N-PDU that is requested to be transmitted.
Parameters (inout):	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 (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:	This function is called by the FlexRay Interface for sending out a FlexRay frame. The trigger transmit is initiated by the FlexRay schedule.	

	This function is called with the PDU ID of the FrIf.

Please note: This function might be called in interrupt context

8.4.2 FrTp_RxIndication

FRTP152:

Service name:	FrTp_RxIndication	

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

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrRxPduId	This parameter contains the identifier of the received Fr N-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:	The FlexRay Interface calls this primitive after the reception of an Fr N-PDU. Within this function, the FlexRay Transport Layer at least copies the received Transport Layer frame to itself. This function has to be called with the PDU-Id of the FrTp, i. e. the FlexRay Interface has to translate its own PDU-Id into the corresponding one of the FrTp.	

8.4.3 FrTp_TxConfirmation

FRTP153:

Service name:	FrTp_TxConfirmation	
Syntax:	`void FrTp_TxConfirmation( PduIdType FrTxPduId )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTxPduId	This parameter contains the identifier of the transmitted Fr N-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called by the FlexRay Interface after the TP-related Pdus have been transmitted over the network. Within this function, the FlexRay TP shall route this confirmation to the configured target transport connection. This function is called with the PDU ID of the FrIf.	

8.5 Scheduled functions

8.5.1 FrTp_MainFunction

FRTP162:

Service name:	FrTp_MainFunction
Syntax:	`void FrTp_MainFunction( )`
Service ID[hex]:	0x10
Timing:	FIXED_CYCLIC
Description:	Schedules the FlexRay TP. (Entry point for scheduling)

Please note: This function is called directly by the Basic Software Scheduler (SchM).

Terms and definitions:
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 precondition: On precondition means that no cycle time can be defined. The function will be called when 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 section lists all interfaces required from other modules.

8.6.1 Mandatory Interfaces

This section defines all interfaces that are required to fulfill the core functionality of the module.

FRTP219:

API function	***Description***
FrIf_Transmit	Requests the sending of a PDU.
PduR_FrTpProvideRxBuffer	Provides Rx Buffer for the FlexRay TP
PduR_FrTpProvideTxBuffer	Provide Tx data for the FlexRay TP
PduR_FrTpRxIndication	Rx indicator for the FlexRay TP
PduR_FrTpTxConfirmation	Tx confirmation for the FlexRay TP

8.6.2 Optional Interfaces

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

FRTP220:

API function	***Description***
Det_ReportError	Service to report development errors.

8.6.3 Configurable interfaces

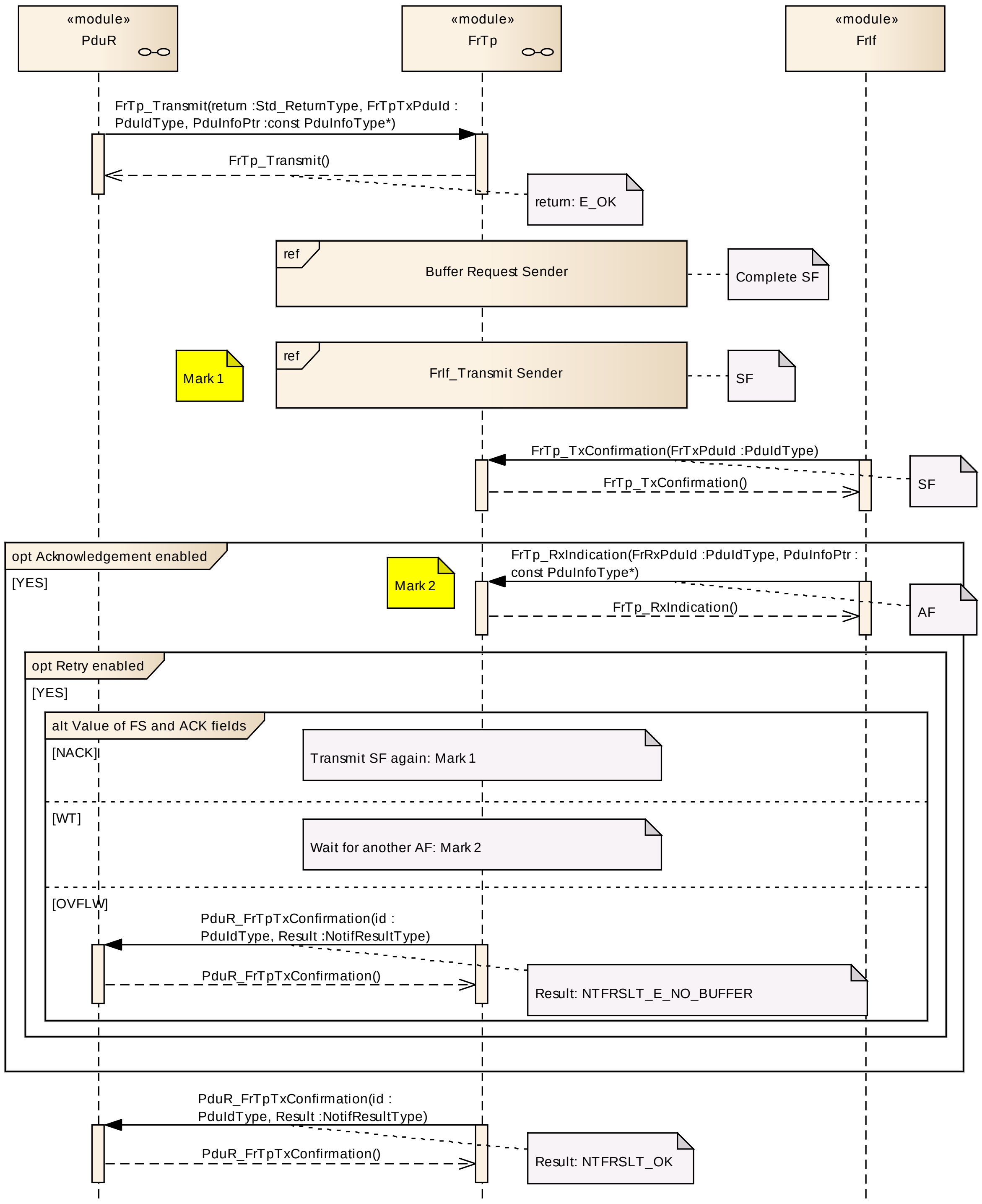
None

9 Sequence diagrams

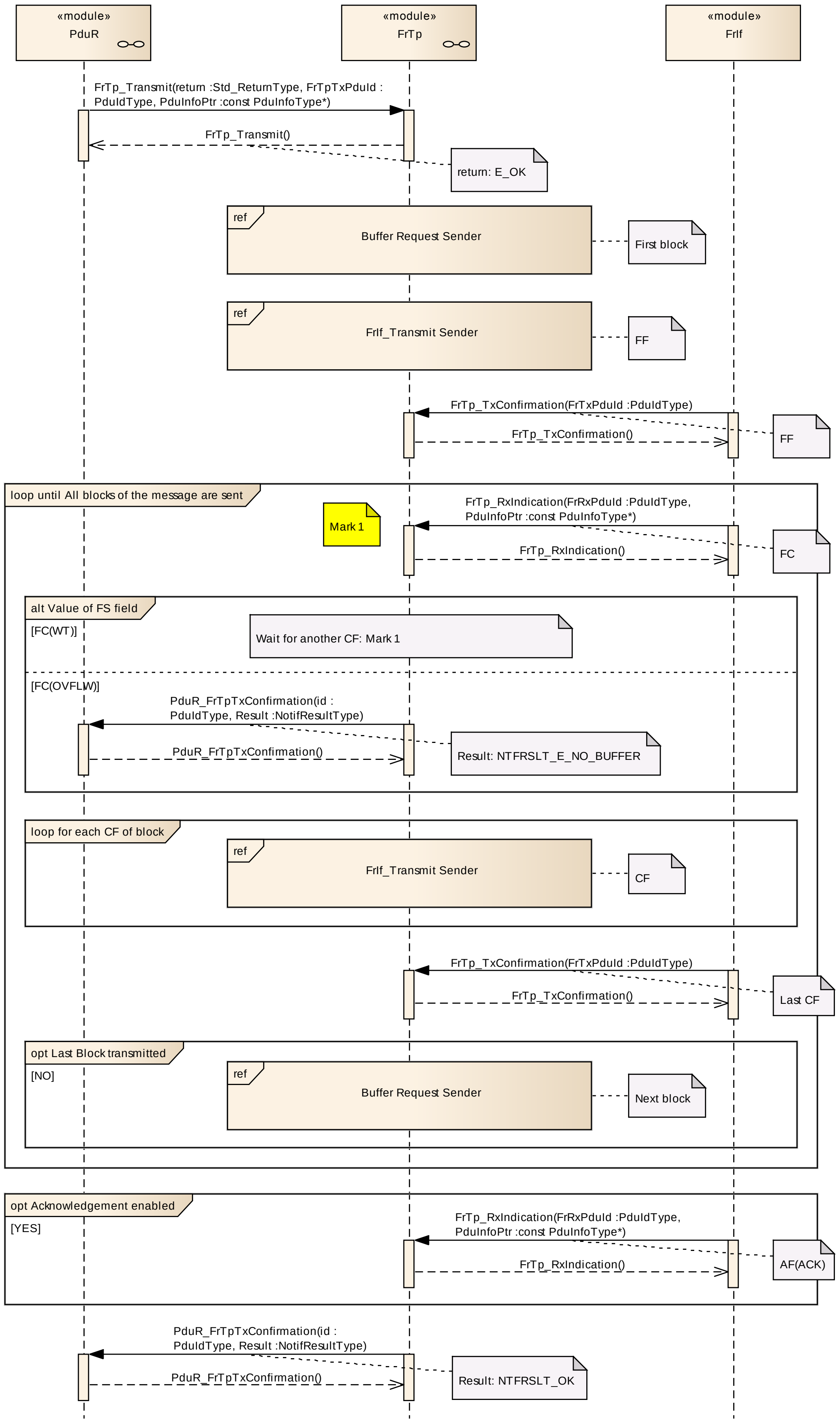
Although the following sequence diagrams are quite detailed, they do not depict every detail. Thus, they should be seen as an addendum to this specification.

9.1 Sending

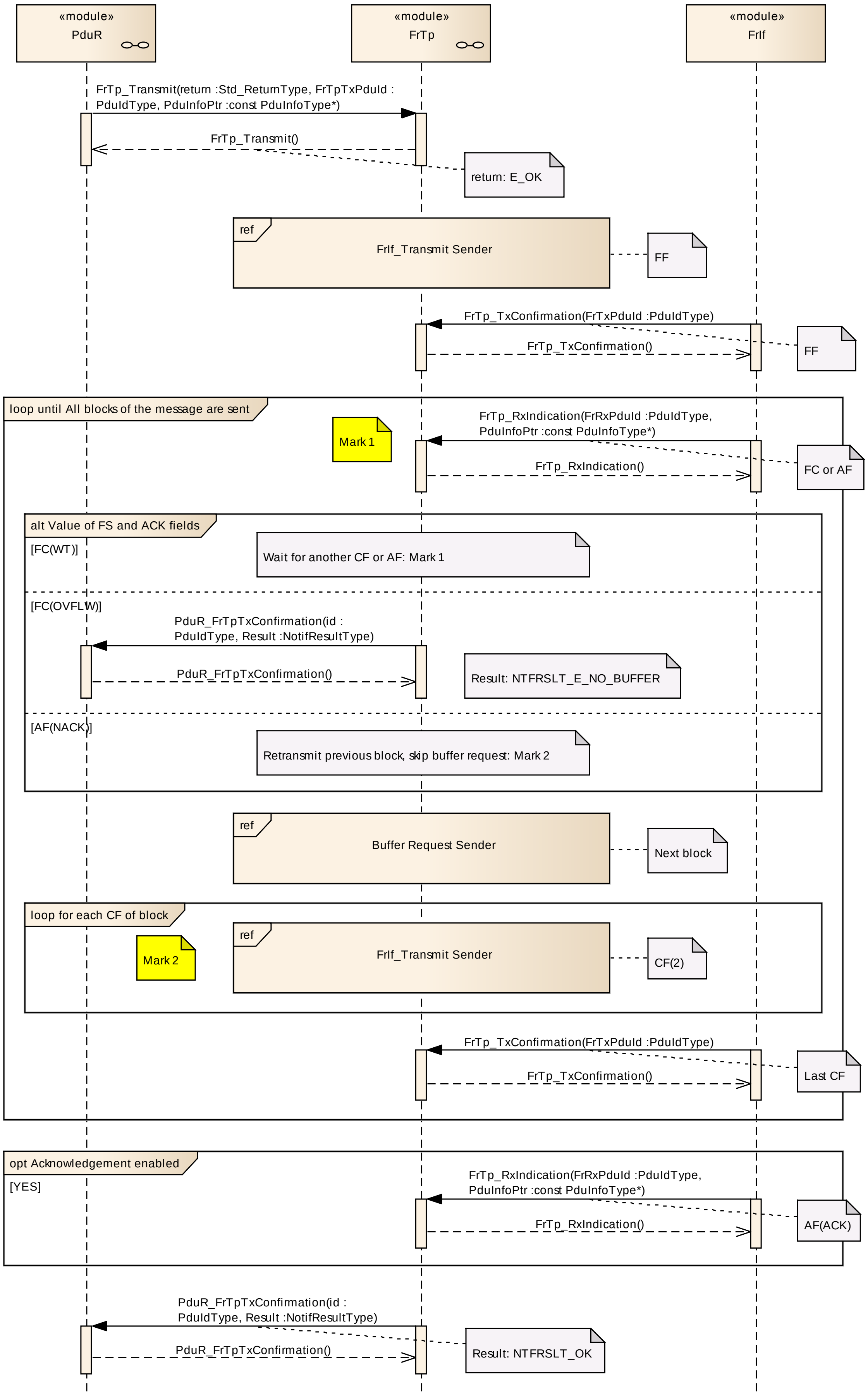
9.1.1 Unsegmented Sending


«module»
PduR
«module»
FrTp
«module»
FrIf
FrTp_Transmit(return :Std_ReturnType, FrTpTxPduId : PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_Transmit()
return: E_OK
ref
Buffer Request Sender
Complete SF
Mark 1
ref
FrIf_Transmit Sender
SF
FrTp_TxConfirmation(FrTxPduId :PduIdType)
FrTp_TxConfirmation()
SF
opt Acknowledgement enabled
[YES]
Mark 2
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr : const PduInfoType*)
FrTp_RxIndication()
AF
opt Retry enabled
[YES]
alt Value of FS and ACK fields
[NACK]
Transmit SF again: Mark 1
[WT]
Wait for another AF: Mark 2
[OVFLW]
PduR_FrTpTxConfirmation(id : PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_E_NO_BUFFER
PduR_FrTpTxConfirmation(id : PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_OK

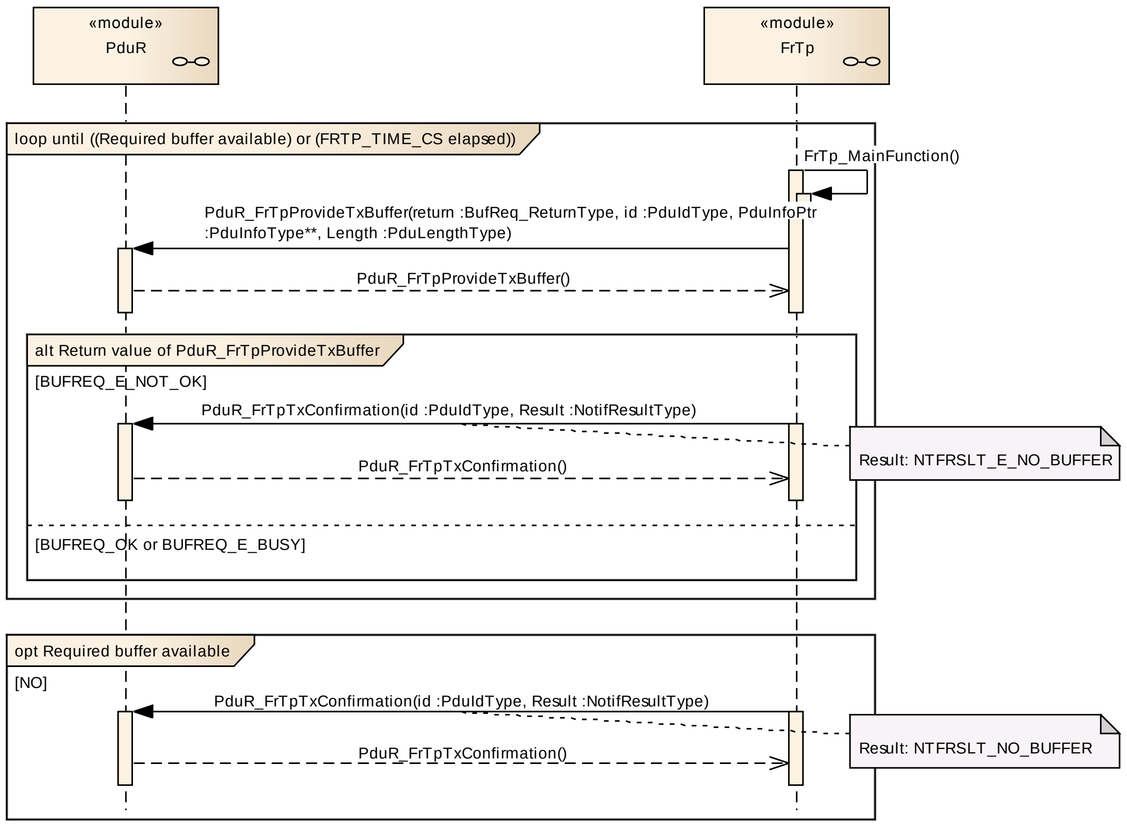
9.1.2 Segmented Sending without Retry


«module» PduR
«module» FrTp
«module» FrIf
FrTp_Transmit(return :Std_ReturnType, FrTpTxPduId : PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_Transmit()
return: E_OK
ref Buffer Request Sender
First block
ref FrIf_Transmit Sender
FF
FrTp_TxConfirmation(FrTxPduId :PduIdType)
FrTp_TxConfirmation()
FF
loop until All blocks of the message are sent
Mark 1
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
FC
alt Value of FS field
[FC(WT)]
Wait for another CF: Mark 1
[FC(OVFLW)]
PduR_FrTpTxConfirmation(id : PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_E_NO_BUFFER
loop for each CF of block
ref FrIf_Transmit Sender
CF
FrTp_TxConfirmation(FrTxPduId :PduIdType)
FrTp_TxConfirmation()
Last CF
opt Last Block transmitted
[NO]
ref Buffer Request Sender
Next block
opt Acknowledgement enabled
[YES]
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
AF(ACK)
PduR_FrTpTxConfirmation(id : PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_OK

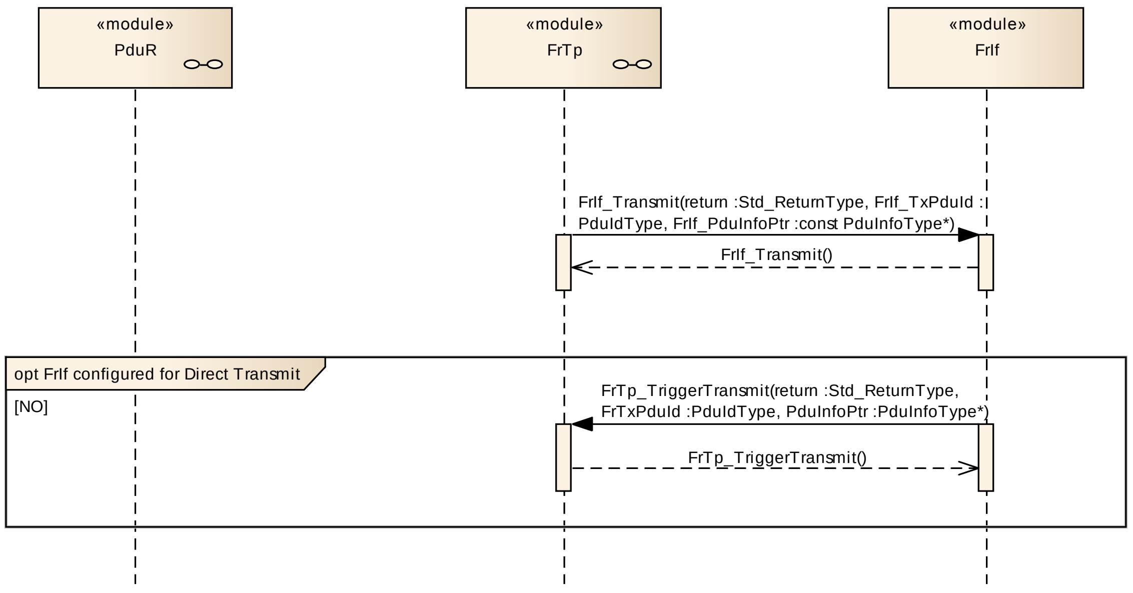
9.1.3 Segmented Sending with Retry


«module»
PduR
«module»
FrTp
«module»
FrIf
FrTp_Transmit(return :Std_ReturnType, FrTpTxPduId : PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_Transmit()
return: E_OK
ref
FrIf_Transmit Sender
FF
FrTp_TxConfirmation(FrTxPduId :PduIdType)
FrTp_TxConfirmation()
FF
loop until All blocks of the message are sent
Mark 1
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
FC or AF
alt Value of FS and ACK fields
[FC(WT)]
Wait for another CF or AF: Mark 1
[FC(OVFLW)]
PduR_FrTpTxConfirmation(id : PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_E_NO_BUFFER
[AF(NACK)]
Retransmit previous block, skip buffer request: Mark 2
ref
Buffer Request Sender
Next block
loop for each CF of block
Mark 2
ref
FrIf_Transmit Sender
CF(2)
FrTp_TxConfirmation(FrTxPduId :PduIdType)
FrTp_TxConfirmation()
Last CF
opt Acknowledgement enabled
[YES]
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
AF(ACK)
PduR_FrTpTxConfirmation(id : PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_OK

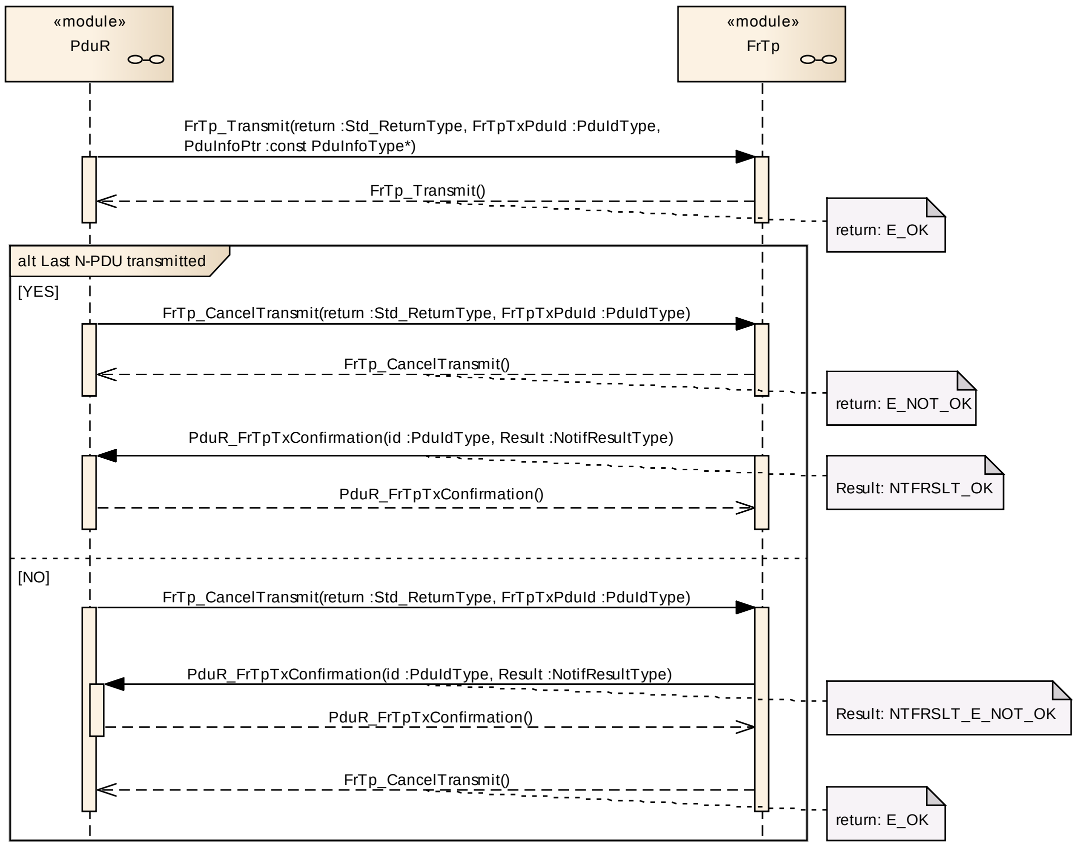
9.1.4 Buffer Request Sender


«module»
PduR
«module»
FrTp
loop until ((Required buffer available) or (FRTP_TIME_CS elapsed))
FrTp_MainFunction()
PduR_FrTpProvideTxBuffer(return :BufReq_ReturnType, id :PduIdType, PduInfoPtr :PduInfoType**, Length :PduLengthType)
PduR_FrTpProvideTxBuffer()
alt Return value of PduR_FrTpProvideTxBuffer
[BUFREQ_E_NOT_OK]
PduR_FrTpTxConfirmation(id :PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_E_NO_BUFFER
[BUFREQ_OK or BUFREQ_E_BUSY]
opt Required buffer available
[NO]
PduR_FrTpTxConfirmation(id :PduIdType, Result :NotifResultType)
PduR_FrTpTxConfirmation()
Result: NTFRSLT_NO_BUFFER

9.1.5 FrIf_Transmit Sender

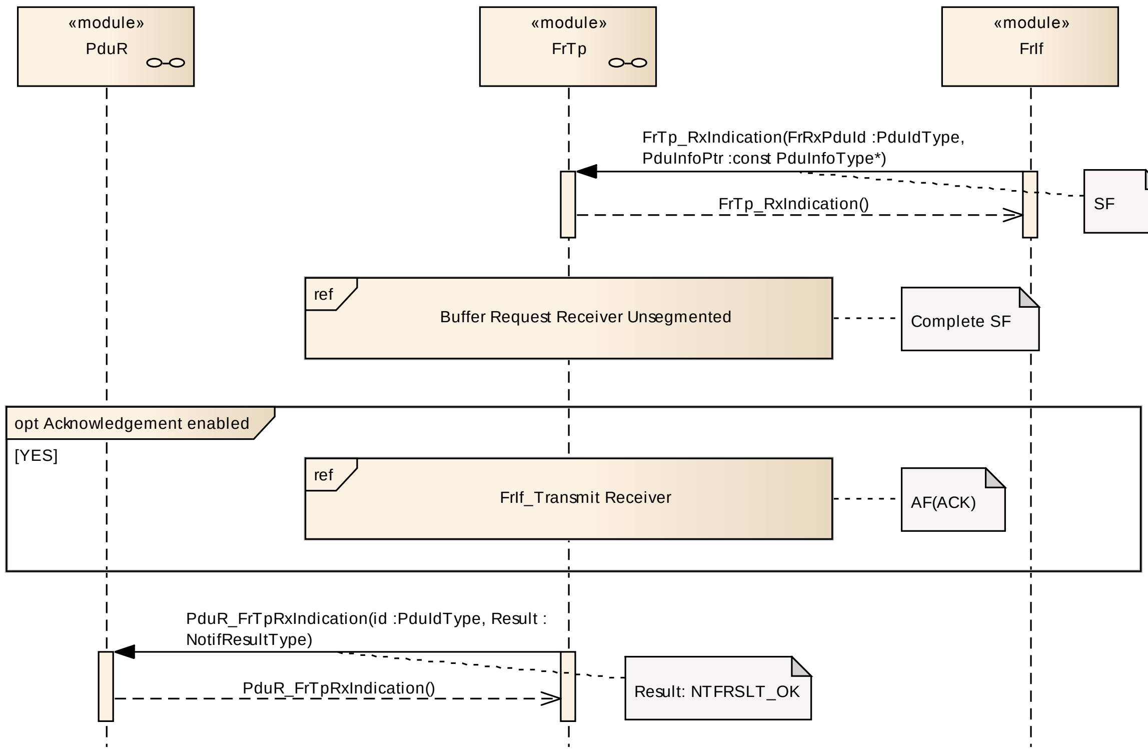

«module»
PduR
«module»
FrTp
«module»
FrIf
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId :PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
opt FrIf configured for Direct Transmit
[NO]
FrTp_TriggerTransmit(return :Std_ReturnType, FrTxPduId :PduIdType, PduInfoPtr :PduInfoType*)
FrTp_TriggerTransmit()

9.1.6 Transmit Cancellation

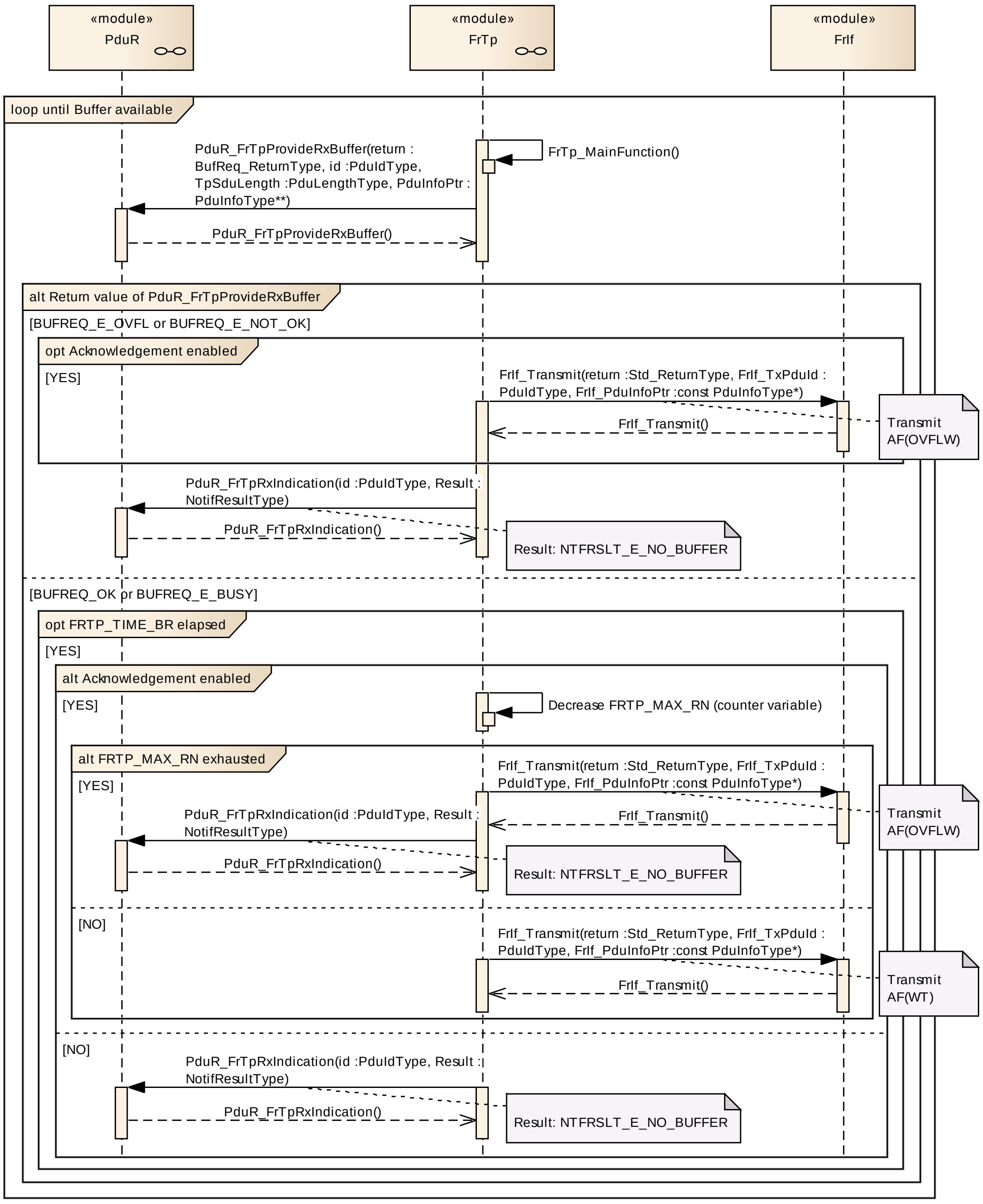

«module»
PduR
«module»
FrTp
FrTp_Transmit(return :Std_ReturnType, FrTpTxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_Transmit()
return: E_OK
alt Last N-PDU transmitted
[YES]
FrTp_CancelTransmit(return :Std_ReturnType, FrTpTxPduId :PduIdType)
FrTp_CancelTransmit()
return: E_NOT_OK
PduR_FrTpTxConfirmation(id :PduIdType, Result :NotifResultType)
Result: NTFRSLT_OK
PduR_FrTpTxConfirmation()
[NO]
FrTp_CancelTransmit(return :Std_ReturnType, FrTpTxPduId :PduIdType)
PduR_FrTpTxConfirmation(id :PduIdType, Result :NotifResultType)
Result: NTFRSLT_E_NOT_OK
PduR_FrTpTxConfirmation()
FrTp_CancelTransmit()
return: E_OK

9.2 Receiving

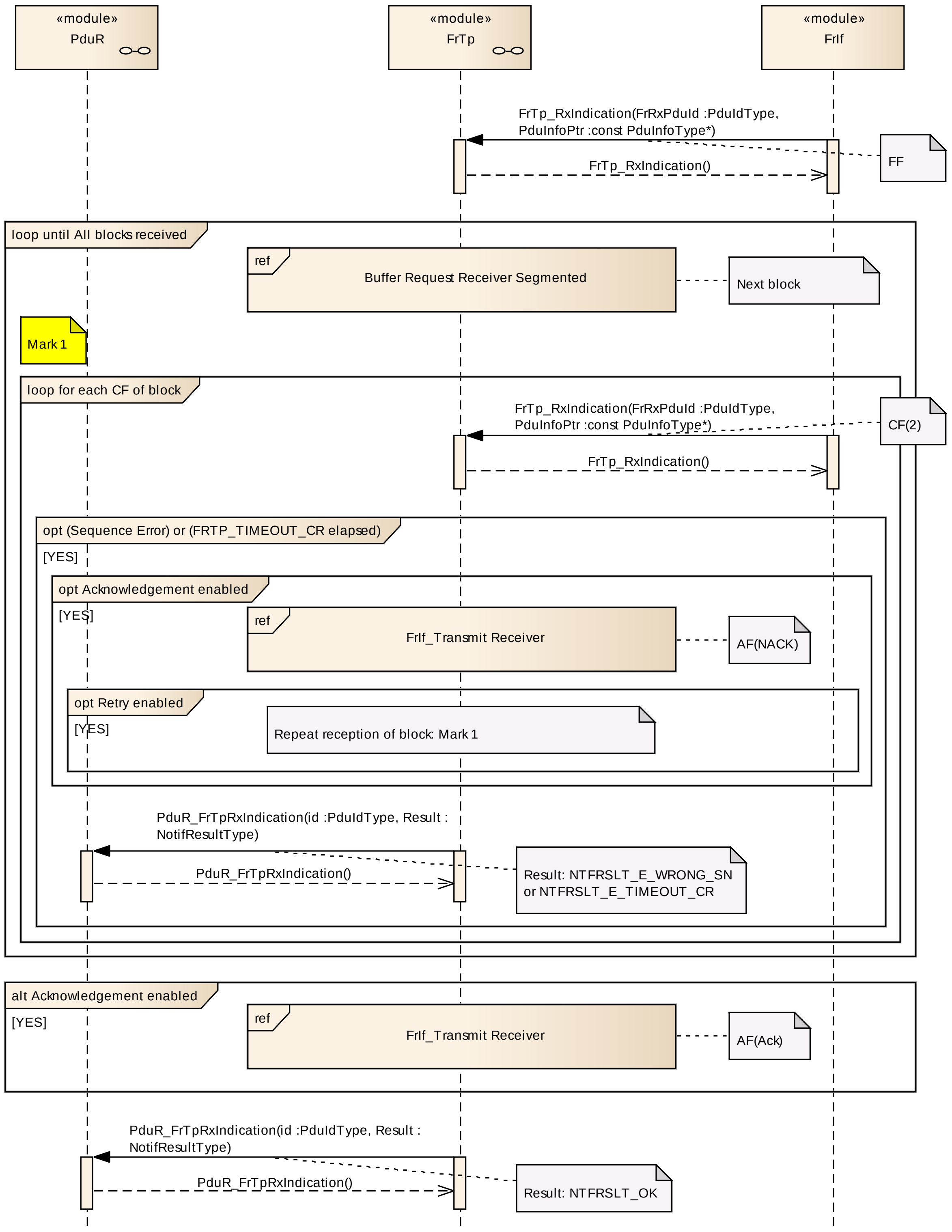
9.2.1 Unsegmented Receiving


«module» PduR
«module» FrTp
«module» FrIf
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
SF
ref
Buffer Request Receiver Unsegmented
Complete SF
opt Acknowledgement enabled
[YES]
ref
FrIf_Transmit Receiver
AF(ACK)
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_OK

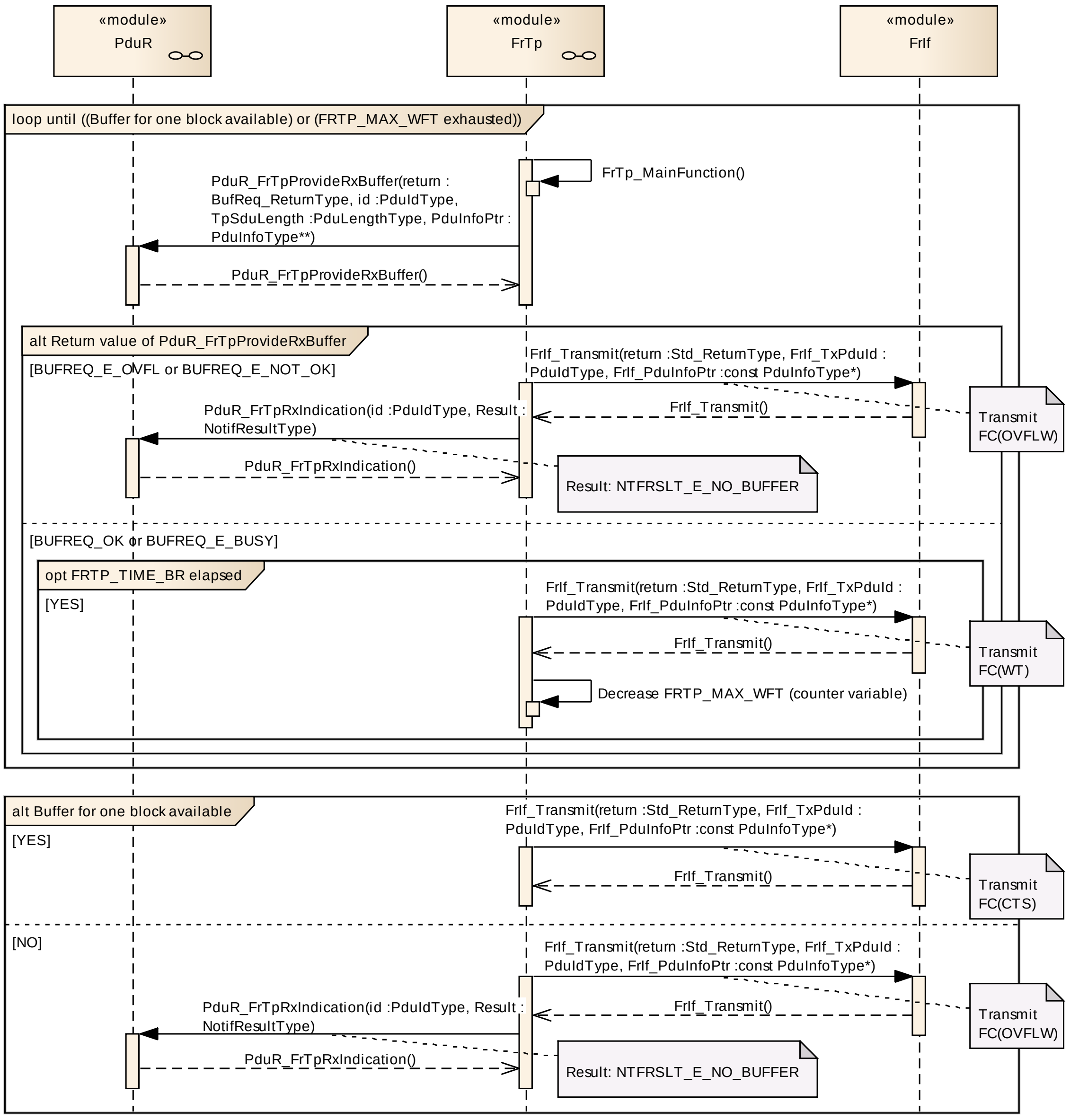
9.2.2 Unsegmented Buffer Request Receiver


«module» PduR
«module» FrTp
«module» FrIf
loop until Buffer available
FrTp_MainFunction()
PduR_FrTpProvideRxBuffer(return : BufReq_ReturnType, id :PduIdType, TpSduLength :PduLengthType, PduInfoPtr : PduInfoType**)
PduR_FrTpProvideRxBuffer()
alt Return value of PduR_FrTpProvideRxBuffer
[BUFREQ_E_OVFL or BUFREQ_E_NOT_OK]
opt Acknowledgement enabled
[YES]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit AF(OVFLW)
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_E_NO_BUFFER
[BUFREQ_OK or BUFREQ_E_BUSY]
opt FRTP_TIME_BR elapsed
[YES]
alt Acknowledgement enabled
[YES]
Decrease FRTP_MAX_RN (counter variable)
alt FRTP_MAX_RN exhausted
[YES]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit AF(OVFLW)
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_E_NO_BUFFER
[NO]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit AF(WT)
[NO]
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_E_NO_BUFFER

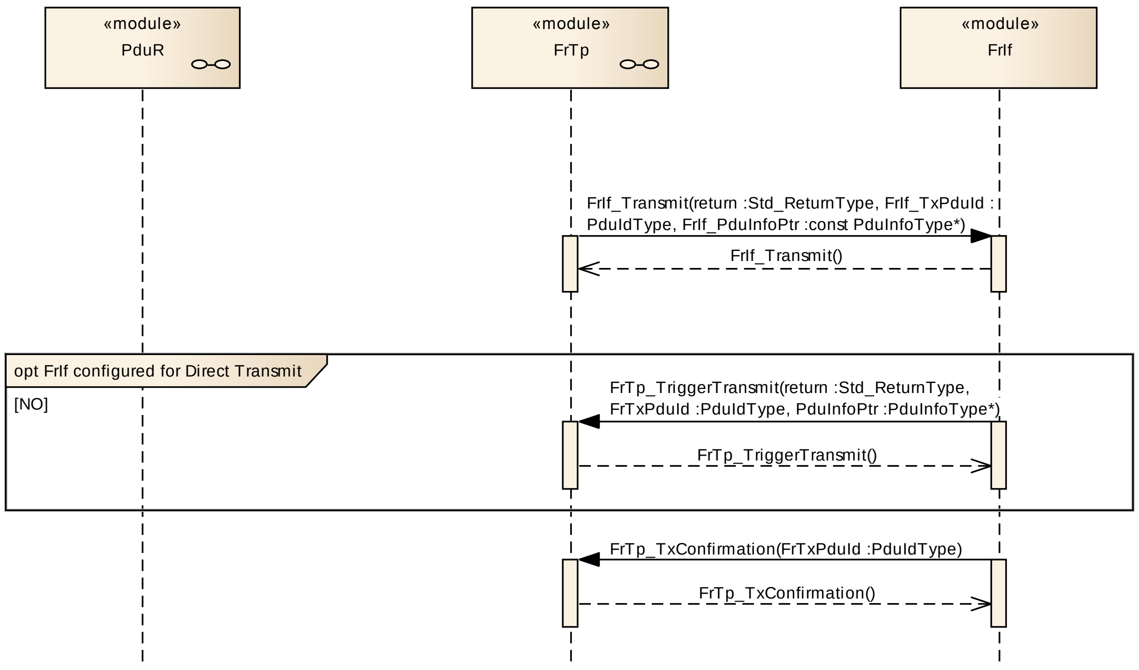
9.2.3 Segmented Receiving


«module»
PduR
«module»
FrTp
«module»
FrIf
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
FF
loop until All blocks received
ref
Buffer Request Receiver Segmented
Next block
Mark 1
loop for each CF of block
FrTp_RxIndication(FrRxPduId :PduIdType, PduInfoPtr :const PduInfoType*)
FrTp_RxIndication()
CF(2)
opt (Sequence Error) or (FRTP_TIMEOUT_CR elapsed)
[YES]
opt Acknowledgement enabled
[YES]
ref
FrIf_Transmit Receiver
AF(NACK)
opt Retry enabled
[YES]
Repeat reception of block: Mark 1
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_E_WRONG_SN or NTFRSLT_E_TIMEOUT_CR
alt Acknowledgement enabled
[YES]
ref
FrIf_Transmit Receiver
AF(Ack)
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_OK

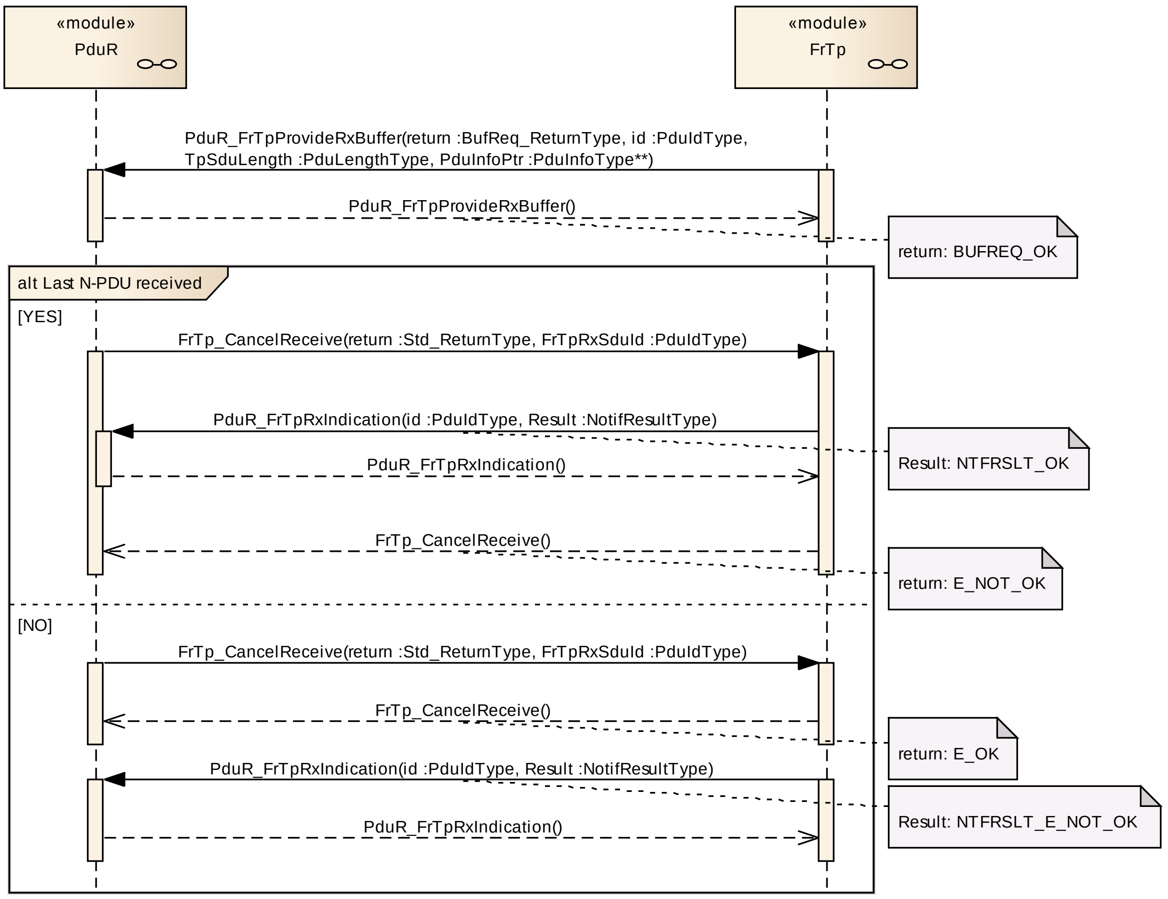
9.2.4 Segmented Buffer Request Receiver


«module»
PduR
«module»
FrTp
«module»
FrIf
loop until ((Buffer for one block available) or (FRTP_MAX_WFT exhausted))
FrTp_MainFunction()
PduR_FrTpProvideRxBuffer(return : BufReq_ReturnType, id :PduIdType, TpSduLength :PduLengthType, PduInfoPtr : PduInfoType**)
PduR_FrTpProvideRxBuffer()
alt Return value of PduR_FrTpProvideRxBuffer
[BUFREQ_E_OVFL or BUFREQ_E_NOT_OK]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit FC(OVFLW)
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_E_NO_BUFFER
[BUFREQ_OK or BUFREQ_E_BUSY]
opt FRTP_TIME_BR elapsed
[YES]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit FC(WT)
Decrease FRTP_MAX_WFT (counter variable)
alt Buffer for one block available
[YES]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit FC(CTS)
[NO]
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
Transmit FC(OVFLW)
PduR_FrTpRxIndication(id :PduIdType, Result : NotifResultType)
PduR_FrTpRxIndication()
Result: NTFRSLT_E_NO_BUFFER

9.2.5 FrIf_Transmit Receiver


«module» PduR
«module» FrTp
«module» FrIf
FrIf_Transmit(return :Std_ReturnType, FrIf_TxPduId : PduIdType, FrIf_PduInfoPtr :const PduInfoType*)
FrIf_Transmit()
opt FrIf configured for Direct Transmit
[NO]
FrTp_TriggerTransmit(return :Std_ReturnType, FrTxPduId :PduIdType, PduInfoPtr :PduInfoType*)
FrTp_TriggerTransmit()
FrTp_TxConfirmation(FrTxPduId :PduIdType)
FrTp_TxConfirmation()

9.2.6 Receive Cancellation


«module» PduR
«module» FrTp
PduR_FrTpProvideRxBuffer(return :BufReq_ReturnType, id :PduIdType, TpSduLength :PduLengthType, PduInfoPtr :PduInfoType**)
PduR_FrTpProvideRxBuffer()
return: BUFREQ_OK
alt Last N-PDU received
[YES]
FrTp_CancelReceive(return :Std_ReturnType, FrTpRxSduId :PduIdType)
PduR_FrTpRxIndication(id :PduIdType, Result :NotifResultType)
Result: NTFRSLT_OK
PduR_FrTpRxIndication()
FrTp_CancelReceive()
return: E_NOT_OK
[NO]
FrTp_CancelReceive(return :Std_ReturnType, FrTpRxSduId :PduIdType)
FrTp_CancelReceive()
return: E_OK
PduR_FrTpRxIndication(id :PduIdType, Result :NotifResultType)
Result: NTFRSLT_E_NOT_OK
PduR_FrTpRxIndication()

10 Configuration specification

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

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

Section 10.3 specifies published information of the module FlexRay AUTOSAR Transport Layer.

10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture.
- AUTOSAR ECU Configuration Specification. 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 an implementation of a module. 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 during the software process: before compile time, before link time or 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

Yes

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 following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time	-	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 sections summarize all configuration parameters. The detailed meaning of these parameters is described in chapters 7 and 8.

The following pictures give an overview of the FrTp configuration:

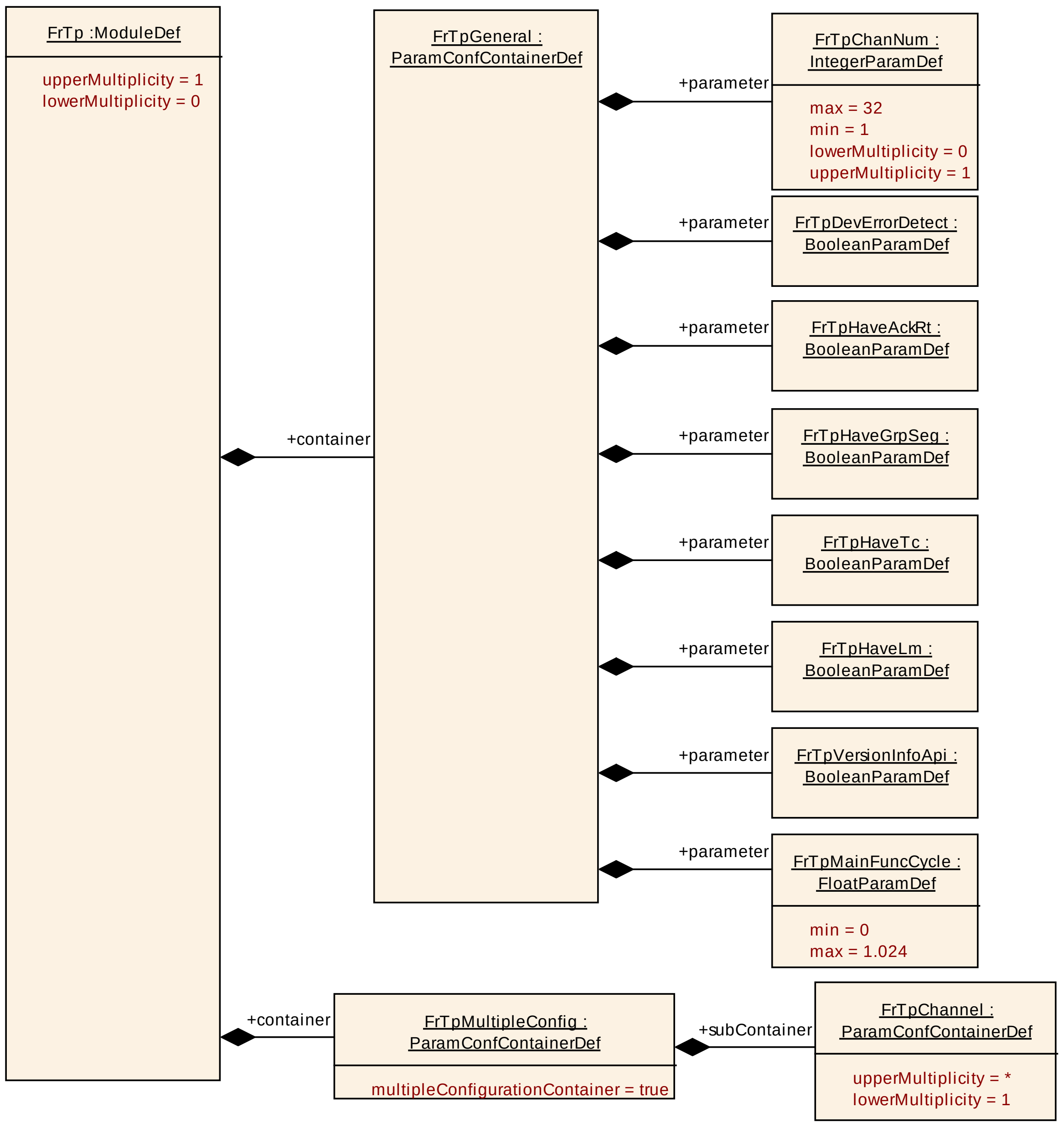

FrTp :ModuleDef
upperMultiplicity = 1
lowerMultiplicity = 0
+container
FrTpGeneral : ParamConfContainerDef
+parameter
FrTpChanNum : IntegerParamDef
max = 32
min = 1
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrTpDevErrorDetect : BooleanParamDef
+parameter
FrTpHaveAckRt : BooleanParamDef
+parameter
FrTpHaveGrpSeg : BooleanParamDef
+parameter
FrTpHaveTc : BooleanParamDef
+parameter
FrTpHaveLm : BooleanParamDef
+parameter
FrTpVersionInfoApi : BooleanParamDef
+parameter
FrTpMainFuncCycle : FloatParamDef
min = 0
max = 1.024
+container
FrTpMultipleConfig : ParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
FrTpChannel : ParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1

Figure 22: FrTp main configuration

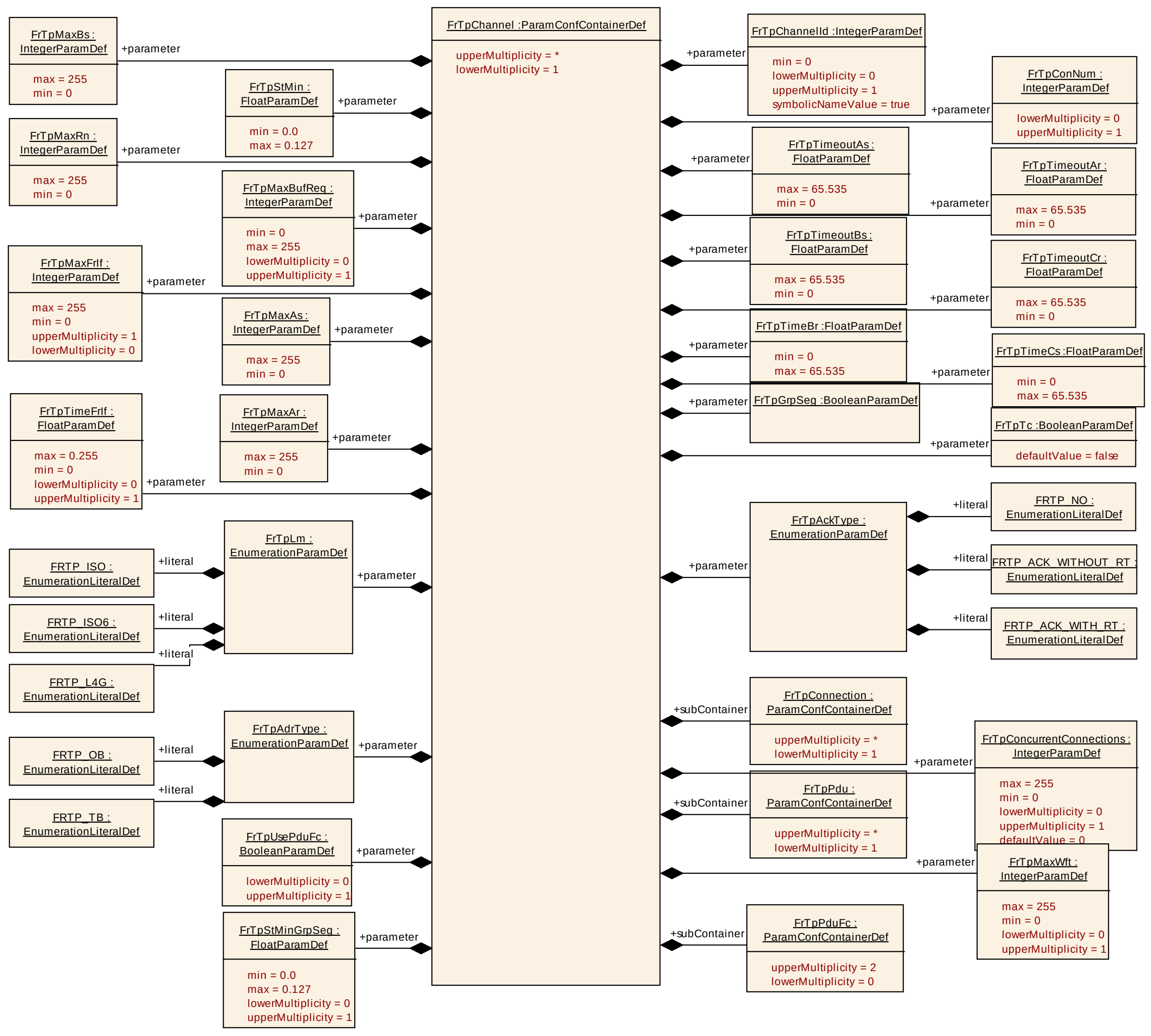

FrTpChannel :ParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
FrTpMaxBs : IntegerParamDef
max = 255
min = 0
FrTpMaxRn : IntegerParamDef
max = 255
min = 0
FrTpStMin : FloatParamDef
min = 0.0
max = 0.127
FrTpMaxBufReq : IntegerParamDef
min = 0
max = 255
lowerMultiplicity = 0
upperMultiplicity = 1
FrTpMaxFrIf : IntegerParamDef
max = 255
min = 0
upperMultiplicity = 1
lowerMultiplicity = 0
FrTpMaxAs : IntegerParamDef
max = 255
min = 0
FrTpTimeFrIf : FloatParamDef
max = 0.255
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
FrTpMaxAr : IntegerParamDef
max = 255
min = 0
FrTpLm : EnumerationParamDef
FRTP_ISO : EnumerationLiteralDef
FRTP_ISO6 : EnumerationLiteralDef
FRTP_L4G : EnumerationLiteralDef
FrTpAdrType : EnumerationParamDef
FRTP_OB : EnumerationLiteralDef
FRTP_TB : EnumerationLiteralDef
FrTpUsePduFc : BooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
FrTpStMinGrpSeq : FloatParamDef
min = 0.0
max = 0.127
lowerMultiplicity = 0
upperMultiplicity = 1
FrTpChannelId :IntegerParamDef
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
symbolicNameValue = true
FrTpConNum : IntegerParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
FrTpTimeoutAs : FloatParamDef
max = 65.535
min = 0
FrTpTimeoutAr : FloatParamDef
max = 65.535
min = 0
FrTpTimeoutBs : FloatParamDef
max = 65.535
min = 0
FrTpTimeoutCr : FloatParamDef
max = 65.535
min = 0
FrTpTimeBr :FloatParamDef
min = 0
max = 65.535
FrTpTimeCs :FloatParamDef
min = 0
max = 65.535
FrTpGrpSeg :BooleanParamDef
FrTpTc :BooleanParamDef
defaultValue = false
FrTpAckType : EnumerationParamDef
FRTP_NO : EnumerationLiteralDef
FRTP_ACK_WITHOUT_RT : EnumerationLiteralDef
FRTP_ACK_WITH_RT : EnumerationLiteralDef
FrTpConnection : ParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
FrTpConcurrentConnections : IntegerParamDef
max = 255
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
defaultValue = 0
FrTpPdu : ParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
FrTpMaxWft : IntegerParamDef
max = 255
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
FrTpPduFc : ParamConfContainerDef
upperMultiplicity = 2
lowerMultiplicity = 0
+parameter
+literal
+subContainer

Figure 23: FrTpChannel configuration

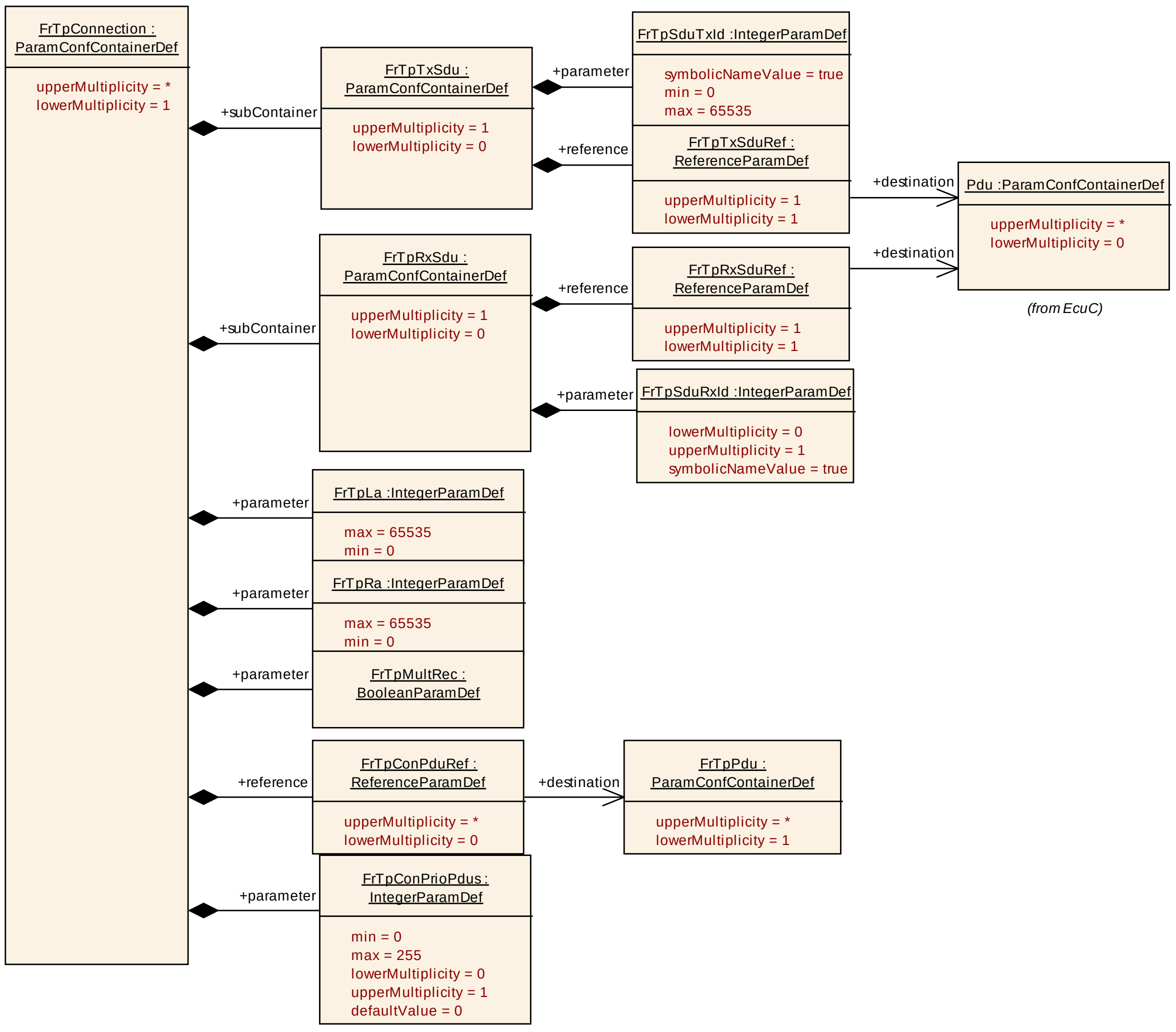


Figure 24: FrTpConnection configuration

10.2.1 Variants

Variant 1: Pre Compile time

Variant 2: Mixture of Pre Compile time and Post Build Time Parameters

10.2.2 FrTp

Module Name	*FrTp*
Module Description	Configuration of the FrTp (FlexRay Transport Protocol) module.

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

10.2.3 FrTpGeneral

SWS Item	:
Container Name	FrTpGeneral
Description	This container contains the general configuration (parameters) of the FlexRay TP.
Configuration Parameters	

SWS Item	:		
Name	FrTpChanNum {FRTP_CHAN_NUM}		
Description	This reference is deprecated and will be removed in a future release. Old description: Preprocessor switch for defining the number of concurrent channels the module supports. Up to 32 channels shall be definable here.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	1 .. 32		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpDevErrorDetect {FRTP_DEV_ERROR_DETECT}		
Description	Preprocessor switch for enabling development error detection.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpHaveAckRt {FRTP_HAVE_ACKRT}		
Description	Preprocessor switch for enabling the Acknowledgement and retry mechanisms.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpHaveGrpSeg {FRTP_HAVE_GRPSEG}		
Description	Preprocessor switch for enabling segmentation of 1:n messages.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpHaveLm {FRTP_HAVE_LM}		
Description	Preprocessor switch for enabling the mechanism for message longer than allowed by.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpHaveTc {FRTP_HAVE_TC}		
Description	Preprocessor switch for enabling Transmit Cancellation and Receive Cancellation.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	:		
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	FloatParamDef		
Range	0 .. 1.024		
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	:		
Name	FrTpVersionInfoApi {FRTP_VERSION_INFO_API}		
Description	Preprocessor switch for enabling the Version info API.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: Module

No Included Containers

All parameters within section 10.2.2 are global and, of course, only present once for the whole software module.

10.2.4 FrTpChannel

SWS Item	:
Container Name	FrTpChannel{FrTpChannelConfiguration}
Description	This container contains the configuration (parameters) of one FlexRay TP channel.
Configuration Parameters	

SWS Item	:		
Name	FrTpAckType {FRTP_ACKTYPE}		
Description	This parameter defines the type of acknowledgement which is used for the specific channel.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRTP_ACK_WITHOUT_RT	Acknowledgement without retry	
	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	:		
Name	FrTpAdrType {FRTP_ADRTYPE}		
Description	This parameter states the addressing type this connection has. The meanings of the values are one byte and two byte.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRTP_OB	One Byte	
	FRTP_TB	Two Bytes	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpChannelId {FRTP_CHANNEL_ID}		
Description	Please note that this parameter is deprecated and will be removed in a future release. Old description: The Id of the channel.		
Multiplicity	0..1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
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	:		
Name	FrTpConNum {FRTP_CON_NUM}		
Description	Please note that this parameter is deprecated and will be removed in a future release. Old description: This parameter states the number of connections used in this channel. At least 256 shall be configurable here.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	..		
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	:		
Name	FrTpConcurrentConnections {FRTP_CONCURRENT_CONNECTIONS}		
Description	This parameter defines the number of connections that can be active at the same time. If set to 0, all configured connections can be active at the same time.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	0		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:		
Name	FrTpGrpSeg {FRTP_GRPSEG}		
Description	Here can be specified, whether segmentation within a 1:n connection is allowed or not.		
Multiplicity	1		
Type	BooleanParamDef		
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	:	
Name	FrTpLm {FRTP_LM}	
Description	This specifies the maximum message length for the particular channel.	
Multiplicity	1	
Type	EnumerationParamDef	
Range	FRTP_ISO	Up to (2**12)-1 Byte message length (No FF-Ex or SF-E or AF shall be used and recognized)
	FRTP_ISO6	As ISO, but the maximum payload length is limited to 6 byte (SF-I, FF-I, CF). This is necessary to route TP on CAN when using Extended Addressing or Mixed Addressing on CAN.
	FRTP_L4G	SF-E allowed (SF of arbitrary length depending on FrTpPduLength), up to (2**32)-1 byte message

		length (all FF-x allowed).	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:		
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	IntegerParamDef		
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	:		
Name	FrTpMaxAs {FRTP_MAX_AS}		
Description	This parameter defines the maximum number of trying t osend a frame when a TIMEOUT AS occurs.		
Multiplicity	1		
Type	IntegerParamDef		
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	:		
Name	FrTpMaxBs {FRTP_MAXBS}		
Description	This parameter defines the number of consecutive CFs between two FCs (block size). Valid values are 1 .. 16 when retry is activated, and 0 .. 255 otherwise.		
Multiplicity	1		
Type	IntegerParamDef		
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	:
Name	FrTpMaxBufReq {FRTP_MAX_BUFREQ}
Description	Please note that this parameter is deprecated and will be removed in a future release. FrTpMaxWft will be used instead to configure the maximum number of wait frames on receiver side. On the sender side, timeCs defines the maximum time for retries. Old description: This parameter defines the maximum number of trying to get a buffer (Transmit / Receive), depending of the return value of PduR_FrTpProvideTxBuffer / PduR_FrTpProvideRxBuffer and on whether retry is configured.
Multiplicity	0..1
Type	IntegerParamDef

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	:		
Name	FrTpMaxFrIf {FRTP_MAX_FRIF}		
Description	This parameter is deprecated and will be removed in future. Old description: This parameter defines the maximum number of trying to send a frame when the FrIf returns an error.		
Multiplicity	0..1		
Type	IntegerParamDef		
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	:		
Name	FrTpMaxRn {FRTP_MAX_RN}		
Description	This parameter defines the maximum number of retries (if retry is configured for the particular channel).		
Multiplicity	1		
Type	IntegerParamDef		
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	:		
Name	FrTpMaxWft {FRTP_MAX_WFT}		
Description	This parameter defines the maximal number of wait frames to be sent for a pending connection.		
Multiplicity	0..1		
Type	IntegerParamDef		
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	:
Name	FrTpStMin {FRTP_STMIN}
Description	This parameter defines the minimum amount of time between two succeeding CFs of a 1:1 segmented transmission in seconds. Valid values are 0, 100µs, 200µs .. 900µs, 1ms, 2ms .. 127ms. The value can be changed at runtime using the FrTp_ChangeParameter interface. FrTpStMin must be an integer multiple of the cycle length multiplied with the multiplexing factor, i.e. FrTpStMin = n * FrIfGdCycle * m, where n is an integer ≥ 0 and m is the cycle multiplexor of those cycles where PDUs of the PDU pool are scheduled. Please note: Due to the

	scheduling strategies of FrTp, FrTpStMin can only be kept to a degree defined by the maximum temporal distance of the PDUs of a PDU pool within one FlexRay cycle.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. 0.127		
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	:		
Name	FrTpStMinGrpSeg {FRTP_STMIN_GRP_SEG}		
Description	This parameter defines the minimum amount of time between two succeeding CFs of a 1:n segmented transmission in seconds. Valid values are 0, 100µs, 200µs ... 900µs, 1ms, 2ms .. 127ms. The value can be changed at runtime using the FrTp_ChangeParameter interface. FrTpStMinGrpSeg must be an integer multiple of the cycle length multiplied with the multiplexing factor, i.e. FrTpStMinGrpSeg = n * FrIfGdCycle * m, where n is an integer ≥ 0 and m is the cycle multiplexor of those cycles where PDUs of the PDU pool are scheduled. Please note: Due to the scheduling strategies of FrTp, FrTpStMinGrpSeg can only be kept to a degree defined by the maximum temporal distance of the PDUs of a PDU pool within one FlexRay cycle.		
Multiplicity	0..1		
Type	FloatParamDef		
Range	0 .. 0.127		
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	:		
Name	FrTpTc {FRTP_TC}		
Description	With this switch Transmit Cancellation and Receive Cancellation can be turned on or off for this channel.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:
Name	FrTpTimeBr {FRTP_TIME_BR}
Description	This parameter defines the time in seconds between receiving the last CF of a block or an FF-x (or SF-x) and sending out an FC or AF. It is obvious that FRTP_TIME_BR + (FRTP_TIMEOUT_AR * FRTP_MAX_AR) < FRTP_TIMEOUT_BS must hold (because the transmission duration on the bus has also to be considered). This parameter is defined in ISO 15765-2. It is contained in the configuration as a performance requirement.
Multiplicity	1
Type	FloatParamDef

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			

SWS Item	:		
Name	FrTpTimeCs {FRTP_TIME_CS}		
Description	This parameter defines the time in seconds between the sending of two consecutive CFs or between reception of an FC or AF and sending of the next CF . It is obvious that FRTP_TIME_CS + (FRTP_TIMEOUT_AS * FRTP_MAX_AS) < FRTP_TIMEOUT_CR must hold (because the transmission duration on the bus has also to be considered). This parameter is defined in ISO 15765-2. It is contained in the configuration as a performance requirement.		
Multiplicity	1		
Type	FloatParamDef		
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			

SWS Item	:		
Name	FrTpTimeFrIf {FRTP_TIME_FRIF}		
Description	This parameter is deprecated and will be removed in future. Old description: This parameter defines the time in seconds of waiting for the next try to send via FrIf_Transmit.		
Multiplicity	0..1		
Type	FloatParamDef		
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	:		
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	FloatParamDef		
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	:
Name	FrTpTimeoutAs {FRTP_TIMEOUT_AS}
Description	This parameter states the timeout in seconds between the PDU transmit request for the first PDU of the group used in the current connection of the Transport

	Layer to the FlexRay Interface and the corresponding confirmation of the FlexRay Interface (when having sent the last PDU of the group used in this connection) on the sender side (SF-x, FF-x, CF).		
Multiplicity	1		
Type	FloatParamDef		
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	:		
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	FloatParamDef		
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	:		
Name	FrTpTimeoutCr {FRTP_TIMEOUT_CR}		
Description	This parameter defines the timeout value in seconds for waiting for a CF or FF-x (in case of retry) after receiving the last CF or after sending an FC or AF on the receiver side.		
Multiplicity	1		
Type	FloatParamDef		
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	:		
Name	FrTpUsePduFc {FRTP_USE_PDU_FC}		
Description	Please note that this parameter is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools. Old description: This switch defines, whether within this channel the dedicated FC/ACK PDU (FrTpPduFc) shall be used or not. If this is not used FC / ACK frames are sent using the normal IDs, otherwise only FrTpPduFc shall be used for sending / receiving FC / ACK frames.		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	--		
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***
FrTpConnection	1..*	This container contains the configuration (parameters) of one FlexRay TP connection. A connection can only belong to one channel.
FrTpPdu	1..*	Container to hold the PDU parameters. ImplementationType: FrTp_PduInfoType
FrTpPduFc	0..2	Please note that this container with all references and parameters is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools. Old description: This is the identifier of the FlexRay Interface PDUs (Fr N-PDU, Fr L-SDU) in which the Transport Layer Flow Control and Acknowledgement Frames of this channel should be transmitted.

All parameters within this section are present for each channel and read-only. They shall be placed outside the source code of the module in order to be modifiable by a flashing process without re-flashing the code itself.

Performance Requirements according to [12]
The two parameters, FrTpTimeBr and FrTpTimeCs, are **not software configuration parameters**, they are contained in [12] as performance requirements. They are just for information.

10.2.5 FrTpPdu

SWS Item	:
Container Name	FrTpPdu
Description	Container to hold the PDU parameters. ImplementationType: FrTp_PduInfoType
Configuration Parameters	

SWS Item	:		
Name	FrTpPduDirection {FRTP_PDU_DIRECTION}		
Description	This parameter defines the direction of the PDU.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRTP_RX	Received PDU	
	FRTP_TX	Transmitted PDU	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	:
Name	FrTpPduId {FRTP_PDU}
Description	This is the identifier of the FlexRay Interface PDUs (Fr N-PDU, Fr L-SDU) in which the Transport Layer Frames of this channel should be transmitted. The FrTpPduId is only required for Rx N-PDUs (FrTpPduDirection == FRTP_RX), and should be omitted for Tx N-PDUs (FrTpPduDirection == FRTP_TX). ImplementationType: PduIdType
Multiplicity	1

Type	IntegerParamDef (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	:		
Name	FrTpPduRef		
Description	--		
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

10.2.6 FrTpPduFc

SWS Item	:
Container Name	FrTpPduFc
Description	Please note that this container with all references and parameters is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools. Old description: This is the identifier of the FlexRay Interface PDUs (Fr N-PDU, Fr L-SDU) in which the Transport Layer Flow Control and Acknowledgement Frames of this channel should be transmitted.
Configuration Parameters	

SWS Item	:		
Name	FrTpPduFcDirection		
Description	This parameter is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools. Old description: This parameter defines the direction of the PDU.		
Multiplicity	0..1		
Type	EnumerationParamDef		
Range	FRTP_FC_RX	Received flow control PDU	
	FRTP_FC_TX	Transmitted flow control PDU	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency			

SWS Item	:
Name	FrTpPduFcId {FRTP_PDU_FC}
Description	This parameter is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools. Old description: This is the identifier of the FlexRay Interface PDUs (Fr N-PDU, Fr L-SDU) in which the Transport Layer Flow Control and Acknowledgement Frames of this channel should be transmitted.
Multiplicity	0..1

Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
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	:		
Name	FrTpPduFcRef		
Description	This reference is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools.		
Multiplicity	0..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 FrTpConnection

SWS Item	:
Container Name	FrTpConnection{FrTpConnectionConfiguration}
Description	This container contains the configuration (parameters) of one FlexRay TP connection. A connection can only belong to one channel.
Configuration Parameters	

SWS Item	:		
Name	FrTpConPrioPdus {FRTP_CON_PRIO_PDUS}		
Description	This parameter defines the number of TxNPdus to which this connection has prioritized access. It must be ensured that the number of prioritized PDUs of all connections is smaller than the total number of TxNPdus in the associated PDU pool.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 255		
Default value	0		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	:
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. Note that in case of 1 byte addressing only the values from 0x0000 - 0x00FF are valid.
Multiplicity	1

Type	IntegerParamDef		
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	:		
Name	FrTpMultRec {FRTP_MULT_REC}		
Description	This parameter defines, whether this connection is an 1:1 ('false') or an 1:n ('true') connection. Of course, if the channel to which the connection is configured has retry or acknowledgement enabled, no retry or acknowledgement will occur in case the connection is an 1:n connection.		
Multiplicity	1		
Type	BooleanParamDef		
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	:		
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. Note that in case of 1 byte addressing only the values from 0x0000 - 0x00FF are valid.		
Multiplicity	1		
Type	IntegerParamDef		
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	:		
Name	FrTpConPduRef {FRTP_CON_PDU}		
Description	This reference is deprecated and will be removed in a future release. It has been rendered irrelevant by the introduction of PDU pools. Old description: Each value defines a PDU to be used for this connection. Thus each value is a PDU-ID given in FrTpPdu and this array cannot be longer than the array FrTpPdu. Please note: Only PDUs of the same size shall be used within a connection. Of course the PDU having the TxConfirmation configured has to be used by every connection.		
Multiplicity	0..*		
Type	Reference to [FrTpPdu]		
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	Describes the Rx SDU
FrTpTxSdu	0..1	Describes the Tx SDU

All parameters within this section are present for each connection and read-only. They shall be placed outside the source code of the module in order to be modifiable by a flashing process without re-flashing the code itself.

10.2.8 FrTpTxSdu

SWS Item	:
Container Name	FrTpTxSdu
Description	Describes the Tx SDU
Configuration Parameters	

SWS Item	:		
Name	FrTpSduTxId {FRTP_SDU_TX_ID}		
Description	This is a unique identifier for a received or a to be transmitted message. With this (and by means of e.g. a lookup table) the PDU Router can route the message appropriately without dealing with the particularities of the Transport Layer. This parameter can also be seen as the identifier of a connection. ImplementationType: PduIdType		
Multiplicity	1		
Type	IntegerParamDef (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	:		
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

10.2.9 FrTpRxSdu

SWS Item	:
Container Name	FrTpRxSdu
Description	Describes the Rx SDU
Configuration Parameters	

SWS Item	:		
Name	FrTpSduRxId {FRTP_SDU_RX_ID}		
Description	Please note that this parameter is deprecated and will be removed in a future release. Old description: This is a unique identifier for a received message. This Id is used in the CancelReceive API call. ImplementationType: PduIdType		
Multiplicity	0..1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
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	:		
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

10.2.10 FrTpMultipleConfig

SWS Item	:
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***
FrTpChannel	1..*	This container contains the configuration (parameters) of one FlexRay TP channel.

10.3 Published Information

FRTP178: published parameters

The following standard common published information is provided in the BSW Module Description Template (see [10] Figure 4.1 and Figure 7.1):

vendorId (FRTP_VENDOR_ID),
moduleId (FRTP_MODULE_ID),
arMajorVersion (FRTP_AR_MAJOR_VERSION),
arMinorVersion (FRTP_ AR_MINOR_VERSION),
arPatchVersion (FRTP_ AR_PATCH_VERSION),
swMajorVersion (FRTP_SW_MAJOR_VERSION),

swMinorVersion (FRTP_ SW_MINOR_VERSION),
swPatchVersion (FRTP_ SW_PATCH_VERSION),
vendorApiInfix (FRTP_VENDOR_API_INFIX)

10.4 Important Issues on Configuration

10.4.1 Start and Stop of the Timing Parameters

FRTP169: Table 5 gives an overview when the time of each of these parameters start to run and when it is stopped. Note that if SF-x is mentioned it is meant in the case acknowledgement is configured (the same for AF).

FRTP170: For 1:n connections only the parameters FrTpTimeoutAs, FrTpTimeCs (only CF) and FrTpTimeoutCr (only CF) hold, since no flow control or acknowledgement is allowed in that case.

Timing Parameter	***Start***	***Stop***
FrTpTimeoutAs	*FrIf_Transmit* (first PDU of the group used by the current connection)	*FrTp_TxConfirmation* (for the last PDU of the group used by the current connection)
FrTpTimeoutAr	*FrIf_Transmit* (FC or AF)	*FrTp_TxConfirmation* (FC or AF)
FrTpTimeoutBs	*FrTp_TxConfirmation* (SF-x, FF-x or last CF of a block), *FrTp_RxIndication* (FC or AF, both in case of FR_FS = WAIT)	*FrTp_RxIndication* (FC or AF)
FrTpTimeBr	*FrTp_RxIndication* (FF-x, last CF of a block or SF-x), *FrTp_TxConfirmation (FC or AF, both in case of FR_FS = WAIT)*	*FrIf_Transmit* (FC or AF)
FrTpTimeoutCr	*FrTp_RxIndication* (CF), *FrTp_TxConfirmation* (FC or AF)	*FrTp_RxIndication* (CF or SF-x, FF-x (the latter two in case of retry))
FrTpTimeCs	*FrTp_TxConfirmation* (CF), *FrTp_RxIndication* (FC or AF (not after the last one))	*FrIf_Transmit* (CF)

Table 5: Start and Stop of the different timeouts and times

10.4.2 How to get an ISO 15765-2 compliant Channel / Connection

FRTP171: To achieve ISO 15765-2 compliance within a channel/connection, there are restrictions for some parameters. Those marked with a "*" are only relevant, if the features are compiled in (see section 10.2.2).

These and those are explained in the table below:

Parameter	***Allowed values***

FrTpAckType (*)	'FRTP_NO'
FrTpGrpSeg (*)	false
FrTpTc (*)	false
FrTpLm (*)	'FRTP_ISO', 'FRTP_ISO6'
N-PDU length	9 [FrTpAdrType == FRTP_OB, FrTpLm == FRTP_ISO6], 10 [FrTpAdrType == FRTP_OB, FrTpLm == FRTP_ISO], 11 [FrTpAdrType == FRTP_TB, FrTpLm == FRTP_ISO6], 12 [FrTpAdrType == FRTP_TB, FrTpLm == FRTP_ISO]

Table 6: Parameter Setting for ISO 15765-2 compliance

All not mentioned parameters can have arbitrary values.

10.4.3 Dependencies among the Parameters

FRTP172: There are several dependencies among the connection specific and channel specific configuration parameters:

- If FrTpMultRec sets the connection to be a 1:1 connection, then the value of FrTpGrpSeg does not play a role for this connection since it is only relevant for 1:n connections.
- If FrTpMultRec sets the connection to be a 1:n connection, then the values of FrTpAckType, FrTpMaxBs, and FrTpMaxRn do not play a role for this connection, since they are only relevant for 1:1 connections.
- If FrTpMultRec sets the connection to be a 1:n connection or FrTpAckType does not activate retry (FRTP_NO, FRTP_ACK_WITHOUT_RT) then the value of FrTpMaxBs does not play a role for this connections since it is only relevant in 1:1 connections within channels with retry being activated.

10.4.4 Timing Constraints

The following Constraints shall hold for the Timing parameters:

FRTP242: $V_E + \mathtt{FrTpTimeBr} + (\mathtt{FrTpTimeoutAr} * \mathtt{FrTpMaxAr}) + V_S < \mathtt{FrTpTimeoutBs}$

FRTP243: $V_s + \mathtt{FrTpTimeCs} + (\mathtt{FrTpTimeoutAs} * \mathtt{FrTpMaxAs}) + V_E < \mathtt{FrTpTimeoutCr}$

Where V_E is the time from Starting the BS Timer until recognition of the frame in the receiver TP and V_S is the time from starting the CR Timer until recognition of the frame in the sender TP.

10.4.5 Configuration Requirements on the FlexRay AUTOSAR Transport Layer

FRTP180: It has to be assured, that FrTpStMin < FrTpTimeoutCr since there will always be a timeout of the latter one otherwise.

FRTP181: The configuration of a connection and a channel shall be, of course, the same at the sender and the receiver side. Only the values of FrTpLa and FrTpRa are swapped.

FRTP275: If a channel references one PDU of a certain direction (received or transmitted) that is also referenced by another channel, both channels must reference exactly the same set of PDUs for this direction. This restriction ensures the pool semantics of referenced PDUs.

FRTP276: All PDUs of a PDU pool must have the same size, and all channels that reference these PDUs must have the same addressing type.

FRTP277: In the set of connections that are associated with a PDU pool, no two connections have the same address information, not even with reversed addresses.

FRTP278: The number of prioritized PDUs of all connections associated with a PDU pool must be less than the number of PDUs in the pool; otherwise, a set of active prioritized connections can lead to timeout in other connections.

10.4.6 Configuration Requirements on the FlexRay Interface

FRTP174: If more than one N-PDU is used for one N-SDU within a connection, the FrIf shall guarantee, that the N-PDUs (L-SDUs) are scheduled (sent over the bus) in the same order the FlexRay AUTOSAR Transport Layer uses them, i.e. in ascending order regarding the N-PDU IDs used in the FlexRay AUTOSAR Transport Layer. To simplify configuration, all PDUs of a pool shall be arranged such that they are always received in the same order in which they have been transmitted, independent of the current cycle in the FlexRay communication round.

This is necessary to avoid CFs coming out of order in a segmented transfer.

FRTP175: For every FrTp L-SDU the PDU-Update/Valid Information of the FrIf shall be activated unless this is the only PDU in a frame.

This is necessary to avoid Rx-Indication at the FrTp for in the current transfer not used N-PDUs or if e.g. in every 2nd FlexRay bus cycle an N-PDU is scheduled.

FRTP182: For the last PDU (in temporal order) of each PDU pool, a TxConfirmation shall be configured.