Document Title	Specification of FlexRay AUTOSAR Transport Layer
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	601
Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Document Change History

Date	Release	Changed by	Change Description
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes
2016-11-30	4.3.0	AUTOSAR Release Management	• Chapters Runtime Errors, and Transient Faults have been established • Development Error Tracer has been replaced by Default Error Tracer • Meta Data handling has been introduced • Requirements about handling negative TxConfirmations has been added.
2015-07-31	4.2.2	AUTOSAR Release Management	• Changed attribute Ecuc.postBuildVariantValue to false for FrArTpSduRxId and FrArTpSduTxId
2014-10-31	4.2.1	AUTOSAR Release Management	• Clarification regarding NULL pointer handling • Removed obsolete ECU configuration elements
2014-03-31	4.1.3	AUTOSAR Release Management	• Clarified meaning of FrArTpTc • Clarified requirements for sending FC(OVFLW) • Revised routing path const correctness • Harmonization of API descriptions

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2013-10-31	4.1.2	AUTOSAR Release Management	• Retry of FrIf_Transmit mechanism has been removed in case this API returns E_NOT_OK • Removed FRARTP prefix for fields of FrTp frames and used camel case notation consistently for EcuC parameters • Removed NotifResultType from ComStackTypes and replaced by Std_ReturnType in the APIs • Removed the 'Timing' row from the API table(s) of chapter 'Scheduled Functions • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Organization of PDUs in PDU pools • Dynamic assignment of Tx N-PDUs to connections at runtime • Reserved Tx N-PDUs for high priority connections • TP API improvements and fixes • Adapted to new BSW General
2011-12-22	4.0.3	AUTOSAR Administration	• Adapted to the 4.x TP API • Removed private types FrTp_ParameterValueType, FrTp_ChangeResultType, FrTp_CancelResultType, FrTp_PduInfoType • Added parameter configPtr to FrArTp_Init • Adapted service IDs of standardized Com Stack API functions

Document Change History

Date	Release	Changed by	Change Description
2011-04-15	4.0.2	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
2010-09-30	3.1.5	AUTOSAR Administration	• Added FRTP222, FRTP223 • Modified FRTP195 • Use parameter PduInfoType in callback RxIndication • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	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
2007-12-21	3.0.1	AUTOSAR Administration	• “Advice for users” revised • “Revision Information” added

Document Change History			
Date	**Release**	**Changed by**	**Change Description**
2007-01-24	2.1.15	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
2006-05-16	2.0	AUTOSAR Administration	• Document structure adapted to common Release 2.0 SWS Template.
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

Table of Contents

1 Introduction and functional overview 9

2 Acronyms and abbreviations 11

3 Related documentation 12

3.1 Input documents 12
3.2 Related standards and norms 12
3.3 Related specification 13

4 Constraints and assumptions 14

4.1 Limitations 14
4.2 Applicability to car domains 14

5 Dependencies to other modules 15

5.1 PDU Router 15
5.2 FlexRay Interface 16
5.3 ECU State Manager 17
5.4 File structure 17
5.4.1 Code file structure 17
5.4.2 Header file structure 17
5.4.3 Design Rules 18

6 Requirements traceability 19

7 Functional specification 20

7.1 Overview 20
7.1.1 Extensions of ISO 15765-2 20
7.1.2 Connections 20
7.1.3 Channels 20
7.1.4 PDU Pools 21
7.1.5 Active Connections 21
7.2 Protocol Processes 21
7.2.1 1:1 Connections 21
7.2.1.1 1:1 Connection in a channel without Acknowledgement 21
7.2.1.2 1:1 Connection in a channel with Acknowledgement without Retry 24
7.2.1.3 1:1 Connection in a channel with Acknowledgement with Retry 26
7.2.2 1:n Connections 29
7.3 Frame Layout 30
7.3.1 General 30
7.3.2 Single Frames (SF-x) 32
7.3.2.1 ISO 15765-2 Single Frame (SF-I) 32
7.3.2.2 Extended Single Frame (SF-E) 33
7.3.3 First Frames (FF-x) 34
7.3.3.1 First Frame ISO 15765-2 (FF-I) 35
7.3.3.2 First Frame Extended (FF-E) 35
7.3.4 Consecutive Frames 36
7.3.5 Flow Control (FC) 37

7.3.6 Acknowledgement Frame (AF) 39
7.3.7 Error Handling of the FT Field 41
7.3.8 Addressing Errors 43
7.4 Further Principles of Working 43
7.4.1 Decision of Segmentation 43
7.4.2 Scheduling of PDUs during Transmission 43
7.4.3 Detection of Receiving Connection 44
7.4.4 Single Frame Handling during Reception 44
7.4.5 Addressing with Meta Data 44
7.4.6 Sending and Receiving within the same connection 45
7.4.7 Behavior on Timeouts and Errors 45
7.4.7.1 Handling of negative TxConfirmations 46
7.4.7.2 No Acknowledgement configured for the Channel 46
7.4.7.3 Acknowledgement without Retry configured for the Channel 46
7.4.7.4 Acknowledgement with Retry configured for the Channel 47
7.4.8 Transmit Cancellation 47
7.4.9 Receive Cancellation 48
7.4.10 Parameter Changing 48
7.4.11 Data Handling, Block Size and WAIT-Frames 48
7.4.11.1 Unsegmented Transfer 49
7.4.11.2 Segmented Transfer 50
7.4.11.3 Buffer Locking 51
7.4.11.4 Data Bytes in First Frames 52
7.4.12 Ignored Frames 52
7.5 Buffer Access Modes in the FlexRay Interface 52
7.6 Error classification 52
7.6.1 Development Errors 52
7.6.2 Runtime Errors 53
7.6.3 Transient Faults 53
7.7 Error detection 53
7.8 Error notification 53
8 API specification 54
8.1 Imported types 54
8.2 Type definitions 54
8.3 Function definitions 54
8.3.1 Standard functions 55
8.3.1.1 FrArTp_GetVersionInfo 55
8.3.2 Initialization and Shutdown 55
8.3.2.1 FrArTp_Init 55
8.3.2.2 FrArTp_Shutdown 55
8.3.3 Normal Operation 56
8.3.3.1 FrArTp_Transmit 56
8.3.3.2 FrArTp_CancelTransmit 56
8.3.3.3 FrArTp_CancelReceive 57
8.3.3.4 FrArTp_ChangeParameter 57
8.4 Call-back notifications 58
8.4.1 FrArTp_TriggerTransmit 58
8.4.2 FrArTp_RxIndication 58
8.4.3 FrArTp_TxConfirmation 59

8.5 Scheduled functions 59
8.5.1 FrArTp_MainFunction 59
8.6 Expected Interfaces 59
8.6.1 Mandatory Interfaces 59
8.6.2 Optional Interfaces 60
9 Sequence diagrams 61
9.1 N-SDU Transmission 61
9.1.1 Unsegmented N-SDU Transmission 61
9.1.2 Segmented N-SDU Transmission without Retry 62
9.1.3 Segmented N-SDU Transmission with Retry 62
9.1.3.1 N-PDU Transmission during N-SDU Transmission 64
9.1.4 N-PDU Data Copying during N-SDU Transmission 65
9.1.4.1 Transmit Cancellation 66
9.2 N-SDU Reception 67
9.2.1 Unsegmented N-SDU Reception 67
9.2.2 N-PDU Data Copying during Unsegmented N-SDU Reception 67
9.2.3 Segmented N-SDU Reception 69
9.2.4 Wait for Buffer during Segmented N-SDU Reception 71
9.2.5 N-PDU Data Copying during Segmented N-SDU Reception 72
9.2.6 N-PDU Transmission during N-SDU Reception 72
9.2.7 Receive Cancellation 73
10 Configuration specification 74
10.1 How to read this chapter 74
10.2 Containers and configuration parameters 74
10.2.1 FrArTp 77
10.2.2 FrArTpGeneral 77
10.2.3 FrArTpChannel 79
10.2.4 FrArTpPdu 86
10.2.5 FrArTpConnection 88
10.2.6 FrArTpTxSdu 90
10.2.7 FrArTpRxSdu 90
10.2.8 FrArTpMultipleConfig 91
10.3 Published Information 92
10.4 Important Issues on Configuration 92
10.4.1 Start and Stop of the Timing Parameters 92
10.4.2 How to get an ISO 15765-2 compliant Channel / Connection 92
10.4.3 Dependencies among the Parameters 93
10.4.4 Timing Constraints 93
10.4.5 Configuration Requirements on the FlexRay AUTOSAR Transport Layer 93
10.4.6 Configuration Requirements on the FlexRay Interface 94
11 Not applicable requirements 95

1 Introduction and functional overview

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

The FlexRay AUTOSAR Transport Layer module resides between the PDU Router [6] and the FlexRay Interface [5] (see Figure 1, according to [2]). The main purpose of FlexRay AUTOSAR Transport Layer module 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 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 FlexRay AUTOSAR Transport Layer module, the PDU Router, the FlexRay NM and the XCP.

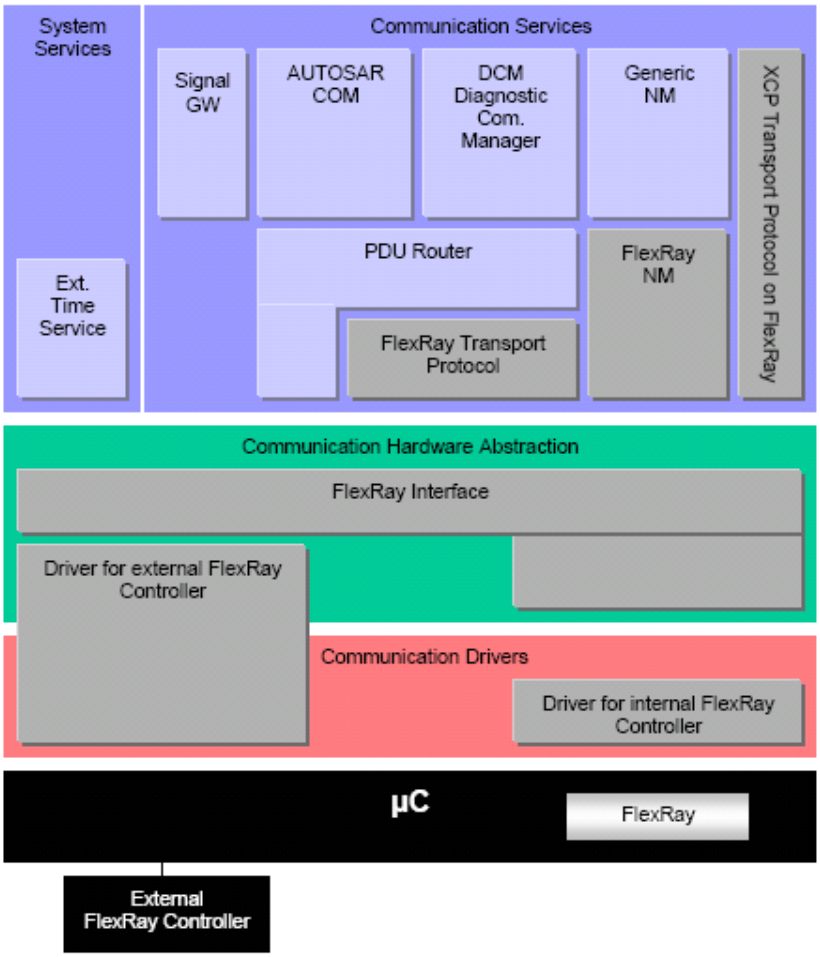


Figure 1: AUTOSAR FlexRay Layered Architecture

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 was used as standard in AUTOSAR release 3.x and below. Since AUTOSAR release 4.0, the standard FlexRay transport layer [11] 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. 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 the specification of Basic Software modules on existing standards. The FlexRay AUTOSAR Transport Layer specification is based on the international standard ISO 15765-2 (Diagnostics on CAN), which is the most common 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 deactivatable at compile time. Even if they are compiled in, they are still deactivatable 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. 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 FlexRay AUTOSAR Transport Layer module to interact with the FlexRay Interface.
Message	Synonymous for N-SDU or I-PDU.
N-PDU	This is a PDU of the FlexRay AUTOSAR 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 AUTOSAR 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 PDU Router 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
FrArTp	FlexRay AUTOSAR Transport Layer (derived from ISO 15765-2)
FrIsoTp	FlexRay ISO Transport Layer (ISO 10681-2)
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_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

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

[4] Requirements on FlexRay
AUTOSAR_SRS_FlexRay.pdf

[5] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface.pdf

[6] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[7] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[8] Specification of Standard Types
AUTOSAR_SWS_StandardTypes.pdf

[9] Specification of Platform Types
AUTOSAR_SWS_PlatformTypes.pdf

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

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

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

3.2 Related standards and norms

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

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

[15] FlexRay Communications System Protocol Specification Version 2.1

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [12] (SWS BSW General), which is also valid for FlexRay AUTOSAR Transport Layer.

Thus, the specification SWS BSW General shall be considered as additional and required specification for FlexRay AUTOSAR Transport Layer.

4 Constraints and assumptions

4.1 Limitations

AUTOSAR architecture defines protocol specific transport layer (CanTp, LinTp, Fr[Ar]Tp, etc.). 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.

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 FlexRay AUTOSAR Transport Layer module (FrArTp) and other AUTOSAR Basic Software modules. It contains brief descriptions of the services of the FrArTp that are called by other modules and of the services of other modules that are called by the FrArTp. The following picture gives a brief overview of the interactions.

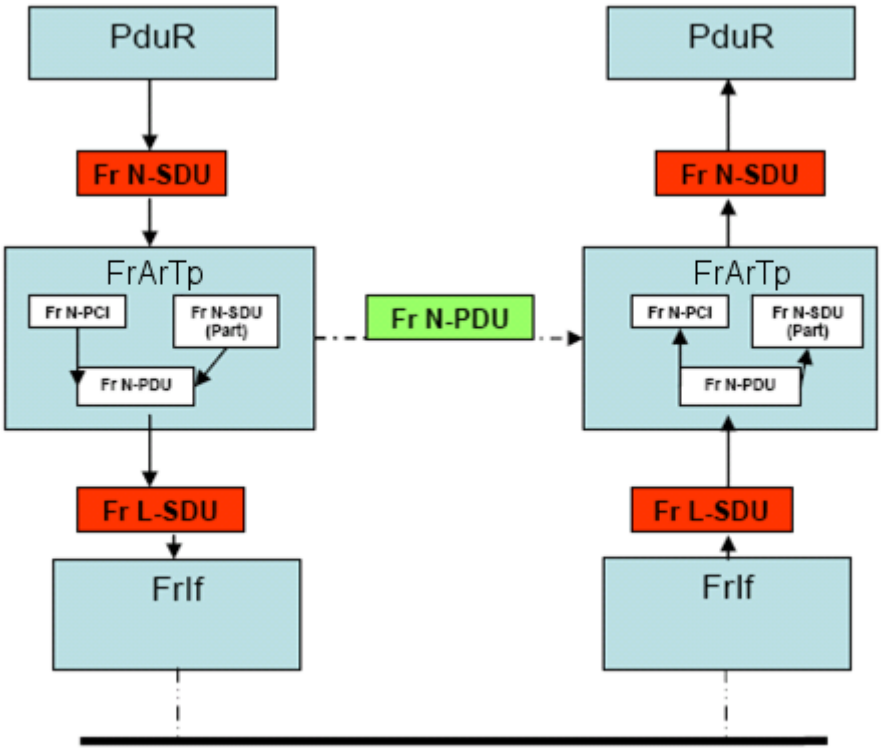


Figure 2: FrArTp interactions

5.1 PDU Router

The following services of the PDU Router are called by the FlexRay AUTOSAR Transport Layer module:

- *PduR_FrArTpStartOfReception*
  By this API service, the FlexRay AUTOSAR Transport Layer module informs the upper layer (e.g. DCM via PDU Router) that a new message is being received.
- *PduR_FrArTpCopyRxData*
  By this API service, the FlexRay AUTOSAR Transport Layer module provides the data of one received segmented message part (N-PDU) to the upper layer.

- *PduR_FrArTpRxIndication*
  By this API service, the FlexRay AUTOSAR Transport Layer module indicates the completed (un)successful reception of a message (N-SDU).
- *PduR_FrArTpCopyTxData*
  By this API service, the FlexRay AUTOSAR Transport Layer module asks the upper layer (e.g. DCM via PDU Router) of the message to provide data for the next segmented message part (N-PDU).
- *PduR_FrArTpTxConfirmation*
  By this API service, the FlexRay AUTOSAR Transport Layer module confirms the (un)successful sending of the complete message (N-SDU) to the actual sender (e.g. DCM).

The following services of the FlexRay AUTOSAR Transport Layer module are called by the PDU Router:

- *FrArTp_Transmit*
  By this API service, the sending of a message (N-SDU) is triggered.
- *FrArTp_CancelTransmit*
  By this API service, the sending of a message (N-SDU) is cancelled. This service is optional (per channel).
- *FrArTp_CancelReceive*
  By this API service, the receiving of a message (N-SDU) is cancelled. This service is optional (per channel).
- *FrArTp_ChangeParameter*
  By this API service, the STmin and BS values of a connection can be changed.

5.2 FlexRay Interface

The following services of the FlexRay Interface are called by the FlexRay AUTOSAR Transport Layer module:

- *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 FrArTp_TriggerTransmit.

The following services of the FlexRay AUTOSAR Transport Layer module are called by the FlexRay Interface:

- *FrArTp_RxIndication*
  By this API service, the FlexRay Interface indicates the reception of an FrArTp frame (N-PDU, please do not mistake this with a FlexRay frame) to the FrArTp. The FrArTp then processes this frame.
- *FrArTp_TxConfirmation*
  By this API service, the FlexRay Interface confirms the (un)successful sending of the frame containing the N-PDU over the FlexRay network.

- *FrArTp_TriggerTransmit*
  By this API service, the FlexRay Interface makes the FrArTp 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 FrArTp are called by the ECU State Manager:

- *FrArTp_Init*
  By this API service, all global variables are initialized and each connection is set into the idle state.
- *FrArTp_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

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

5.4.2 Header file structure

[SWS_FrArTp_00195] ⌈ The Header file structure shall include the following files named:
- FrArTp.h - general header file
- FrArTp_Cfg.h - pre-compile time configuration parameters
- Det.h – header file of the Default Error Tracer
- PduR_FrArTp.h – header file of PDU Router
- FrIf.h – header file of FrIf
- SchM_FrArTp.h – header file of TP related SchM declarations
- FrArTp_MemMap.h – header file for Memory Mapping
- Std_Types.h – header file for standard types
- ComStack_Types.h – header file for ComStack types
- FrArTp_Types.h – header file for FrArTp specific types⌋ (SRS_BSW_00346, SRS_BSW_00381, SRS_BSW_00383, SRS_BSW_00404)

[SWS_FrArTp_00222] ⌈ The FrArTp.h file shall include FrArTp_Types.h.⌋ ()

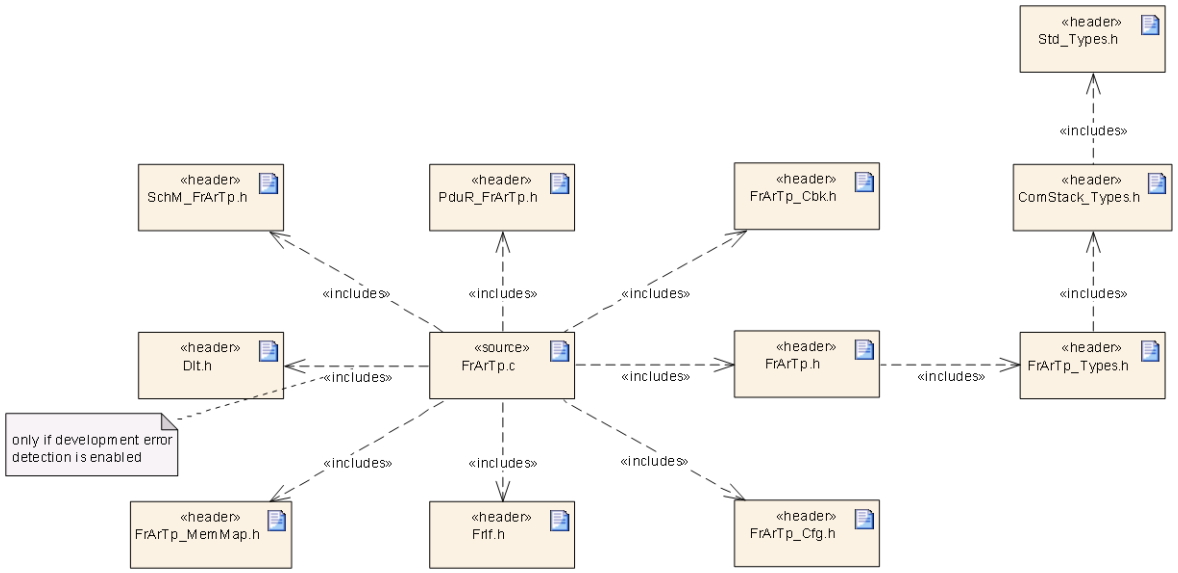


Figure 3: FrArTp include hierarchy

5.4.3 Design Rules

[SWS_FrArTp_00213] [The source code of the FrArTp shall not be processor and compiler dependent.] (SRS_BSW_00006)

6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00004	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	SWS_FrArTp_00201
SRS_BSW_00006	The source code of software modules above the µC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_FrArTp_00213
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_FrArTp_00147
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_FrArTp_00180, SWS_FrArTp_00181
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_FrArTp_00291
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_FrArTp_00148
SRS_BSW_00337	Classification of development errors	SWS_FrArTp_00179
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_FrArTp_00195
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_FrArTp_00149, SWS_FrArTp_00154
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_FrArTp_00195
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_FrArTp_00195
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_FrArTp_00195
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_FrArTp_00292
SRS_BSW_00411	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	SWS_FrArTp_00215
SRS_Fr_05075	The FlexRay Transport Layer implementation shall be independent of the network configuration	SWS_FrArTp_00149
SRS_Fr_05088	FlexRay Transport Layer's variables shall be initialized	SWS_FrArTp_00147
SRS_Fr_05089	The FlexRay Transport Layer services shall not be operational before initializing the module.	SWS_FrArTp_00179
SRS_Fr_05090	The FlexRay Transport Layer shall support per connection the ISO 10681-2 / ISO 15765-2 service N_ChangeParameter	SWS_FrArTp_00104
SRS_Fr_05093	A cancellation service of transmission shal be provided at any time	SWS_FrArTp_00099

7 Functional specification

The FlexRay AUTOSAR Transport Layer module (FrArTp) 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.

[SWS_FrArTp_00192] ⌈ The FlexRay AUTOSAR Transport Layer provides full duplex capabilities for PDU pools and connections.⌋ ()

[SWS_FrArTp_00201]⌈ The FrArTp shall perform a preprocessor-check if its source and header files belong to the same version.⌋ (SRS_BSW_00004)

7.1 Overview

7.1.1 Extensions of ISO 15765-2

[SWS_FrArTp_00009] ⌈ 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 FrArTpConcurrentConnections. 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 next to-be-transmitted frame(s) until enough data is available, and until the frame has been transferred to FrIf via *FrArTp_TriggerTransmit*.

7.2 Protocol Processes

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

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, FrArTpAdrType and FrArTpLm), there is no need to segment this message.

The transfer takes place as illustrated in Figure 4:

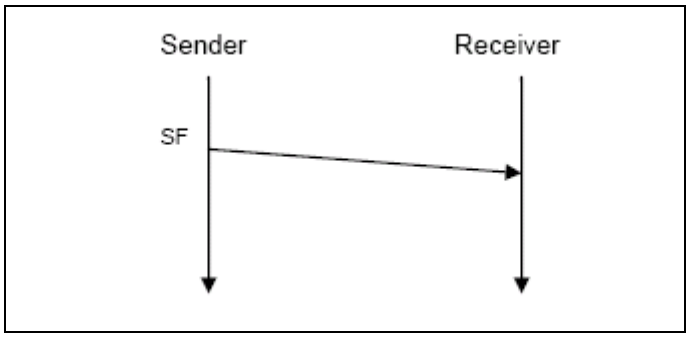


Figure 4: Unsegmented 1:1 transfer without acknowledgement

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 5:

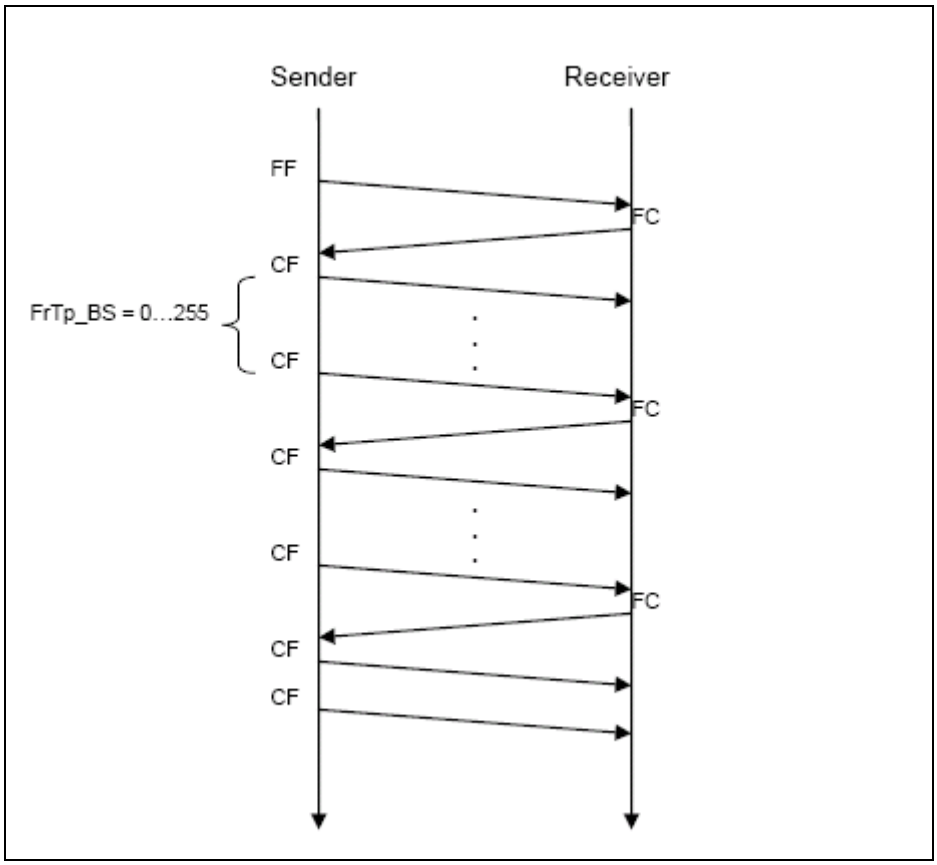


Figure 5: Segmented 1:1 transfer without acknowledgement

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 space for the whole message available.

There shall be a statically defined upper limit (FrArTpMaxWft) for the number of allowed WT's. If this number has been reached, the transmission shall be aborted and within *PduR_FrArTpTxConfirmation*, the result E_NOT_OK 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, .. 900 µs). 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 6:

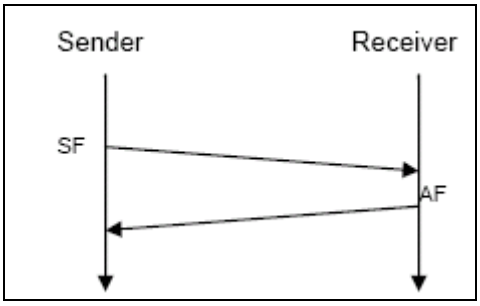


Figure 6: Unsegmented 1:1 transfer with Acknowledgement without Retry

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 FrArTpMaxRn, 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 7:

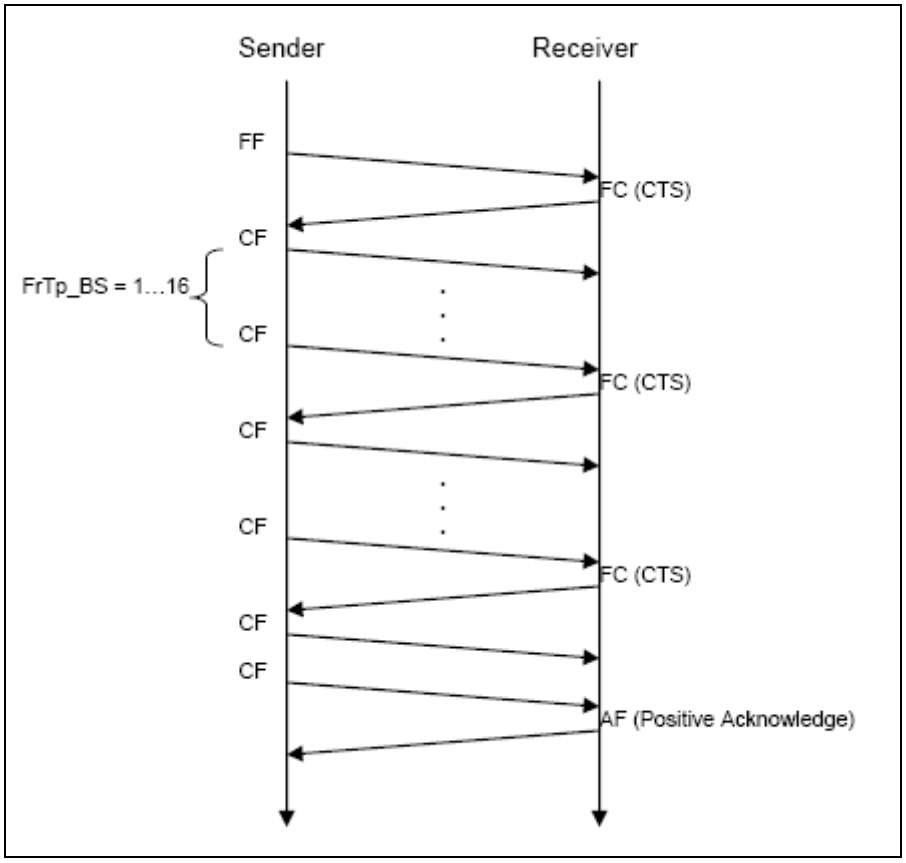


Figure 7: Segmented 1:1 transfer with Acknowledgement without Retry

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_FrArTpRxIndication*) the sender aborts the transmission and informs its upper layer (*PduR_FrArTpTxConfirmation*).

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 8 and Figure 9:

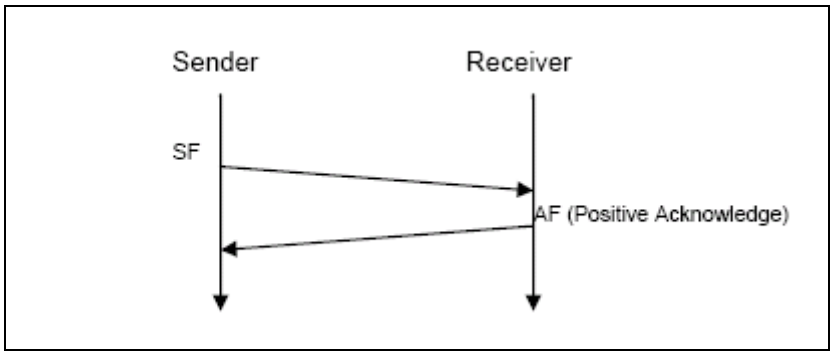


Figure 8: Unsegmented 1:1 transfer, Acknowledgement with Retry configured, Positive Acknowledgement

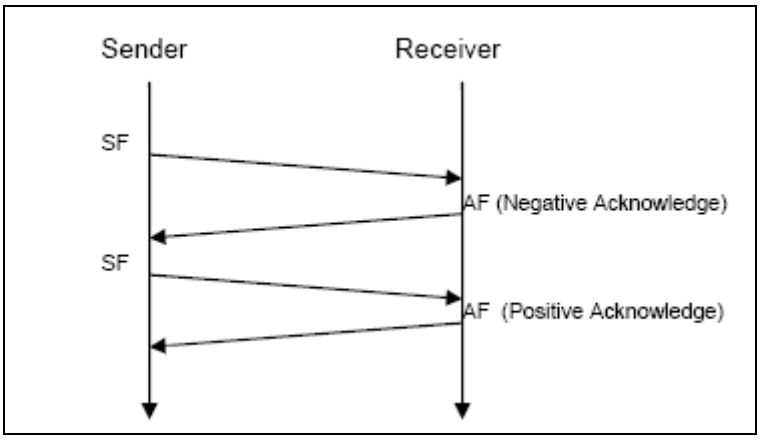


Figure 9: Unsegmented 1:1 transfer, Acknowledgement with Retry configured, Negative Acknowledgement

If in Figure 9 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 (FrArTpMaxRn) 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_FrArTpTxConfirmation* and *PduR_FrArTpRxIndication*, 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 FrArTpMaxRn limits the retry attempts.

The Retry mechanism is shown in Figure 10 for the case of a block size of 4:

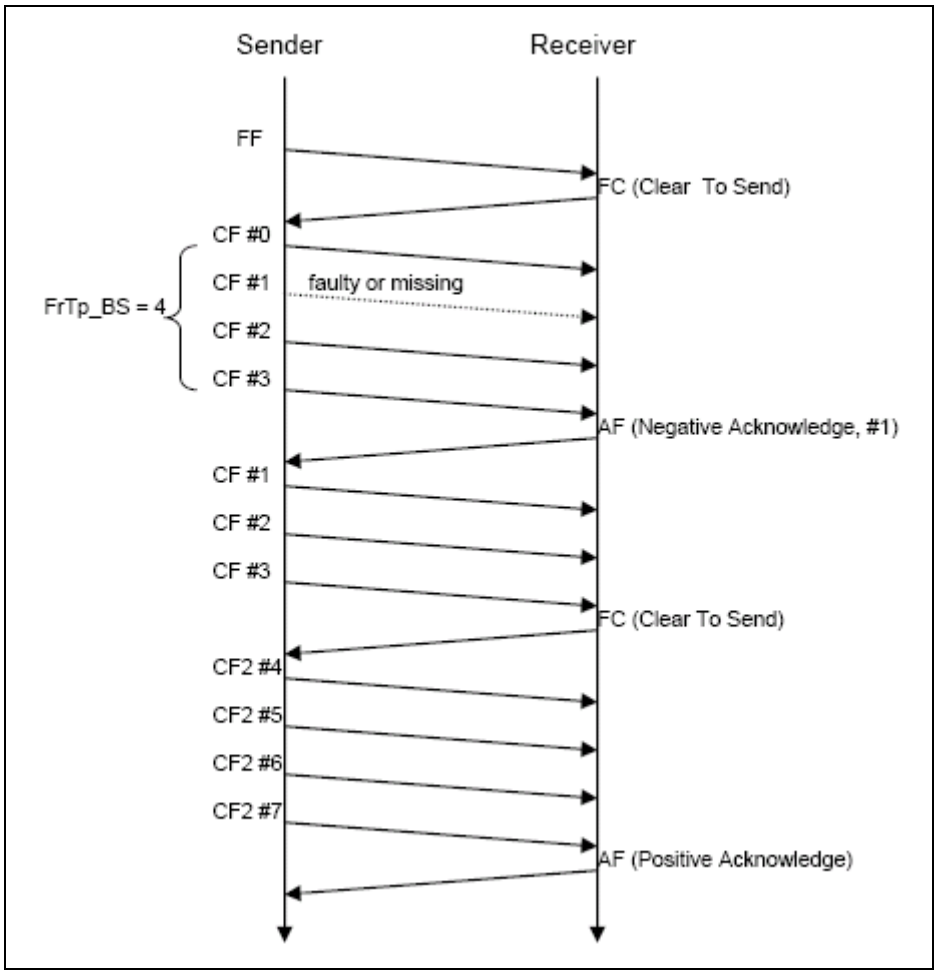


Figure 10: Segmented 1:1 transfer with Acknowledgement with Retry

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 AUTOSAR Transport Layer module 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:

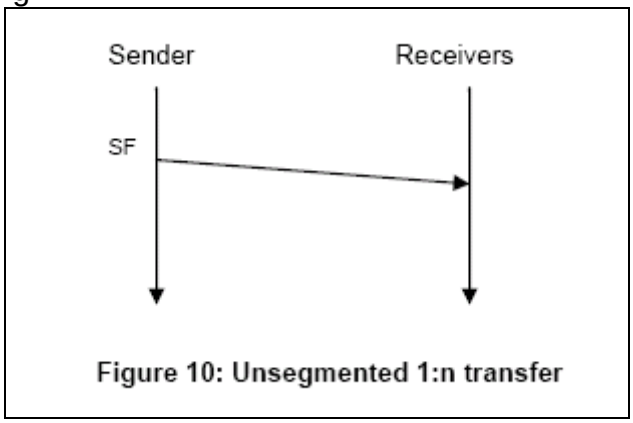


Figure 11: Unsegmented 1:n transfer

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.

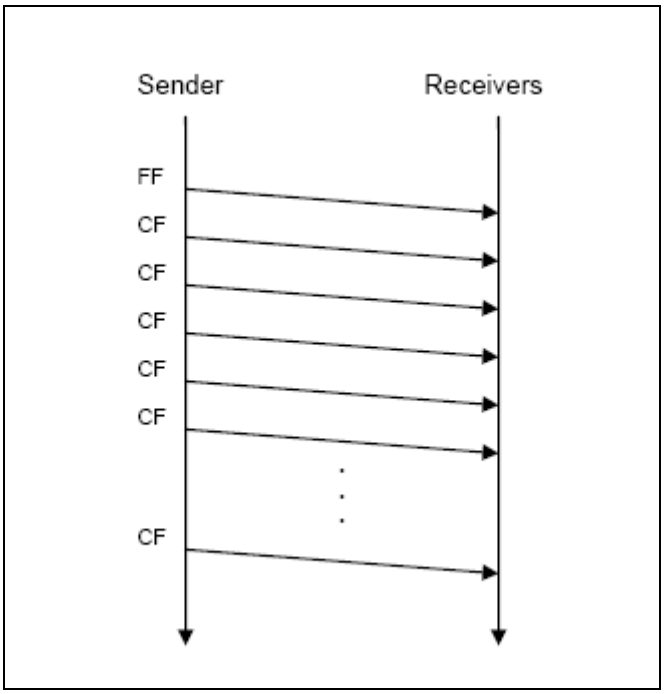


Figure 12: Segmented 1:n transfer

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

The distance of consecutive CFs is defined by the configuration parameter FrArTpStMinGrpSeg, similar to FrArTpStMin 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 13:

Figure 13: Structure of a FlexRay AUTOSAR transport protocol 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.

[SWS_FrArTp_00255] [When the FrArTp frame does not require the whole length of its N-PDU (in a SingleFrame, a FirstFrame, or the last ConsecutiveFrame 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

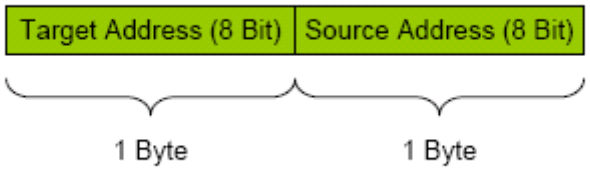


Figure 14: Address header for 1 Byte addressing

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

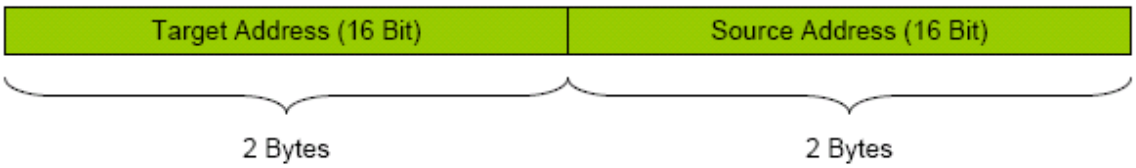


Figure 15: Address header for 2 Byte addressing

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

As seen in Figure 13, 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):

[SWS_FrArTp_00021] ⌈

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)

[SWS_FrArTp_00022] ⌈ 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. FrArTpLm = FRARTP_L4G) only SF-E is allowed.⌋ ()

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

[SWS_FrArTp_00023] ⌈ A SF-I looks as follows (address information header is not depicted):

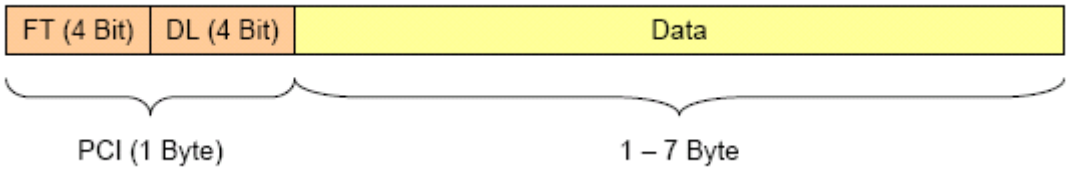


Figure 16: Single Frame ISO 15765-2⌋ ()

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.

[SWS_FrArTp_00024] ⌈ For ISO 15765-2 Single Frames the FT field shall be set to 0x0.⌋ ()

[SWS_FrArTp_00025] ⌈ The DL field states the amount of the actual data bytes, according to ISO 15765-2 the values 0x1 – 0x7 (0x6 in FRARTP_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.⌋ ()

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

Error Handling

[SWS_FrArTp_00028] DL field
⌈ Incoming SF-I frames with an invalid DL value of 0x0 or higher than 0x7 (0x6 in FRARTP_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**.

[SWS_FrArTp_00029] ⌈ An SF-E allows using the whole possible FlexRay payload of 254 Bytes for an unsegmented transfer. It looks as depicted in Figure 17:

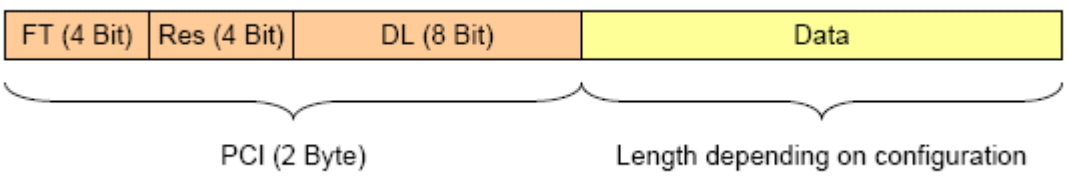


Figure 17: Single Frame Extended⌋ ()

[SWS_FrArTp_00030] ⌈ 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.⌋ ()

[SWS_FrArTp_00031] ⌈ 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.⌋ ()

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 [15]). The actual frame length can be derived considering the addressing mode and looking in the length statement of the corresponding PduInfoType struct.

Error Handling

[SWS_FrArTp_00286] 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.⌋ ()

[SWS_FrArTp_00287] General
⌈ If messages longer than allowed by ISO 15765-2 are not configured (FrArTpLm) 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.

[SWS_FrArTp_00034] ⌈ The FrArTp 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.⌋ ()

[SWS_FrArTp_00035] ⌈ 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.⌋ ()

[SWS_FrArTp_00036] 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)

[SWS_FrArTp_00237] ⌈ The figure below shows the layout of a FF-I:

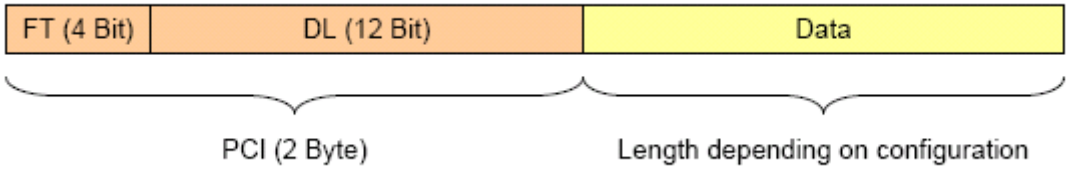


Figure 18: First Frame ISO 15765-2⌋ ()

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.

[SWS_FrArTp_00037] ⌈ For a FF-I, the FT field shall be set to 0x1.⌋ ()

[SWS_FrArTp_00299] ⌈ 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. ⌋ ()

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 FRARTP_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.

Error Handling

[SWS_FrArTp_00039] 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**.

[SWS_FrArTp_00238] ⌈ The layout of FF-E is shown below:

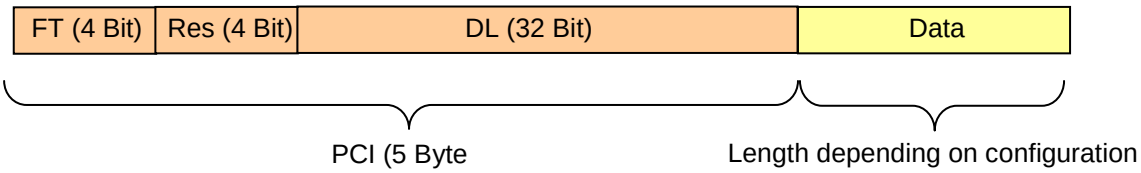


Figure 19: First Frame Extended⌋ ()

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.

[SWS_FrArTp_00054] ⌈ The DL field is 4 Byte long, so it allows transporting up to 2^{32}-1 bytes.⌋ ()

[SWS_FrArTp_00055] ⌈ The FT field is set to 0x5.⌋ ()

[SWS_FrArTp_00056] ⌈ 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.

Error Handling

[SWS_FrArTp_00057] DL field
⌈ 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

[SWS_FrArTp_00058] ⌈ If no error occurred, an FF-x is followed by Consecutive Frames until the whole message is transmitted.⌋ ()

[SWS_FrArTp_00059] **Not compliant to ISO 15765-2**
⌈ If configured for the specific channel, a Consecutive Frame can also appear in a 1:n connection.⌋ ()

[SWS_FrArTp_00239] ⌈ As shown below, Consecutive Frames consist of one byte PCI and the payload.

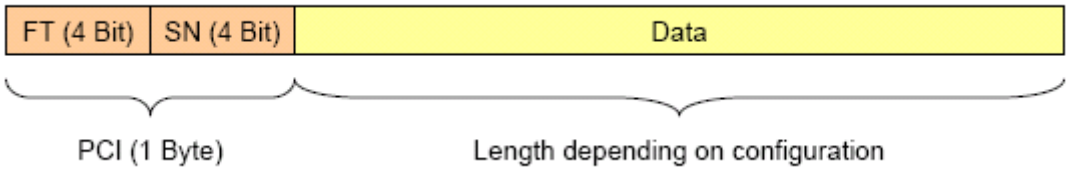


Figure 20: Consecutive Frame⌋ ()

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

[SWS_FrArTp_00060] ⌈ 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).⌋ ()

[SWS_FrArTp_00061] ⌈ 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.⌋ ()

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 FRARTP_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

[SWS_FrArTp_00063] 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_FrArTpRxIndication* the result E_NOT_OK 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 FrArTpMaxRn times).⌋ ()

7.3.5 Flow Control (FC)

[SWS_FrArTp_00064] ⌈ 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. ⌋ ()

[SWS_FrArTp_00240] ⌈ The layout of FC is shown below:

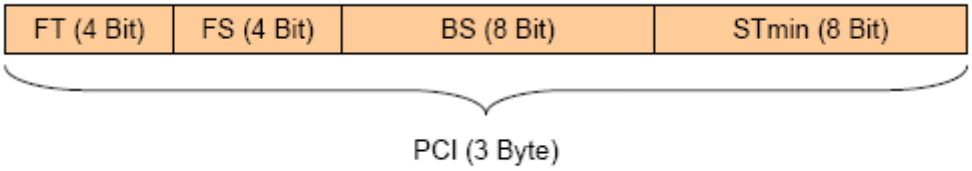


Figure 21: Flow Control frame⌋ ()

[SWS_FrArTp_00065]⌈ A Flow Control frame only consists of Protocol Control Information.⌋ ()

[SWS_FrArTp_00066]⌈ As usual, the FT field states the frame type, thus for Flow control frames, it shall be set to 0x3.⌋ ()

[SWS_FrArTp_00067]⌈ The FS field may contains 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 FrArTpTimeoutBs). 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)⌋ ()

[SWS_FrArTp_00068] ⌈ 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.⌋ ()

[SWS_FrArTp_00069] ⌈ 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.⌋ ()

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

Error Handling

[SWS_FrArTp_00285] FS field
⌈ If acknowledgment with Retry is configured, instead of abortion of the transfer, the frame shall be ignored.⌋ ()

[SWS_FrArTp_00244] BS field
⌈ 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_FrArTpTxConfirmation* shall be called with E_NOT_OK, otherwise the frame shall be ignored. ⌋ ()

[SWS_FrArTp_00245] 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. ⌋ ()

7.3.6 Acknowledgement Frame (AF)

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

[SWS_FrArTp_00072] ⌈ 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.⌋ ()

[SWS_FrArTp_00241] ⌈ The layout of FC is shown below:

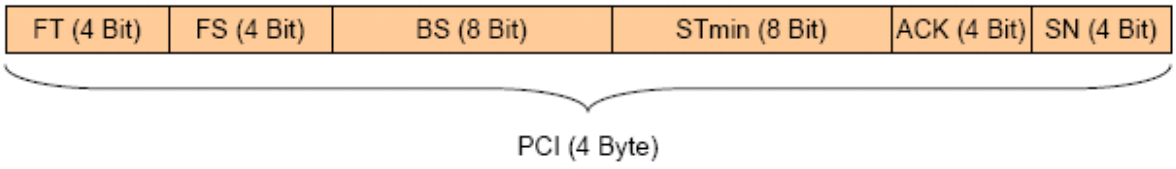


Figure 22: Acknowledgement Frame⌋ ()

[SWS_FrArTp_00073] ⌈ This frame is identified by the value 0x7 of the FT field.⌋ ()

[SWS_FrArTp_00074] ⌈ FS field is the same as in an FC frame.⌋ ()

[SWS_FrArTp_00075] ⌈ 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).⌋ ()

[SWS_FrArTp_00076] ⌈ STmin is the same as in FC frames.⌋ ()

[SWS_FrArTp_00077] ACK field gives the type of the acknowledgement, Positive (0x0) or Negative (0x1). All other values are reserved.⌋ ()

[SWS_FrArTp_00078] ⌈ SN field contains the number of the first faulty CF within the last block. All values are valid.⌋ ()

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.

[SWS_FrArTp_00284] FS field
⌈ In a segmented transfer, all values higher than 0x2 shall lead to the abortion of the transfer and *PduR_FrArTpTxConfirmation* shall be called with the result 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.⌋ ()

[SWS_FrArTp_00251] FS field with Acknowledgement
⌈ 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_FrArTpTxConfirmation* shall be called with the result E_NOT_OK.⌋ ()

[SWS_FrArTp_00246] BS field
⌈ The value 0x00 and all values higher than 0x10 shall cause the abortion of the transfer and *PduR_FrArTpTxConfirmation* shall be called with the result 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.⌋ ()

[SWS_FrArTp_00247] 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.⌋ ()

[SWS_FrArTp_00248] ACK field
⌈ Values higher than 0x1 are invalid and shall cause the abortion of the transfer and *PduR_FrArTpTxConfirmation* shall be called with the result 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.⌋ ()

[SWS_FrArTp_00249] 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_FrArTpTxConfirmation* shall be called with the result E_NOT_OK.
If additionally Retry is configured, the transfer shall not be aborted. Instead, the frame shall be ignored.⌋ ()

[SWS_FrArTp_00250] 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.] ()

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.

[SWS_FrArTp_00082] [A value of the FR_FT field higher than 0x7 shall always be ignored.] ()

[SWS_FrArTp_00083] [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_FrArTpRxIndication* with the result E_NOT_OK and process the SF-I as the start of a new reception.	Terminate the current reception, report a *PduR_FrArTpRxIndication* with the result E_NOT_OK 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:

[SWS_FrArTp_00283] SF-x, FF-x, CF/CF2 and FC frames
⌈ 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 FrArTpLm.

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.⌋ ()

[SWS_FrArTp_00252] AF frame
⌈ 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:

- **[SWS_FrArTp_00269]** ⌈ 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.⌋ ()
- **[SWS_FrArTp_00270]** ⌈ If a non-faulty AF with positive acknowledgement arrives during a block, it shall be ignored.⌋ ()

- **[SWS_FrArTp_00271]** ⌈ 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.⌋ ()
- **[SWS_FrArTp_00272]** ⌈ If a faulty AF arrives during a block, it shall be ignored.⌋ ()

7.3.8 Addressing Errors

SF-x frame
No restrictions.

[SWS_FrArTp_00086] FF-x and CF frames
⌈ If not explicitly configured by the parameter FrArTpGrpSeg for the particular channel, a FF-x or CF in a 1:n connection shall be ignored.⌋ ()

[SWS_FrArTp_00087] FC and AF frames
⌈ 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 FlexRay AUTOSAR Transport Layer module to segment a message (N-SDU) or not.
The values of the following parameters play a role hereby:
N-PDU length, FrArTpLm, FrArTpAdrType, FrArTpMultRec, FrArTpGrpSeg, and the length of the to-be-transmitted message (N-SDU).

[SWS_FrArTp_00091] ⌈ 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 (FrArTpAdrType) are needed to state to address information. The frames that are allowed to be utilized, and the payload they can carry depend on the value of FrArTpLm (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 (FrArTpMultRec), the parameter FrArTpGrpSeg states whether segmentation is allowed or not.
With all this information and the length of the to-be-transmitted N-SDU, the FrArTp 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.

[SWS_FrArTp_00256] ⌈ 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.⌋ ()

[SWS_FrArTp_00257] ⌈ 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.⌋ ()

[SWS_FrArTp_00258] ⌈ 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.⌋ ()

[SWS_FrArTp_00259] ⌈ It must be ensured that the last PDU of a PDU pool within a FlexRay cycle is always used by the scheduling.⌋ ()

[SWS_FrArTp_00260] ⌈ 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.

[SWS_FrArTp_00261] ⌈ 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

[SWS_FrArTp_00262] ⌈ 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_FrArTpStartOfReception* and, upon successful return of this function, *PduRFrArTpCopyRxData* and *PduR_FrArTpRxIndication*.⌋ ()

The behavior in case of unsuccessful reception is described in **[SWS_FrArTp_00298]** and **[SWS_FrArTp_00289]**.

7.4.5 Addressing with Meta Data

[SWS_FrArTp_00401] ⌈ During transmission, the FrArTp shall use addressing information provided by the upper layer via the meta data items SOURCE_ADDRESS_16 and TARGET_ADDRESS_16 as local address and remote address of the transmitted N-PDUs and to identify received flow control N-PDUs.⌋ ()

[SWS_FrArTp_00402] ⌈ During reception, the FrArTp shall forward addressing information received as remote address and local address in the N-PDU to the upper layer via the meta data items SOURCE_ADDRESS_16 and TARGET_ADDRESS_16, and shall use the same address information when transmitting flow control N-PDUs.⌋ ()

[SWS_FrArTp_00403] ⌈ If FrArTpLa and/or FrArTpRa are configured for a transmitted N-SDU, they are used even when the addressing information is provided by the upper layer. If not, the address information in the N-PDU shall be set according to the provided address information.⌋ ()

[SWS_FrArTp_00404] ⌈ If FrArTpLa and/or FrArTpRa are not configured for a received N-SDU, any received addressing information can be assigned to this N-SDU. N-SDUs with configured FrArTpLa and/or FrArTpRa shall be preferred during reception over those without these configuration parameters.⌋ ()

7.4.6 Sending and Receiving within the same connection

[SWS_FrArTp_00094] ⌈ The FrArTp 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 *FrArTp_Transmit()* occurs, the local peer becomes the sender in this connection (Source Address of TP frame = FrArTpLa, Target Address of TP frame = FrArTpRa). Otherwise, if an *FrArTp_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 = FrArTpRa, Target Address of TP frame = FrArTpLa).

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.

7.4.7 Behavior on Timeouts and Errors

[SWS_FrArTp_00095] ⌈ The behavior in case a timeout occurs depends on the value of FrArTpAckType, i.e. what kind of acknowledgement is configured for the corresponding channel.⌋ ()

[**SWS_FrArTp_00302**] ⌈ The FrArTp shall abort the connection when FrIf_Transmit returns E_NOT_OK⌋ ()

7.4.7.1 Handling of negative TxConfirmations

[**SWS_FrArTp_00410**] ⌈ If the API *FrArTp_TxConfirmation* called with result E_NOT_OK for an ongoing reception process, the reception of the SDU shall be terminated immediately and within *PduR_FrArTpRxIndication* the result E_NOT_OK shall be returned.⌋ ()

[**SWS_FrArTp_00411**] ⌈ If the API *FrArTp_TxConfirmation* called with result E_NOT_OK for an ongoing transmission process, the transmission of the SDU shall be terminated immediately and within *PduR_FrArTpTxConfirmation* the result E_NOT_OK shall be returned.⌋ ()

7.4.7.2 No Acknowledgement configured for the Channel

In this case, the behavior shall be as described in [13], i.e.:

- **[SWS_FrArTp_00282]** ⌈ If the AS timer (FrArTpTimeoutAs) expires, depending on the value of FrArTpMaxAs, sending shall be retried (because of still remaining attempts) or the transmission shall be aborted, and within *PduR_FrArTpTxConfirmation* the result E_NOT_OK shall be returned.⌋ ()
- **[SWS_FrArTp_00263]** ⌈ If the AR timer (FrArTpTimeoutAr) expires, depending on the value of FrArTpMaxAr, sending shall be retried (because of still remaining attempts) or the transmission shall be aborted, and within *PduR_FrArTpRxIndication* the result E_NOT_OK shall be returned.⌋ ()
- **[SWS_FrArTp_00264]** ⌈ If the BS timer (FrArTpTimeoutBs) expires, the transmission shall be aborted, and within *PduR_FrArTpTxConfirmation* the result E_NOT_OK shall be returned.⌋ ()
- **[SWS_FrArTp_00265]** ⌈ If the CR timer (FrArTpTimeoutCr) expires, the transmission shall be aborted, and within *PduR_FrArTpRxIndication,* the result E_NOT_OK shall be returned. If previously in the current block a sequence error occurred, this error shall be reported at the block end in *PduR_FrArTpRxIndication*.⌋ ()

7.4.7.3 Acknowledgement without Retry configured for the Channel

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

In this case, the behavior is the following:

- **[SWS_FrArTp_00281]** ⌈ In case of a timeout of timer AS, AR or BS, the behavior shall be as mentioned in section 7.4.7.2.⌋ ()
- **[SWS_FrArTp_00266]** ⌈ If the CR timer (FrArTpTimeoutCr) expires, an AF with negative acknowledgement shall be sent, the transmission shall be aborted, and within *PduR_FrArTpRxIndication,* the result E_NOT_OK shall be returned. If previously in the current block a sequence error occurred, this error shall be reported at the block end in *PduR_FrArTpRxIndication*.⌋ ()

7.4.7.4 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:

- **[SWS_FrArTp_00280]** ⌈ In case of a timeout of timer AS or AR, the behavior shall be as mentioned in section 7.4.7.2.⌋ ()
- **[SWS_FrArTp_00267]** ⌈ If the BS timer (FrArTpTimeoutBs) expires, the sender shall retransmit the whole block up to FrArTpMaxRn times. After that, the transmission shall be aborted, and within *PduR_FrArTpTxConfirmation* the result E_NOT_OK shall be returned.⌋ ()
- **[SWS_FrArTp_00268]** ⌈ If the CR timer (FrArTpTimeoutCr) expires, the receiver shall send an AF with negative acknowledgement and the sequence number of the missed CF. This shall be done up to FrArTpMaxRn times. After that, the transmission shall be aborted, and within *PduR_FrArTpRxIndication,* the result E_NOT_OK shall be returned. If previously in the current block a sequence error occurred, this error shall be reported at the block end in *PduR_FrArTpRxIndication*.⌋ ()

7.4.8 Transmit Cancellation

[SWS_FrArTp_00099] ⌈ This feature can be (de)activated by static configuration (parameter FrArTpTc). Transmit Cancellation is triggered by the call of *FrArTp_CancelTransmit*.⌋ (SRS_Fr_05093)

[SWS_FrArTp_00236] ⌈ When a transmission is still in progress, FrArTp_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, FrArTp_CancelTransmit shall return E_NOT_OK.⌋ ()

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

- **[SWS_FrArTp_00279]** ⌈ If no transmit request is pending for the corresponding connection, there is nothing to do.⌋ ()
- **[SWS_FrArTp_00273]** ⌈ If a request is pending but the transmission has not been started, the FrArTp shall immediately call PduR_FrArTpTxConfirmation and free the connection.⌋ ()

- **[SWS_FrArTp_00274]** ⌈ If the transmission already has been started, the FrArTp shall immediately call PduR_FrArTpTxConfirmation, 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 either be transferred to the FrIf as if they had not been canceled, or E_NOT_OK shall be returned.⌋ ()

[SWS_FrArTp_00103] ⌈ If a transfer was cancelled by the call of *FrArTp_CancelTransmit*, *PduR_FrArTpTxConfirmation* shall be called with E_NOT_OK.⌋ ()

7.4.9 Receive Cancellation

[SWS_FrArTp_00224] ⌈ If development error detection is enabled the function FrArTp_CancelReceive shall check the validity of FrArTpRxSduId parameter.⌋ ()

[SWS_FrArTp_00226] ⌈ The FrArTp shall abort the reception of the current N-SDU if the service FrArTp_CancelReceive provides a valid FrArTpRxSduId.⌋ ()

[SWS_FrArTp_00227] ⌈ The FrArTp shall reject the request for receive cancellation by returning E_NOT_OK when
a) the cancelled connection is not active, or when
b) the FrArTp has already received the last frame of an unacknowledged connection, or when
c) the FrArTp has already provided the final AF of an acknowledged connection.⌋ ()

[SWS_FrArTp_00228] ⌈ If the FrArTp_CancelReceive service has been successfully executed, the FrArTp shall call the PduR_FrArTpRxIndication with E_NOT_OK.⌋ ()

7.4.10 Parameter Changing

[SWS_FrArTp_00104] ⌈ The FrArTp also supports the optional service for changing the parameters FrArTpMaxBs and FrArTpStMin / FrArTpStMinGrpSeg mentioned in [13] via the API call *FrArTp_ChangeParameter*. A change is not possible during an ongoing reception and shall lead to the return value E_NOT_OK. ⌋ (SRS_Fr_05090)

7.4.11 Data Handling, Block Size and WAIT-Frames

The FlexRay AUTOSAR Transport Layer does not provide message buffers, neither for sending nor for receiving. Instead, it provides received data and requests transmitted data chunk wise from/to the upper layer.

[SWS_FrArTp_00221] ⌈ When a new reception is initiated by the reception of a FF or SF, the TP checks for the availability of the associated channel and then calls

PduR_FrArTpStartOfReception to inform the upper layer of the expected message size, and to retrieve information about the currently available buffer. With this call, the FrArTp provides the total size of the received data (SDU), and the data and size of the FF or SF to the upper layer. When this call succeeds, the connection is set to established, and *PduR_FrArTpCopyRxData* is called to provide the payload of the frame to the upper layer. ⌋ ()

[SWS_FrArTp_00230] ⌈ When a new transmission is initiated by the call of *FrArTp_Transmit*, the TP checks for the availability of the associated channel, and sets the connection to established. Then the TP calls *PduR_FrArTpCopyTxData* to acquire the data for the SF or FF and following CFs.⌋ ()

[SWS_FrArTp_00105] ⌈ Depending on the message length and configuration of the FrArTp, a segmented or an unsegmented transfer will take place.⌋ ()

[SWS_FrArTp_00232] ⌈ The API function *PduR_FrArTpCopyTxData* has a parameter named `retry`, which is a pointer to a structure of type `RetryInfoType`. When Retry is disabled, `retry` shall always be set to `NULL`. Otherwise, the different values of `retry.TpDataState` shall be used to handle retries.⌋ ()

7.4.11.1 Unsegmented Transfer

[SWS_FrArTp_00106] ⌈ At the sender side, this principle works as follows:

1. The PDU Router calls the service *FrArTp_Transmit*.
2. The FrArTp shall call *PduR_FrArTpCopyTxData* in order to get all the data bytes of the SF-x. `retry.TpDataState` shall be set to `TP_CONFPENDING` when Retry is enabled.⌋ ()

[SWS_FrArTp_00107] ⌈ If *PduR_FrArTpCopyTxData* for the SF-x returns `BUFREQ_E_BUSY`, the call shall be repeated until FrArTpTimeCs expires. When the return value is `BUFREQ_E_NOT_OK` or after FrArTpTimeCs expired, the transfer shall be aborted by calling *PduR_FrArTpTxConfirmation* with `E_NOT_OK`.⌋ ()
[SWS_FrArTp_00233] ⌈ If Retry is enabled, the SF-x shall be sent again after reception of a negative AF. The FrArTp shall finish the transfer after reception of a positive AF by the call to *PduR_FrArTpTxConfirmation* with `E_OK`.⌋ ()

[SWS_FrArTp_00108] ⌈ At the receiver side, the principle works as follows:

1. FrIf calls the service *FrArTp_RxIndication*.
2. The FrArTp shall call *PduR_FrArTpStartOfReception* to prepare reception.
3. The FrArTp shall call *PduR_FrArTpCopyRxData* to forward SF data.⌋ ()

[SWS_FrArTp_00298] ⌈ If *PduR_FrArTpStartOfReception* for the SF-x returns `BUFREQ_E_NOT_OK` or `BUFREQ_E_OVFL`, the transfer shall be aborted immediately without calling *PduR_FrArTpRxIndication*. ⌋ ()

[SWS_FrArTp_00289] ⌈ If the available buffer reported by *PduR_FrArTpStartOfReception* is too small for the SF-x, or if *PduR_FrArTpCopyRxData* for the SF-x returns `BUFREQ_E_NOT_OK`, the FrArTp shall abort the transfer and call *PduR_FrArTpRxIndication* with `E_NOT_OK`. ⌋ ()

[SWS_FrArTp_00253] ⌈ If acknowledgement is configured: In case of failing to copy the received data (either `BUFREQ_E_NOT_OK` or `BUFREQ_E_OVFL` was returned, or the available buffer is too small), an AF with a negative acknowledgement and FS = OVFLW is sent back to the sender.⌋ ()

7.4.11.2 Segmented Transfer

[SWS_FrArTp_00110] ⌈ At the sender side, this principle works as follows:

1. The PDU Router calls the service *FrArTp_Transmit*.
2. The FrArTp shall call *PduR_FrArTpCopyTxData* in order to get the data bytes of the FF and following CFs.⌋ ()

[SWS_FrArTp_00111] ⌈ When Retry is enabled, the TP sends the FF-x without data. After reception of an FC, the data for the first CF is acquired with `retry.TpDataState` set to `TP_DATACONF`. For the following CFs, `retry.TpDataState` shall be set to `TP_CONFPENDING`.⌋ ()

[SWS_FrArTp_00234] ⌈ When Retry is enabled, after reception of a negative AF, the last block must be retransmitted. To achieve this, the data for the first CF of the block is acquired with `retry.TpDataState` set to `TP_DATARETRY` while `retry.TxTpDataCnt` contains the size of the previously sent block in bytes. The buffer of the last block in the upper layer is only freed after reception of a positive AF by the call to *PduR_FrArTpTxConfirmation* with `E_OK` .⌋ ()

[SWS_FrArTp_00296] ⌈ If *PduR_FrArTpCopyTxData* for the FF-x (Retry not configured) or any of the CFs returns `BUFREQ_E_BUSY`, the call shall be repeated until `FrArTpTimeCs` expires.⌋ ()

[SWS_FrArTp_00293] ⌈ If *PduR_FrArTpCopyTxData* returns `BUFREQ_E_NOT_OK` or when FrArTpTimeCs expired, the transfer shall be aborted by calling *PduR_FrArTpTxConfirmation* with `E_NOT_OK`.⌋ ()

[SWS_FrArTp_00114] ⌈ At the receiver side, this principle works as follows:

1. FrIf calls the service *FrArTp_RxIndication*.
2. The FrArTp shall call *PduR_FrArTpStartOfReception* to prepare reception.
3. The FrArTp shall call *PduR_FrArTpCopyRxData* to forward FF and CF data.⌋ ()

[SWS_FrArTp_00115] ⌈ The block size used during reception is constant. The value is configured via FrArTpMaxBs, and can be changed via the API *FrArTp_ChangeParameter*.⌋ ()

[SWS_FrArTp_00294] ⌈ If Retry is not enabled, and *PduR_FrArTpStartOfReception* returns an available buffer size that is too small for the FF-x, FrArTp shall abort the transfer and call *PduR_FrArTpRxIndication* with `E_NOT_OK`.⌋ ()

[SWS_FrArTp_00300] ⌈ If Retry is enabled, and *PduR_FrArTpStartOfReception* returns an available buffer size that is too small for the first block, FrArTp shall call *PduR_FrArTpCopyRxData* with `info.SduLength` equal to 0 until the available buffer is large enough for the first block. The calls shall be repeated until FrArTpTimeBr expires.⌋ ()

[SWS_FrArTp_00297] ⌈ If *PduR_FrArTpCopyRxData* for the FF-x (Retry not enabled) or for the last CF of a block (independent of Retry configuration) returns an available buffer size that is not large enough for the next block, *PduR_FrArTpCopyRxData* shall be called repeatedly with `info.SduLength` equal to 0 until the available buffer is large enough. The calls shall be repeated until FrArTpTimeBr expires.⌋ ()

[SWS_FrArTp_00301] ⌈ When FrArTpTimeBr expires during calls to *PduR_FrArTpCopyRxData* with `info.SduLength` equal to 0, a WAIT frame (FC frame with FS = WT) shall be sent, and the retry phase shall start again, but at most FrArTpMaxWft times.⌋ ()

[SWS_FrArTp_00295] ⌈ If *PduR_FrArTpCopyRxData* returns `BUFREQ_E_NOT_OK`, or when FrArTpMaxWft expired, the transfer shall be aborted and *PduR_FrArTpRxIndication* shall be called with `E_NOT_OK`.⌋ ()

[SWS_FrArTp_00117] ⌈ In case of failing to copy the received data, the remaining CFs of the current block shall be discarded. When the failure occured 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 initial call to PduR_FrTpStartOfReception returned BUFREQ_E_OVFL.⌋ ()

7.4.11.3 Buffer Locking

At the sender side, the originator of the transmission (e.g. DCM or COM) shall not change the buffer after a successful Transmit call until the connection is closed by a call to TxConfirmation. When a cyclic buffer is used, the data in this buffer must be kept until it is explicitly freed by *PduR_FrArTpCopyTxData* with `retry.TpDataState` set to `TP_DATACONF`.

At the receiver side, the module that provides the receive buffer (e.g. DCM or COM) shall not change this buffer after a successful call to StartOfReception until the connection is closed by a call to *PduR_FrArTpRxIndication*. When a cyclic buffer is

used, it may assume that all data provided via *PduR_FrArTpCopyRxData* is valid and may be discarded immediately.

7.4.11.4 Data Bytes in First Frames

[SWS_FrArTp_00120] Not compliant to ISO 15765-2
⌈ As stated in 7.4.11.2, 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 to retain backwards compatibility to the AUTOSAR release 3 FrTp on bus level.

[SWS_FrArTp_00121] ⌈ If acknowledgment without Retry (or no acknowledgement) is configured, there are data bytes within an FF-x.⌋ ()

7.4.12 Ignored Frames

[SWS_FrArTp_00139] ⌈ 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.⌋ ()

[SWS_FrArTp_00140] ⌈ 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

[SWS_FrArTp_00187] ⌈ The FrArTp 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 FrArTp's 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 module.

7.6.1 Development Errors

[SWS_FrArTp_00179] ⌈ Development Error Types

Type or error	*Related error code*	*Value*

		[hex]
API service called while module is not initialized	FRARTP_E_UNINIT	0x1
API service called with invalid pointer	FRARTP_E_PARAM_POINTER	0x2
API service called with invalid SDU or PDU ID	FRARTP_E_INVALID_PDU_SDU_ID	0x3
Invalid configuration set selection	FRARTP_E_INIT_FAILED	0x4

Table 4: Development Error Types

⌋ (SRS_BSW_00337, SRS_Fr_05089)

7.6.2 Runtime Errors

Currently there are no Runtime Errors can be reported by the module.

7.6.3 Transient Faults

Currently there are no Transient Faults can be reported by the module.

7.7 Error detection

[SWS_FrArTp_00291] ⌈ If development error detection is enabled, all APIs with a parameter containing an SDU or a PDU identifier shall check the identifier and raise the development error FRARTP_E_INVALID_PDU_SDU_ID when the identifier has not been configured.⌋ (SRS_BSW_00323)

[SWS_FrArTp_00292] ⌈ If development error detection is enabled, the FrArTp shall raise the development error FRARTP_E_UNINIT when any API (other than FrArTp_GetVersionInfo) is called before initialization via FrArTp_Init.⌋ (SRS_BSW_00407)

7.8 Error notification

For details, refer to the sections 7.3 “Error Detection” and 7.4 “Error notification” in *SWS_BSWGeneral.*

8 API specification

8.1 Imported types

The following types are defined within AUTOSAR and used for the FrArTp:
[SWS_FrArTp_00141] [

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

] ()

8.2 Type definitions

[SWS_FrArTp_00288] [

Name:	`FrArTp_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.

8.3.1 Standard functions

8.3.1.1 FrArTp_GetVersionInfo

[SWS_FrArTp_00215] [

Service name:	FrArTp_GetVersionInfo	

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

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

] (SRS_BSW_00411)

8.3.2 Initialization and Shutdown

8.3.2.1 FrArTp_Init

[SWS_FrArTp_00147] [

Service name:	FrArTp_Init	

Syntax:	`void FrArTp_Init(`
	`  const FrArTp_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 AUTOSAR Transport Layer and sets all states to idle.	

] (SRS_BSW_00101, SRS_Fr_05088)
Please note: The call of this service is mandatory before using the FrArTp for further processing.

8.3.2.2 FrArTp_Shutdown

[SWS_FrArTp_00148] [

Service name:	FrArTp_Shutdown

Syntax:	`void FrArTp_Shutdown(`
	`  void`
	`)`

Service ID[hex]:	0x01

Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This service closes all pending transport protocol connections by simply stopping operation, frees all resources and stops the FrArTp Module

] (SRS_BSW_00336)

8.3.3 Normal Operation

8.3.3.1 FrArTp_Transmit

[SWS_FrArTp_00149] [

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

] (SRS_BSW_00369, SRS_Fr_05075)

8.3.3.2 FrArTp_CancelTransmit

[SWS_FrArTp_00150] [

Service name:	FrArTp_CancelTransmit	
Syntax:	`Std_ReturnType FrArTp_CancelTransmit( PduIdType TxPduId )`	
Service ID[hex]:	0x4a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	Identification of the PDU to be cancelled.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Cancellation was executed successfully by the destination module. E_NOT_OK: Cancellation was rejected by the destination

		module.
Description:	Requests cancellation of an ongoing transmission of a PDU in a lower layer communication module.	

] ()

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

8.3.3.3 FrArTp_CancelReceive

[SWS_FrArTp_00229] [

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

] ()

8.3.3.4 FrArTp_ChangeParameter

[SWS_FrArTp_00151] [

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

] ()

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 FrArTp_TriggerTransmit

[SWS_FrArTp_00154] [

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

] (SRS_BSW_00369)
Please note: This function might be called in interrupt context

8.4.2 FrArTp_RxIndication

[SWS_FrArTp_00152] [

Service name:	FrArTp_RxIndication	
Syntax:	`void FrArTp_RxIndication( PduIdType RxPduId, const PduInfoType* PduInfoPtr )`	
Service ID[hex]:	0x42	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	RxPduId	ID of the received PDU.
	PduInfoPtr	Contains the length (SduLength) of the received PDU, a pointer to a buffer (SduDataPtr) containing the PDU, and the MetaData related to this PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	

Description:	Indication of a received PDU from a lower layer communication interface module.

] ()

8.4.3 FrArTp_TxConfirmation

[SWS_FrArTp_00153] [

Service name:	FrArTp_TxConfirmation	

Syntax:	`void FrArTp_TxConfirmation(`
	`  PduIdType TxPduId,`
	`  Std_ReturnType result`
	`)`	

Service ID[hex]:	0x40	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different PduIds. Non reentrant for the same PduId.	
Parameters (in):	TxPduId	ID of the PDU that has been transmitted.
	result	E_OK: The PDU was transmitted. E_NOT_OK: Transmission of the PDU failed.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.	

] ()

8.5 Scheduled functions

8.5.1 FrArTp_MainFunction

[SWS_FrArTp_00162] [

Service name:	FrArTp_MainFunction

Syntax:	`void FrArTp_MainFunction(`
	`  void`
	`)`

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

] ()

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

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.

[SWS_FrArTp_00219] [

API function	***Description***

FrIf_Transmit	Requests transmission of a PDU.
PduR_FrArTpCopyRxData	This function is called to provide the received data of an I-PDU segment (N-PDU) to the upper layer. Each call to this function provides the next part of the I-PDU data. The size of the remaining buffer is written to the position indicated by bufferSizePtr.
PduR_FrArTpCopyTxData	This function is called to acquire the transmit data of an I-PDU segment (N-PDU). Each call to this function provides the next part of the I-PDU data unless retry->TpDataState is TP_DATARETRY. In this case the function restarts to copy the data beginning at the offset from the current position indicated by retry->TxTpDataCnt. The size of the remaining data is written to the position indicated by availableDataPtr.
PduR_FrArTpRxIndication	Called after an I-PDU has been received via the TP API, the result indicates whether the transmission was successful or not.
PduR_FrArTpStartOfReception	This function is called at the start of receiving an N-SDU. The N-SDU might be fragmented into multiple N-PDUs (FF with one or more following CFs) or might consist of a single N-PDU (SF). The service shall provide the currently available maximum buffer size when invoked with TpSduLength equal to 0.
PduR_FrArTpTxConfirmation	This function is called after the I-PDU has been transmitted on its network, the result indicates whether the transmission was successful or not.

] ()

8.6.2 Optional Interfaces

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

[SWS_FrArTp_00220] [

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

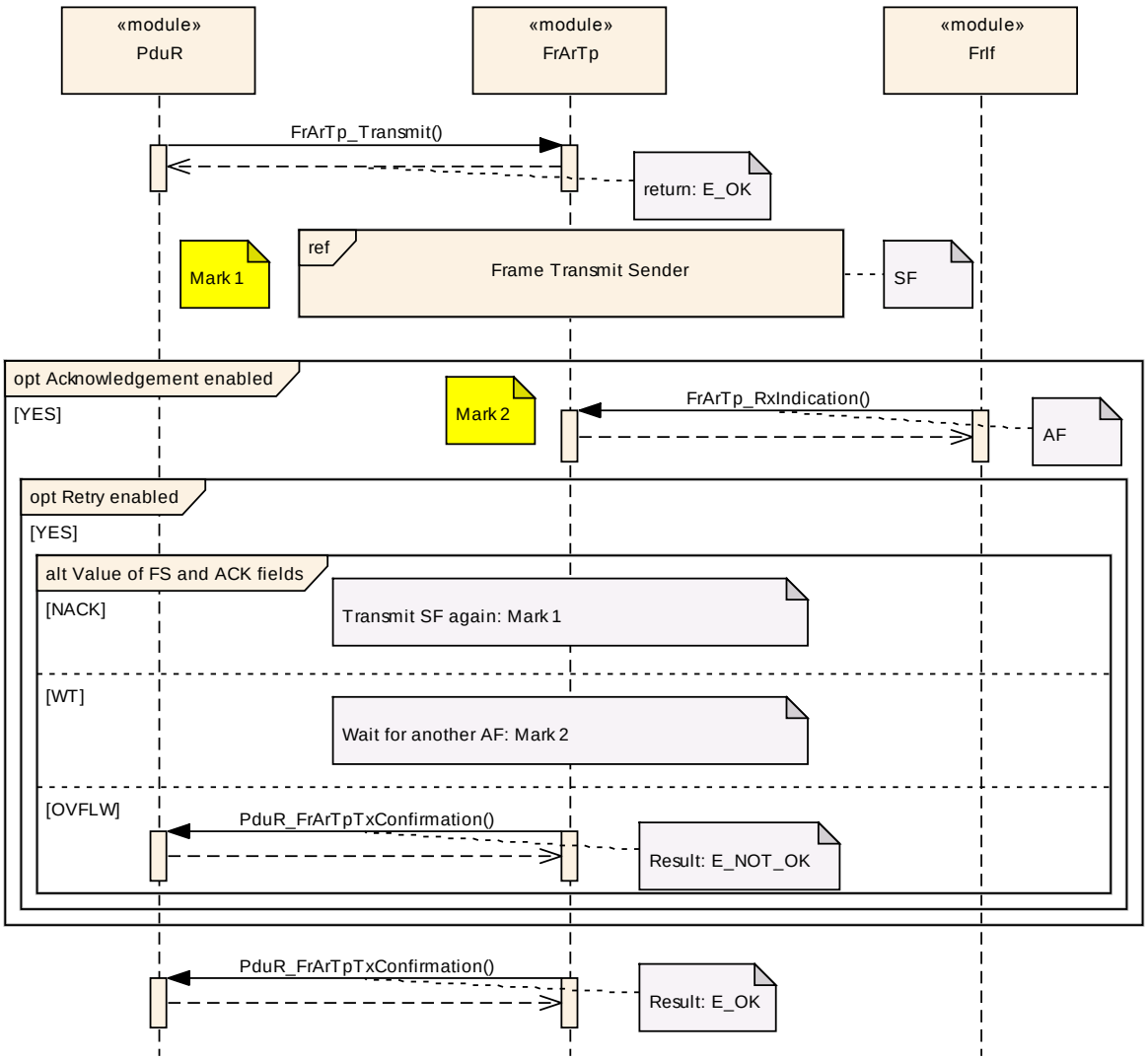
] ()

9 Sequence diagrams

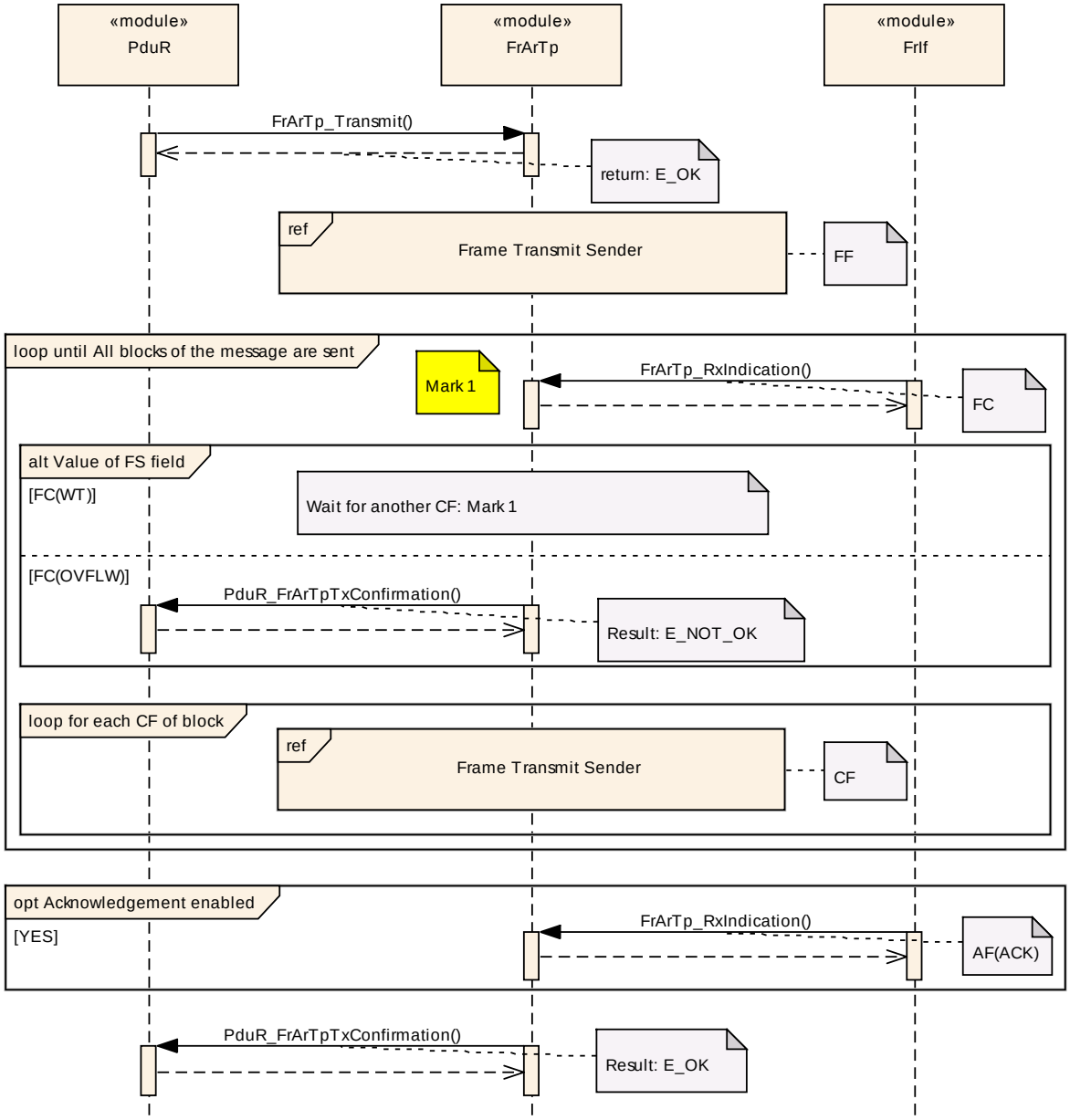
Although the following sequence diagrams are quite detailed, they do not depict every detail. Thus, they should be seen as an addendum to this specification.

9.1 N-SDU Transmission

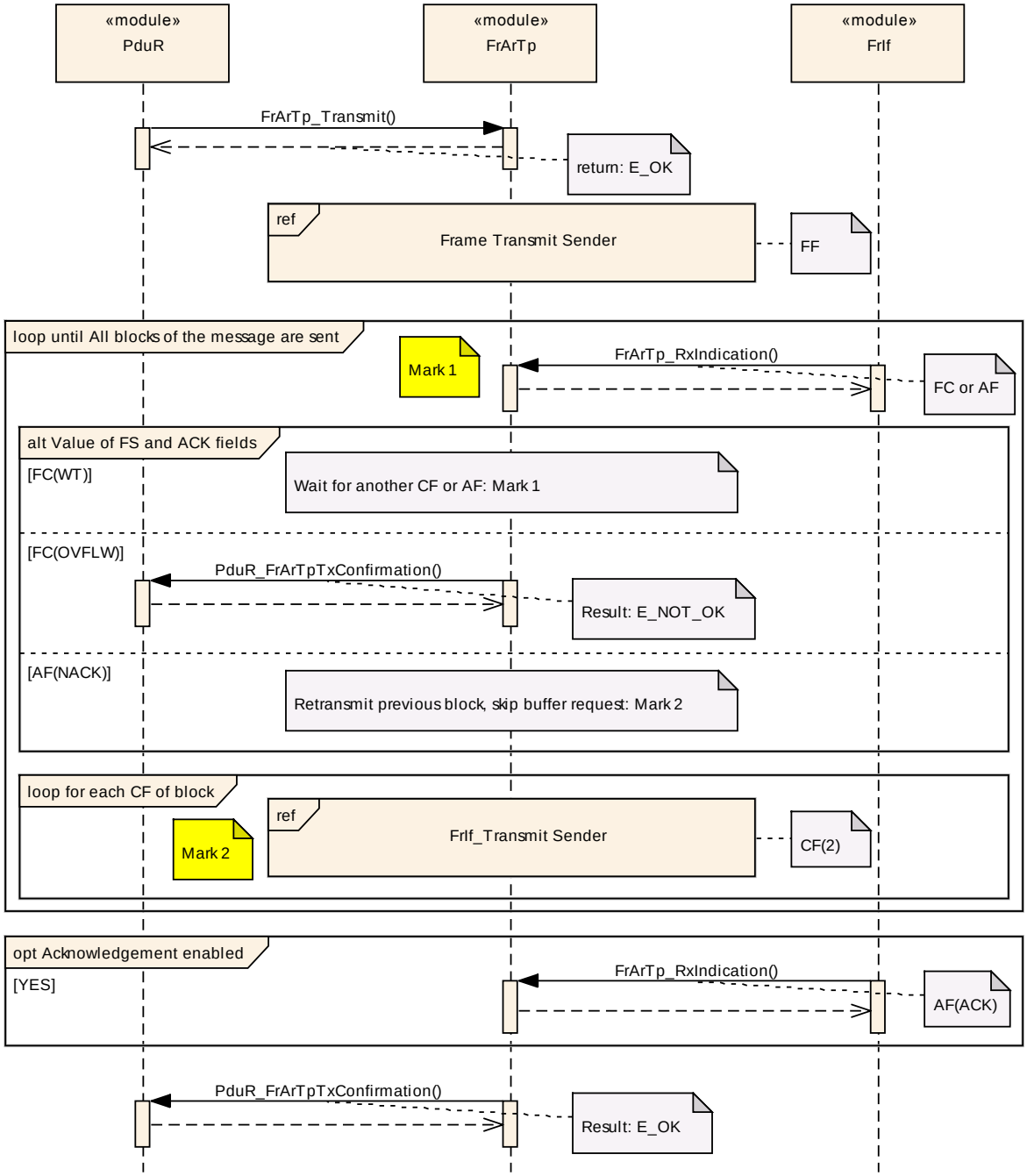
9.1.1 Unsegmented N-SDU Transmission



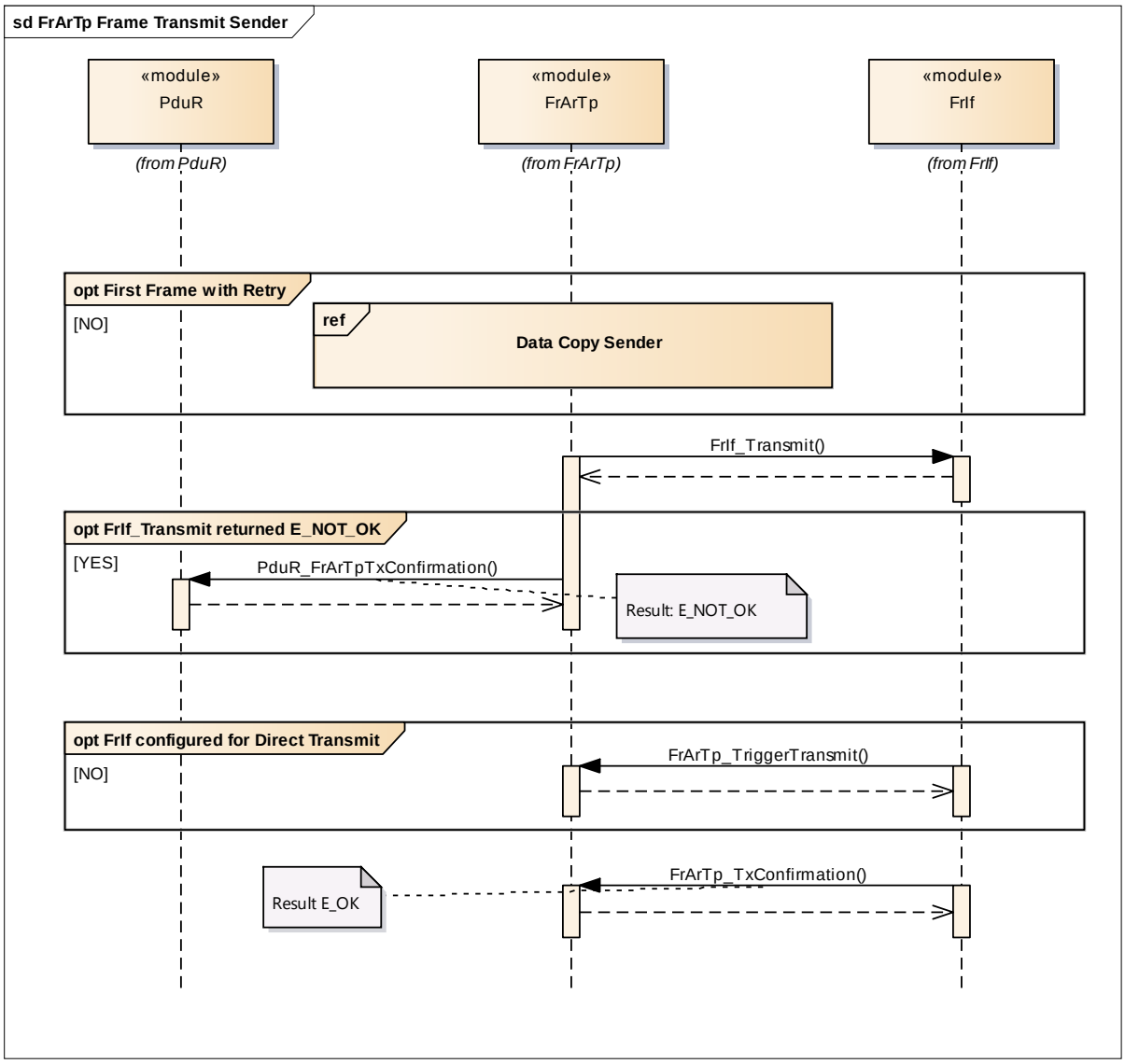
9.1.2 Segmented N-SDU Transmission without Retry



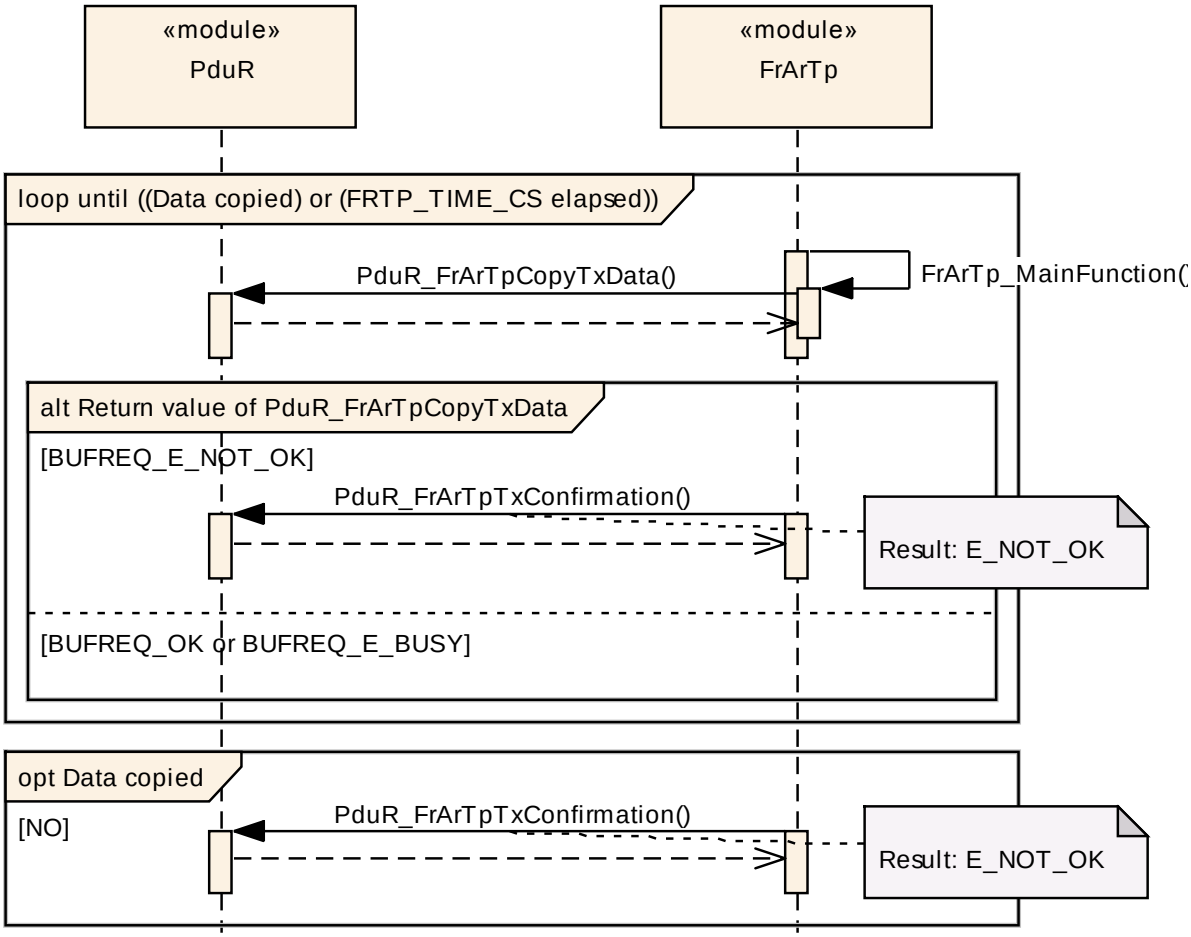
9.1.3 Segmented N-SDU Transmission with Retry


«module»
PduR
«module»
FrArTp
«module»
FrIf
FrArTp_Transmit()
return: E_OK
ref
Frame Transmit Sender
FF
loop until All blocks of the message are sent
Mark 1
FrArTp_RxIndication()
FC or AF
alt Value of FS and ACK fields
[FC(WT)]
Wait for another CF or AF: Mark 1
[FC(OVFLW)]
PduR_FrArTpTxConfirmation()
Result: E_NOT_OK
[AF(NACK)]
Retransmit previous block, skip buffer request: Mark 2
loop for each CF of block
ref
FrIf_Transmit Sender
Mark 2
CF(2)
opt Acknowledgement enabled
[YES]
FrArTp_RxIndication()
AF(ACK)
PduR_FrArTpTxConfirmation()
Result: E_OK

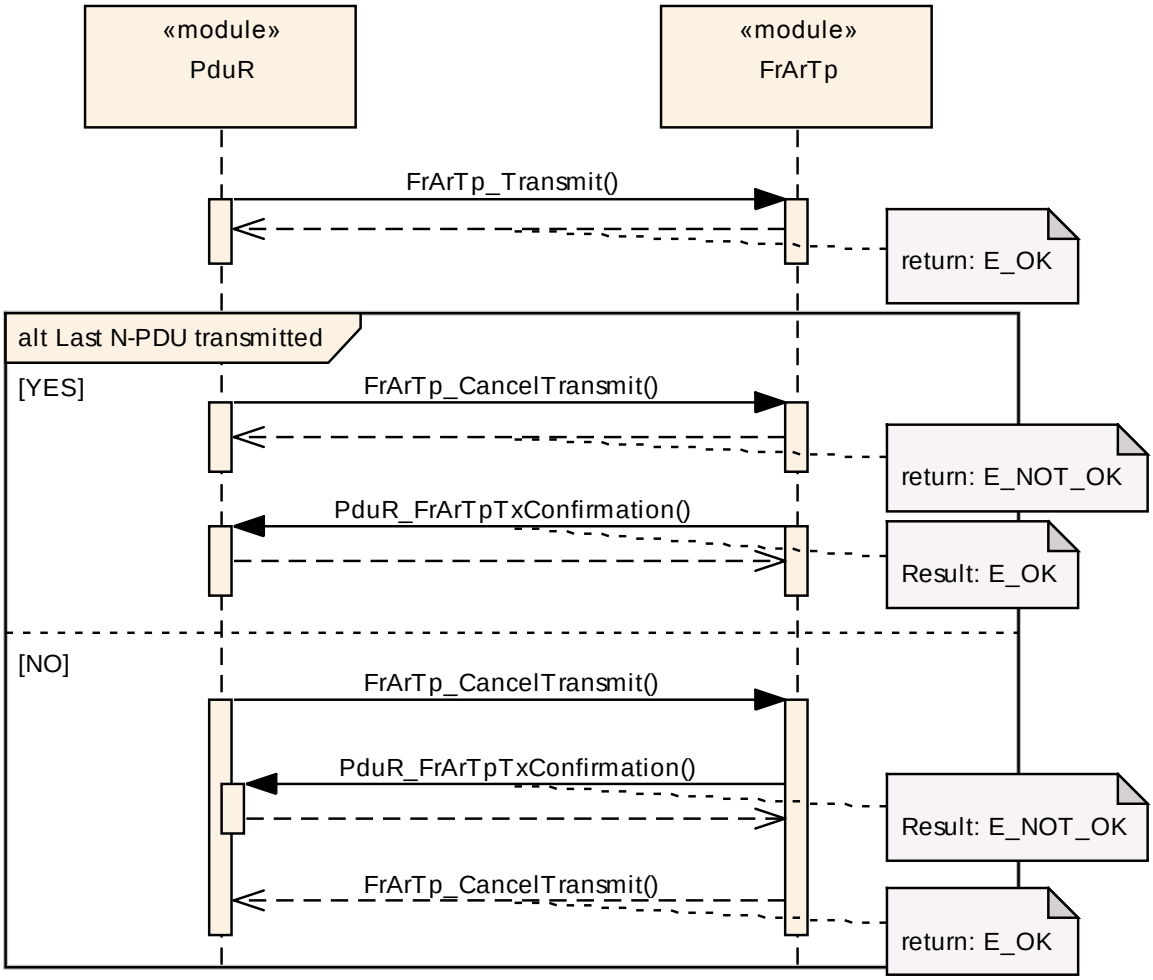
9.1.3.1 N-PDU Transmission during N-SDU Transmission



9.1.4 N-PDU Data Copying during N-SDU Transmission

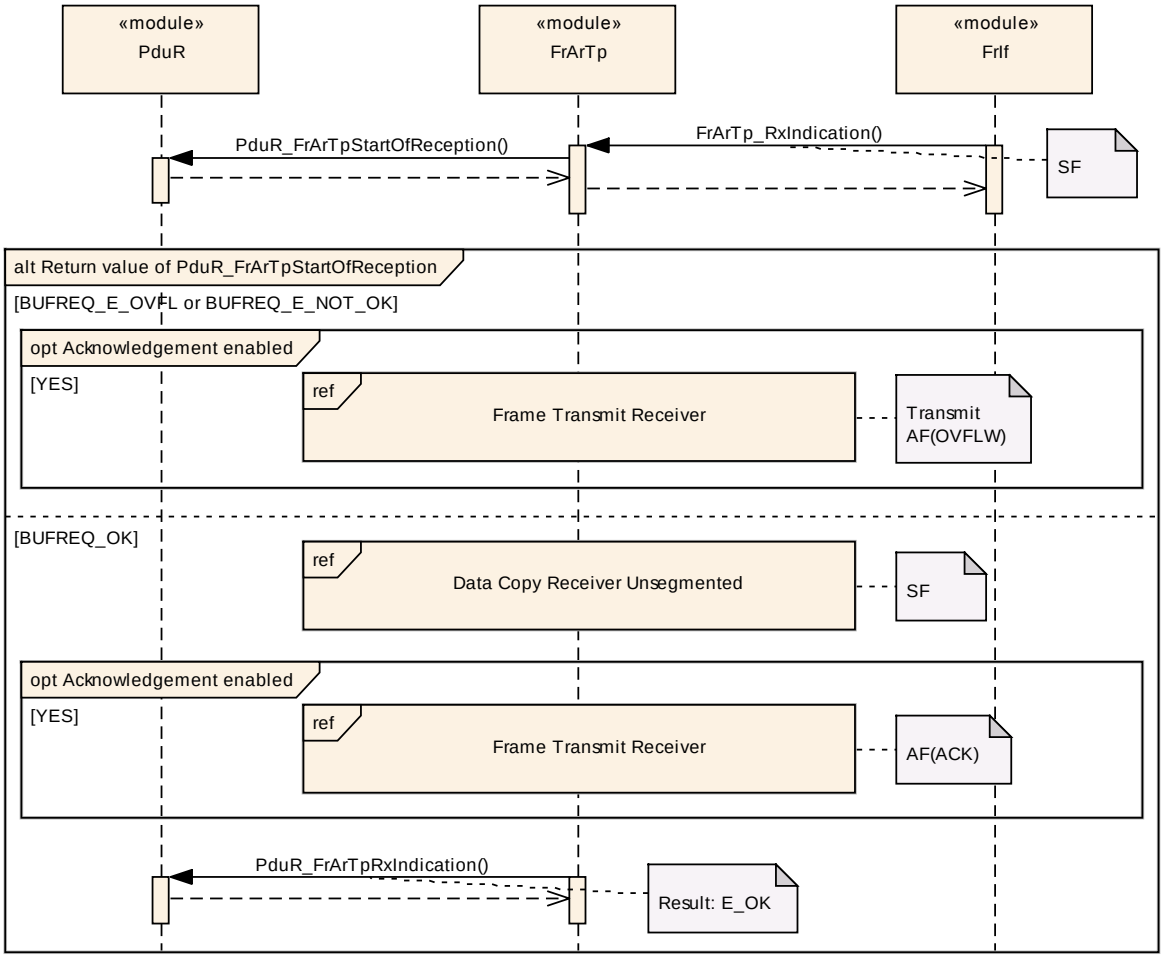


9.1.4.1 Transmit Cancellation

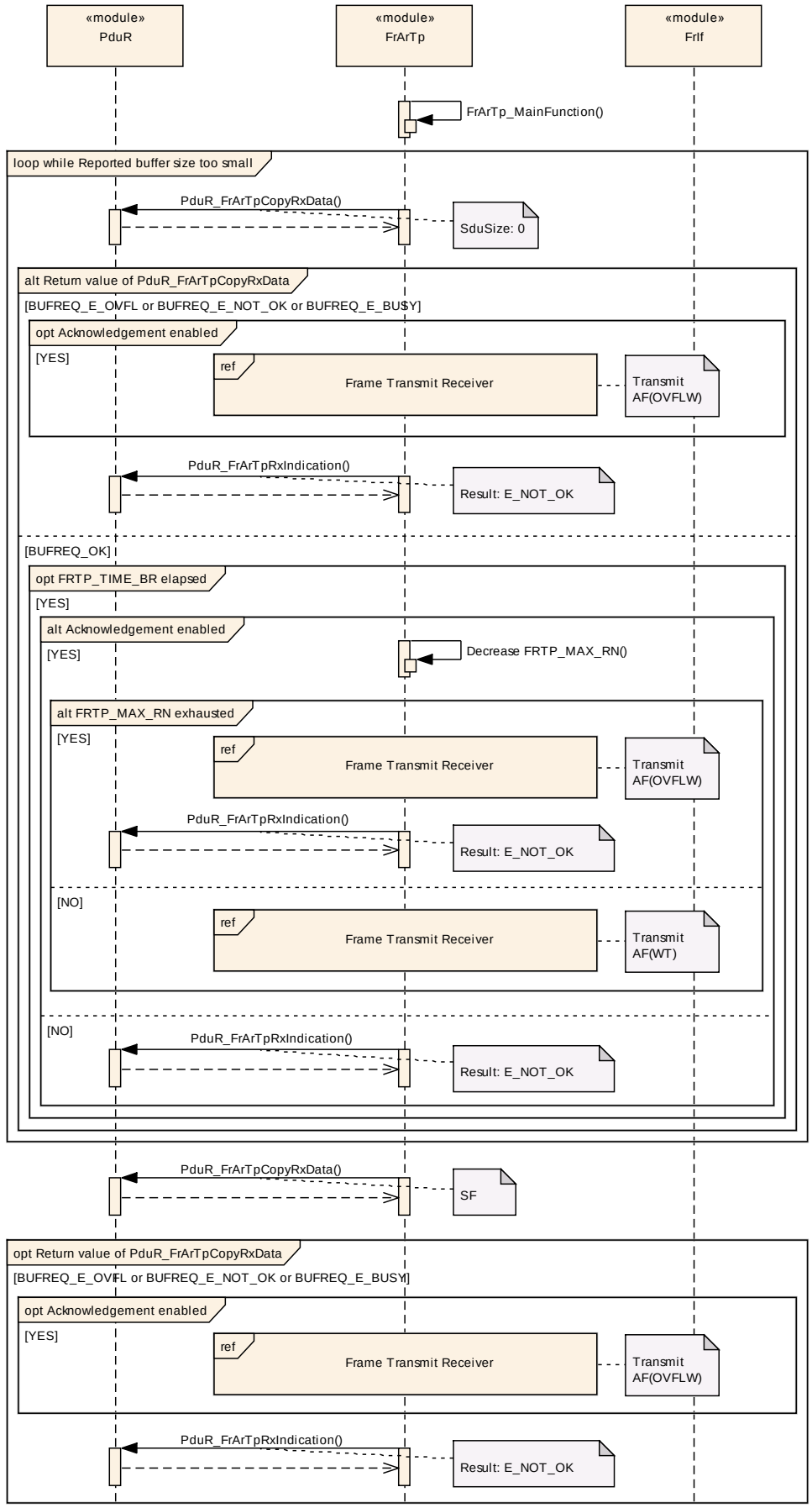


9.2 N-SDU Reception

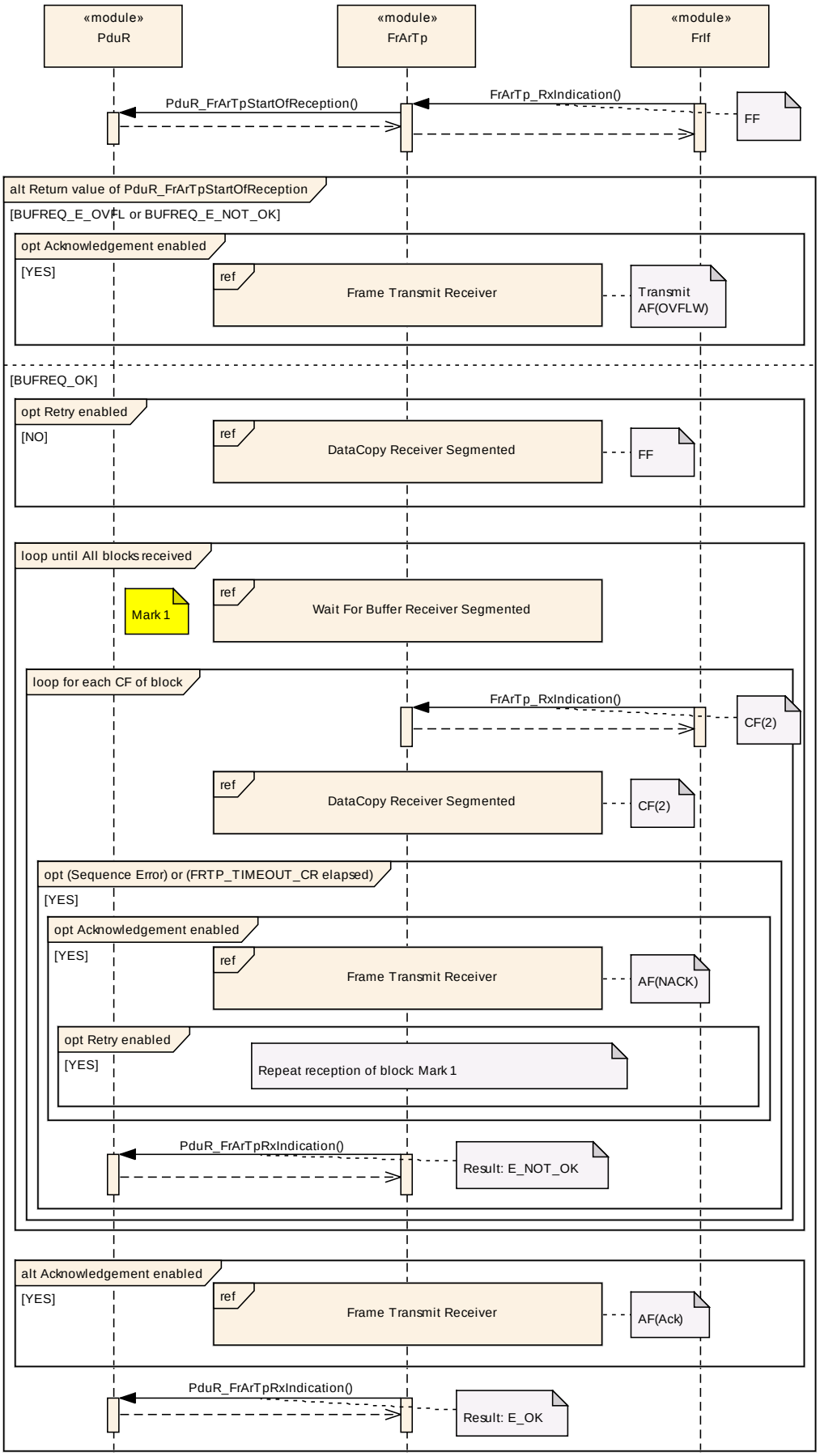
9.2.1 Unsegmented N-SDU Reception



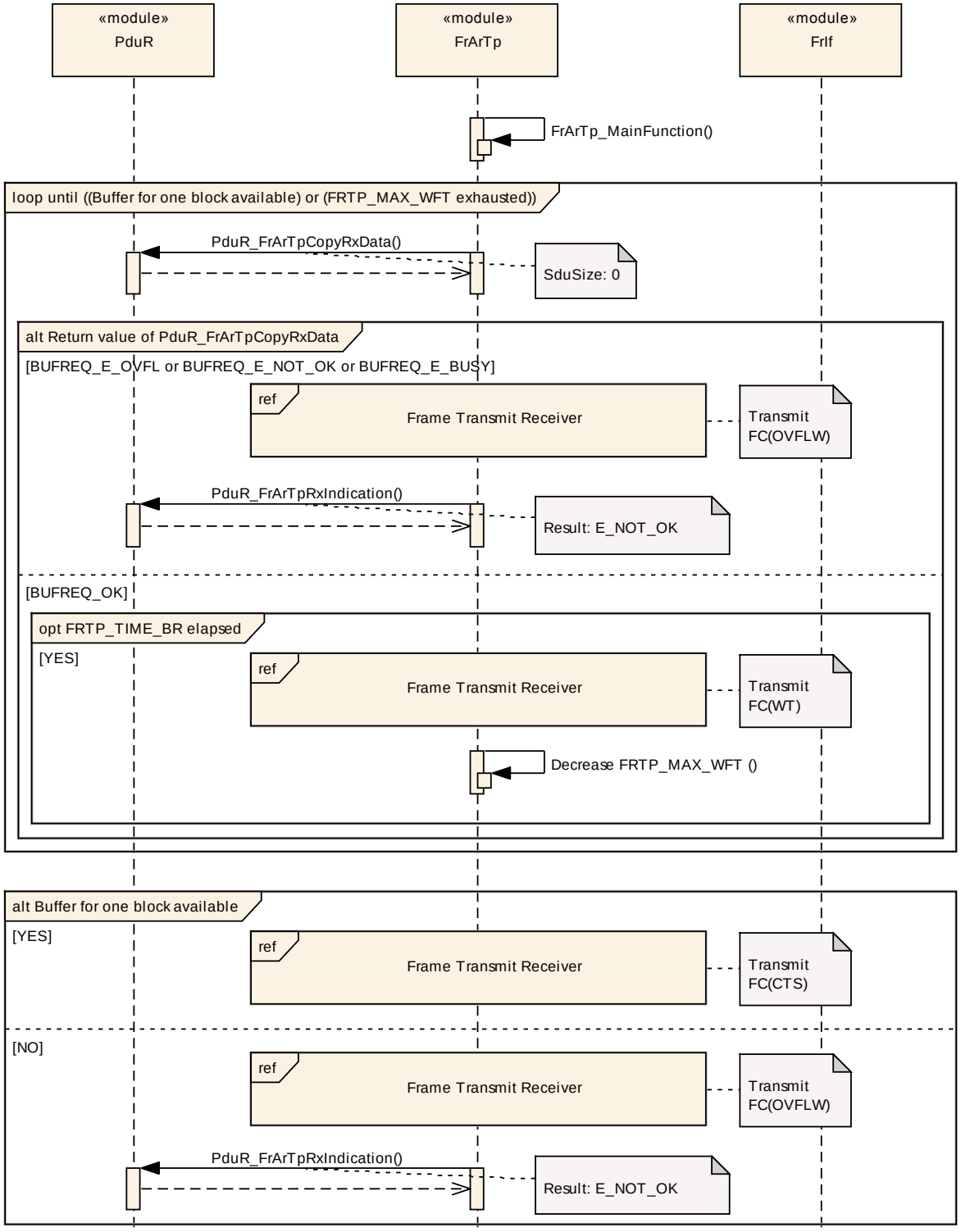
9.2.2 N-PDU Data Copying during Unsegmented N-SDU Reception


«module»
PduR
«module»
FrArTp
«module»
FrIf
FrArTp_MainFunction()
loop while Reported buffer size too small
PduR_FrArTpCopyRxData()
SduSize: 0
alt Return value of PduR_FrArTpCopyRxData
[BUFREQ_E_OVFL or BUFREQ_E_NOT_OK or BUFREQ_E_BUSY]
opt Acknowledgement enabled
[YES]
ref
Frame Transmit Receiver
Transmit
AF(OVFLW)
PduR_FrArTpRxIndication()
Result: E_NOT_OK
[BUFREQ_OK]
opt FRTP_TIME_BR elapsed
[YES]
alt Acknowledgement enabled
[YES]
Decrease FRTP_MAX_RN()
alt FRTP_MAX_RN exhausted
[YES]
ref
Frame Transmit Receiver
Transmit
AF(OVFLW)
PduR_FrArTpRxIndication()
Result: E_NOT_OK
[NO]
ref
Frame Transmit Receiver
Transmit
AF(WT)
[NO]
PduR_FrArTpRxIndication()
Result: E_NOT_OK
PduR_FrArTpCopyRxData()
SF
opt Return value of PduR_FrArTpCopyRxData
[BUFREQ_E_OVFL or BUFREQ_E_NOT_OK or BUFREQ_E_BUSY]
opt Acknowledgement enabled
[YES]
ref
Frame Transmit Receiver
Transmit
AF(OVFLW)
PduR_FrArTpRxIndication()
Result: E_NOT_OK

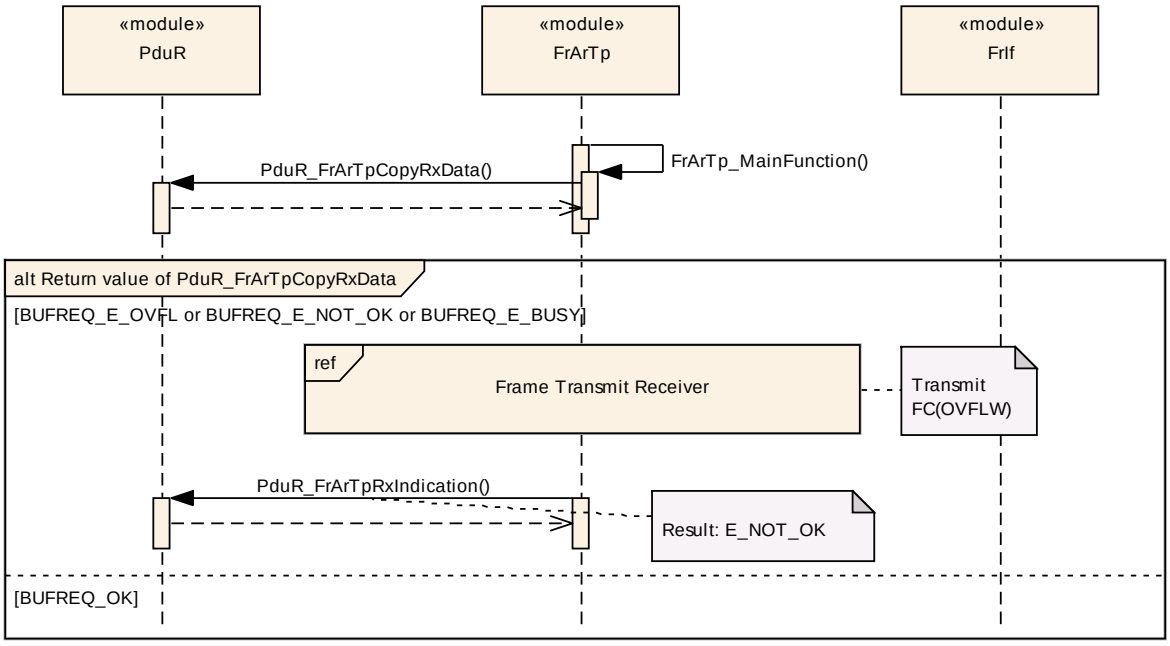
9.2.3 Segmented N-SDU Reception


«module»
PduR
«module»
FrArTp
«module»
FrIf
PduR_FrArTpStartOfReception()
FrArTp_RxIndication()
FF
alt Return value of PduR_FrArTpStartOfReception
[BUFREQ_E_OVFL or BUFREQ_E_NOT_OK]
opt Acknowledgement enabled
[YES]
ref
Frame Transmit Receiver
Transmit
AF(OVFLW)
[BUFREQ_OK]
opt Retry enabled
[NO]
ref
DataCopy Receiver Segmented
FF
loop until All blocks received
Mark 1
ref
Wait For Buffer Receiver Segmented
loop for each CF of block
FrArTp_RxIndication()
CF(2)
ref
DataCopy Receiver Segmented
CF(2)
opt (Sequence Error) or (FRTP_TIMEOUT_CR elapsed)
[YES]
opt Acknowledgement enabled
[YES]
ref
Frame Transmit Receiver
AF(NACK)
opt Retry enabled
[YES]
Repeat reception of block: Mark 1
PduR_FrArTpRxIndication()
Result: E_NOT_OK
alt Acknowledgement enabled
[YES]
ref
Frame Transmit Receiver
AF(Ack)
PduR_FrArTpRxIndication()
Result: E_OK

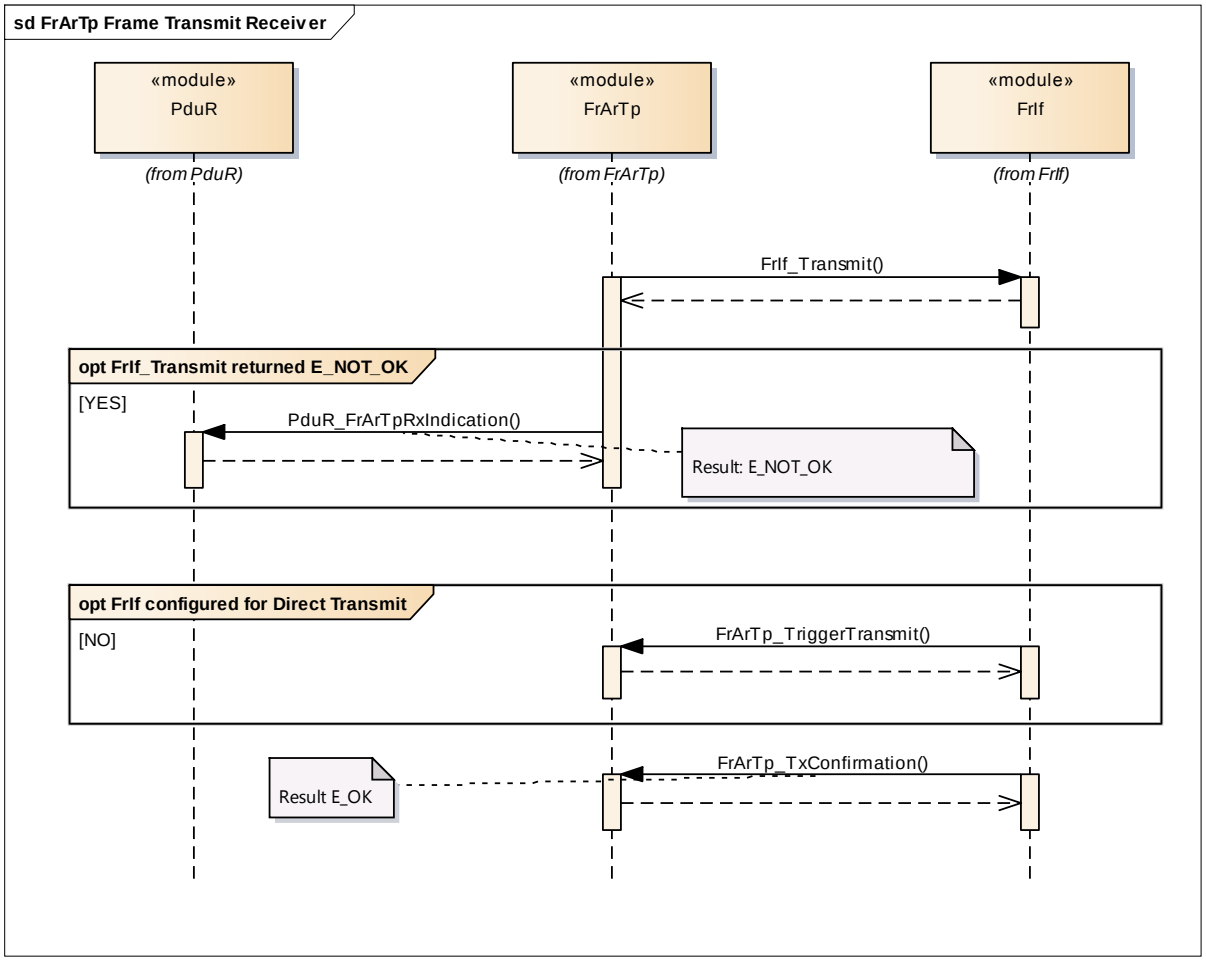
9.2.4 Wait for Buffer during Segmented N-SDU Reception



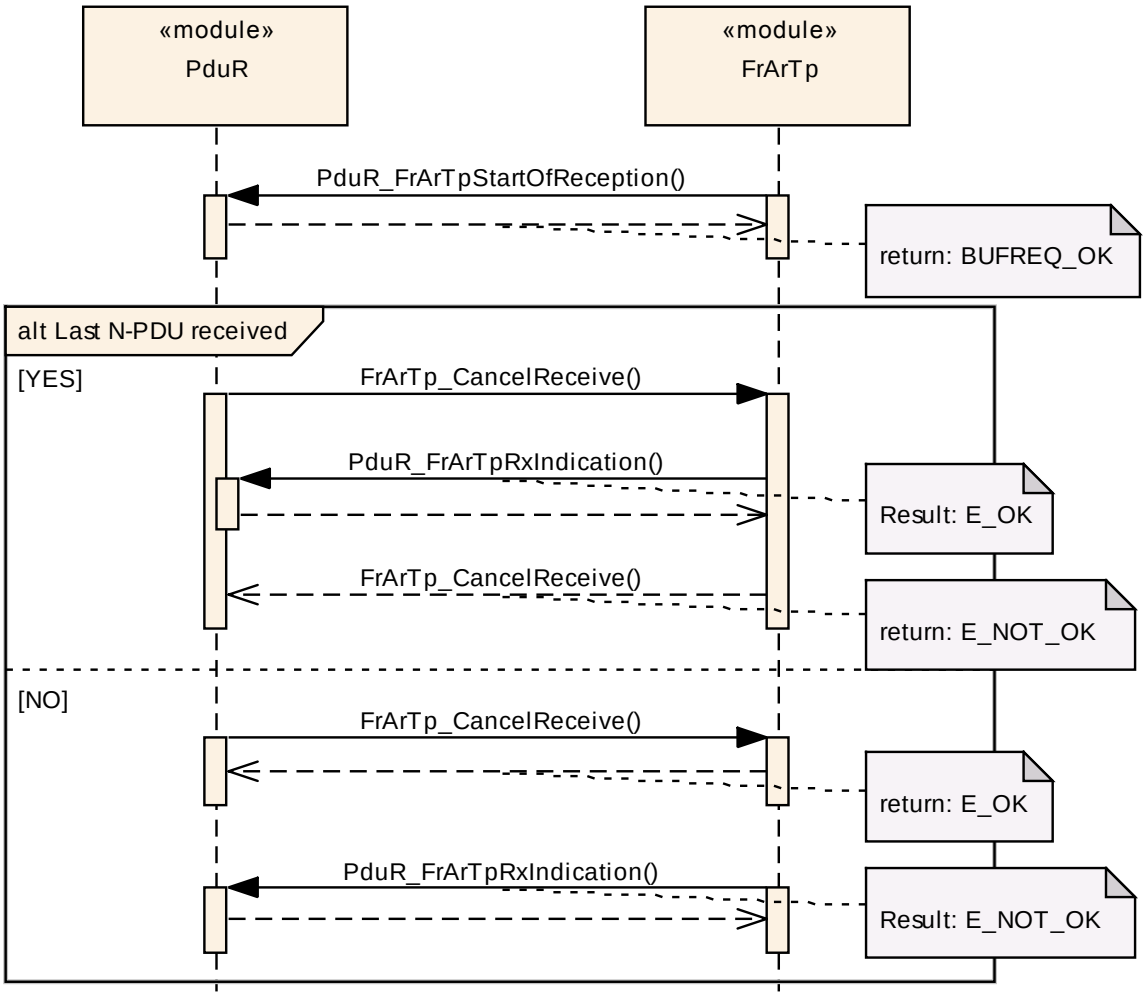
9.2.5 N-PDU Data Copying during Segmented N-SDU Reception



9.2.6 N-PDU Transmission during N-SDU Reception



9.2.7 Receive Cancellation



10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers.

Sections 10.1 and 10.3 refer to the corresponding sections in the SWS BSW General.

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

Section 10.4 specifies restrictions on some of the parameters of the FlexRay AUTOSAR Transport Layer module.

10.1 How to read this chapter

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

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 configuration:

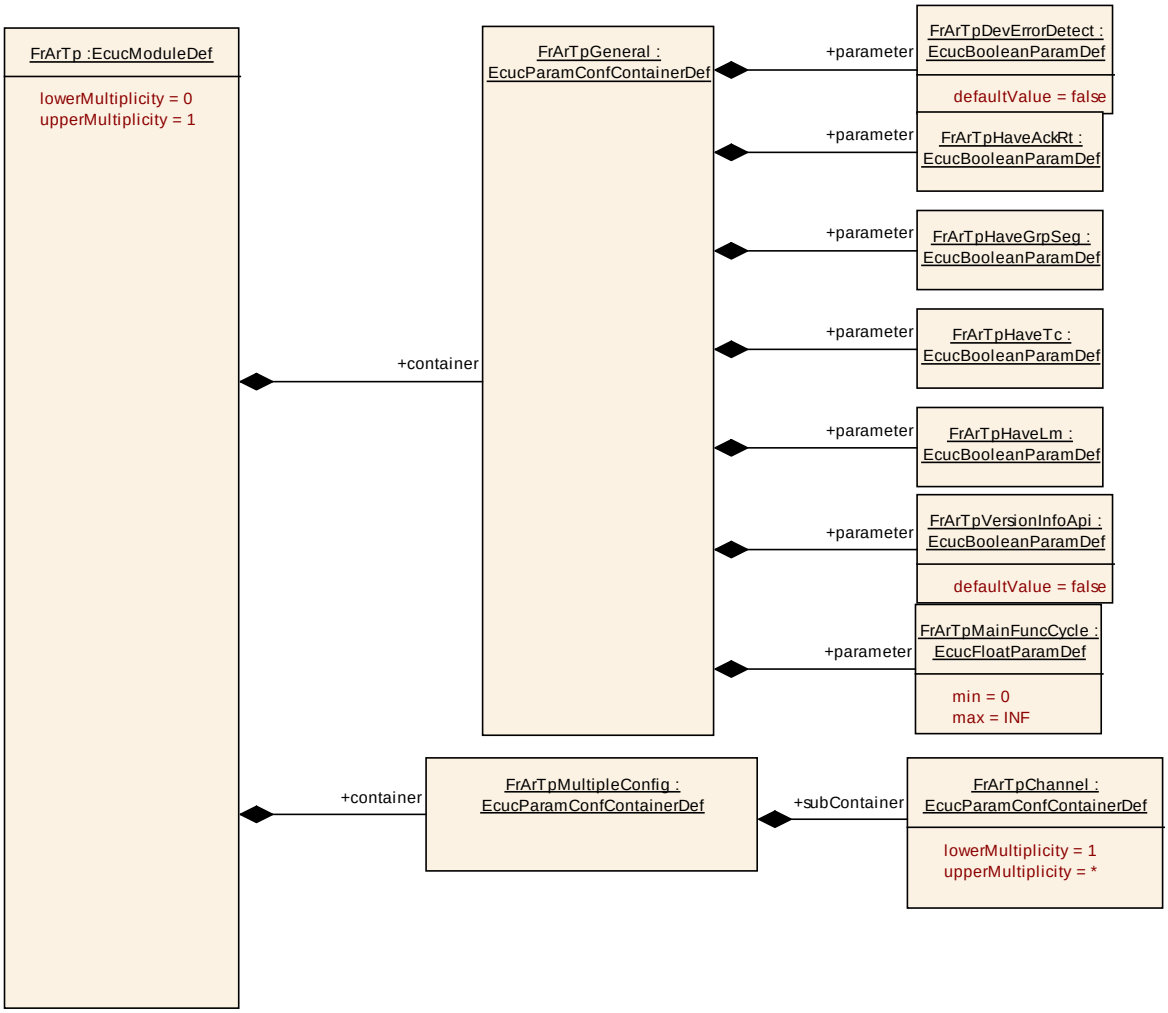


Figure 23: FrArTp main configuration

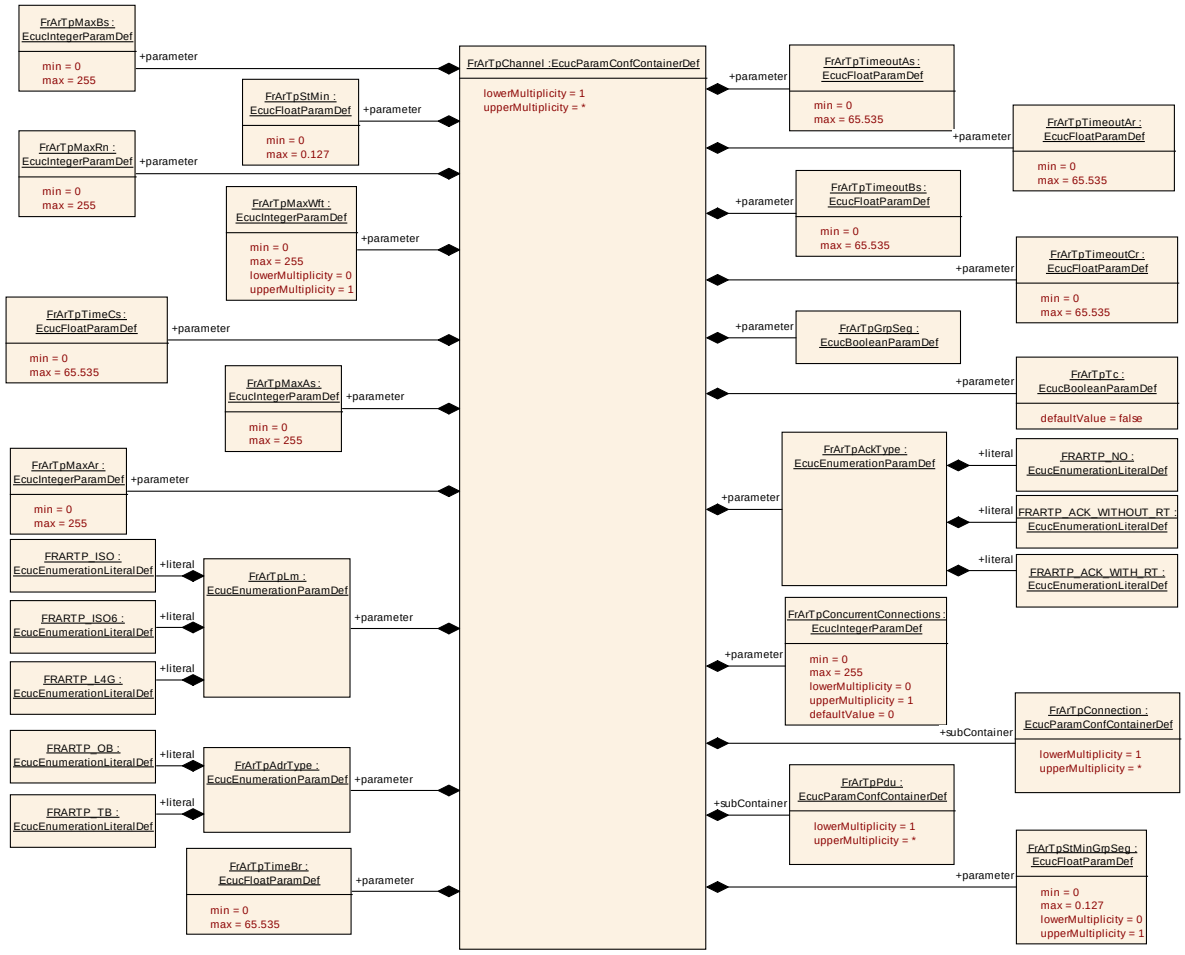


Figure 24: FrArTpChannel configuration

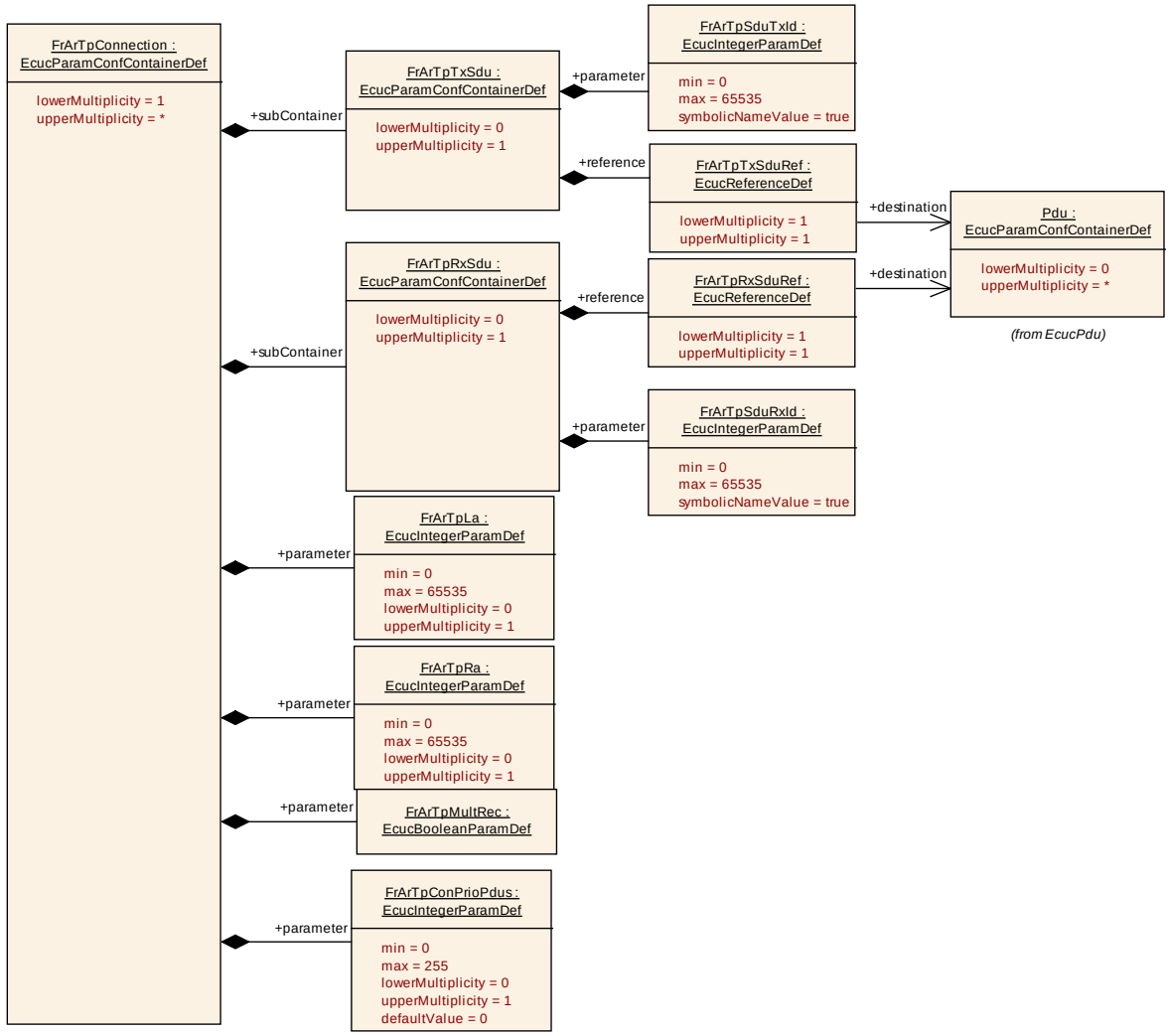


Figure 25: FrArTpConnection configuration

10.2.1 FrArTp

SWS Item	**ECUC_FrArTp_00001 :**
Module Name	*FrArTp*
Module Description	Configuration of the FrArTp (FlexRay Transport Protocol) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrArTpGeneral	1	This container contains the general configuration (parameters) of the FlexRay TP.
FrArTpMultipleConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR FrArTp module.

10.2.2 FrArTpGeneral

SWS Item	**ECUC_FrArTp_00012 :**
Container Name	FrArTpGeneral
Description	This container contains the general configuration (parameters) of the FlexRay TP.
Configuration Parameters	

SWS Item	**ECUC_FrArTp_00011 :**		
Name	FrArTpDevErrorDetect		
Parent Container	FrArTpGeneral		
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00014 :**		
Name	FrArTpHaveAckRt		
Parent Container	FrArTpGeneral		
Description	Preprocessor switch for enabling the Acknowledgement and retry mechanisms.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00015 :**		
Name	FrArTpHaveGrpSeg		
Parent Container	FrArTpGeneral		
Description	Preprocessor switch for enabling segmentation of 1:n messages.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00016 :**		
Name	FrArTpHaveLm		
Parent Container	FrArTpGeneral		
Description	Preprocessor switch for enabling the mechanism for message longer than allowed by.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	

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

SWS Item	**ECUC_FrArTp_00017 :**		
Name	FrArTpHaveTc		
Parent Container	FrArTpGeneral		
Description	Preprocessor switch for enabling Transmit Cancellation and Receive Cancellation.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00020 :**		
Name	FrArTpMainFuncCycle		
Parent Container	FrArTpGeneral		
Description	This parameter contains the calling period of the TPs Main Function. The parameter is specified in seconds.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_FrArTp_00054 :**		
Name	FrArTpVersionInfoApi		
Parent Container	FrArTpGeneral		
Description	Preprocessor switch for enabling the Version info API.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

All parameters within section 10.2.3 are global and, of course, only present once for the whole module.

10.2.3 FrArTpChannel

SWS Item	**ECUC_FrArTp_00005 :**
Container Name	FrArTpChannel
Description	This container contains the configuration (parameters) of one FlexRay TP

	channel.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_FrArTp_00002 :**		
Name	FrArTpAckType		
Parent Container	FrArTpChannel		
Description	This parameter defines the type of acknowledgement which is used for the specific channel.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRARTP_ACK_WITHOUT_RT	Acknowledgement without retry	
	FRARTP_ACK_WITH_RT	Acknowledgement with retry	
	FRARTP_NO	No acknowledgement	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00008 :**		
Name	FrArTpAdrType		
Parent Container	FrArTpChannel		
Description	This parameter states the addressing type this connection has. The meanings of the values are one byte and two byte.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRARTP_OB	One Byte	
	FRARTP_TB	Two Bytes	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00057 :**	
Name	FrArTpConcurrentConnections	
Parent Container	FrArTpChannel	
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	EcucIntegerParamDef	
Range	0 .. 255	
Default value	0	
Post-Build Variant Multiplicity	true	
Post-Build Variant Value	true	

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

SWS Item	**ECUC_FrArTp_00013 :**		
Name	FrArTpGrpSeg		
Parent Container	FrArTpChannel		
Description	Here can be specified, whether segmentation within a 1:n connection is allowed or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00019 :**		
Name	FrArTpLm		
Parent Container	FrArTpChannel		
Description	This specifies the maximum message length for the particular channel.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRARTP_ISO		Up to (2**12)-1 Byte message length (No FF-Ex or SF-E or AF shall be used and recognized)
	FRARTP_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.
	FRARTP_L4G		SF-E allowed (SF of arbitrary length depending on FrArTpPduLength), up to (2**32)-1 byte message length (all FF-x allowed).
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00021 :**	
Name	FrArTpMaxAr	
Parent Container	FrArTpChannel	
Description	This parameter defines the maximum number of trying to send a frame when a TIMEOUT AR occurs.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	

Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00022 :**		
Name	FrArTpMaxAs		
Parent Container	FrArTpChannel		
Description	This parameter defines the maximum number of trying to send a frame when a TIMEOUT AS occurs.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00023 :**		
Name	FrArTpMaxBs		
Parent Container	FrArTpChannel		
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	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00026 :**		
Name	FrArTpMaxRn		
Parent Container	FrArTpChannel		
Description	This parameter defines the maximum number of retries (if retry is configured for the particular channel).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00059 :**		
Name	FrArTpMaxWft		
Parent Container	FrArTpChannel		

Description	This parameter defines the maximal number of wait frames to be sent for a pending connection.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00042 :**		
Name	FrArTpStMin		
Parent Container	FrArTpChannel		
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 FrArTp_ChangeParameter interface. FrArTpStMin must be an integer multiple of the cycle length multiplied with the multiplexing factor, i.e. FrArTpStMin = 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 FrArTp, FrArTpStMin 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	EcucFloatParamDef		
Range	[0 .. 0.127]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00060 :**
Name	FrArTpStMinGrpSeg
Parent Container	FrArTpChannel
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 FrArTp_ChangeParameter interface. FrArTpStMinGrpSeg must be an integer multiple of the cycle length multiplied with the multiplexing factor, i.e. FrArTpStMinGrpSeg = 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 FrArTp, FrArTpStMinGrpSeg 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	EcucFloatParamDef		
Range	[0 .. 0.127]		
Default value	--		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00043 :**		
Name	FrArTpTc		
Parent Container	FrArTpChannel		
Description	With this switch Transmit Cancellation and Receive Cancellation can be turned on or off for this channel.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00044 :**		
Name	FrArTpTimeBr		
Parent Container	FrArTpChannel		
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 FRARTP_TIME_BR + (FRARTP_TIMEOUT_AR * FRARTP_MAX_AR) < FRARTP_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	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00046 :**
Name	FrArTpTimeCs
Parent Container	FrArTpChannel
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 FRARTP_TIME_CS + (FRARTP_TIMEOUT_AS * FRARTP_MAX_AS) < FRARTP_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	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00048 :**		
Name	FrArTpTimeoutAr		
Parent Container	FrArTpChannel		
Description	This parameter states the timeout in seconds between the PDU transmit request of the Transport Layer to the FlexRay Interface and the corresponding confirmation of the FlexRay Interface on the receiver side (for FC or AF).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00049 :**		
Name	FrArTpTimeoutAs		
Parent Container	FrArTpChannel		
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	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00050 :**
Name	FrArTpTimeoutBs
Parent Container	FrArTpChannel
Description	This parameter defines the timeout in seconds for waiting for an FC or AF

	on the sender side in a 1:1 connection.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00051 :**		
Name	FrArTpTimeoutCr		
Parent Container	FrArTpChannel		
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	EcucFloatParamDef		
Range	[0 .. 65.535]		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrArTpConnection	1..*	This container contains the configuration (parameters) of one FlexRay TP connection. A connection can only belong to one channel.
FrArTpPdu	1..*	Container to hold the PDU parameters. ImplementationType: PduInfoType

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 [13]
The two parameters, FrArTpTimeBr and FrArTpTimeCs, are **not software configuration parameters**, they are contained in [13] as performance requirements. They are just for information.

10.2.4 FrArTpPdu

SWS Item	**ECUC_FrArTp_00029 :**		
Container Name	FrArTpPdu		
Description	Container to hold the PDU parameters. ImplementationType: PduInfoType		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_FrArTp_00030 :**		
Name	FrArTpPduDirection		
Parent Container	FrArTpPdu		
Description	This parameter defines the direction of the PDU.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRARTP_RX	Received PDU	
	FRARTP_TX	Transmitted PDU	
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00035 :**		
Name	FrArTpPduId		
Parent Container	FrArTpPdu		
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. For FrArTpPduDirection == FRARTP_RX, this parameter specifies the ID that is used by FrIf when calling FrArTp_RxIndication, while for FrArTpPduDirection == FRARTP_TX this ID is used by FrIf when calling FrArTp_TxConfirmation or FrArTp_TriggerTransmit. ImplementationType: PduIdType		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_FrArTp_00036 :**		
Name	FrArTpPduRef		
Parent Container	FrArTpPdu		
Description	--		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.5 FrArTpConnection

SWS Item	**ECUC_FrArTp_00010 :**		
Container Name	FrArTpConnection		
Description	This container contains the configuration (parameters) of one FlexRay TP connection. A connection can only belong to one channel.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_FrArTp_00058 :**		
Name	FrArTpConPrioPdus		
Parent Container	FrArTpConnection		
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	EcucIntegerParamDef		
Range	0 .. 255		
Default value	0		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00018 :**		
Name	FrArTpLa		
Parent Container	FrArTpConnection		
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. If this parameter is not configured, all related Rx N-SDUs must be configured to use the meta data item TARGET_ADDRESS_16, and all related Tx-N-SDUs must be configured to use the meta data item SOURCE_ADDRESS_16.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE

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

SWS Item	**ECUC_FrArTp_00027 :**		
Name	FrArTpMultRec		
Parent Container	FrArTpConnection		
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	EcucBooleanParamDef		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrArTp_00037 :**		
Name	FrArTpRa		
Parent Container	FrArTpConnection		
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. If this parameter is not configured, all related Rx N-SDUs must be configured to use the meta data item SOURCE_ADDRESS_16, and all related Tx-N-SDUs must be configured to use the meta data item TARGET_ADDRESS_16.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrArTpRxSdu	0..1	Describes the Rx N-SDU. This N-SDU can produce meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.
FrArTpTxSdu	0..1	Describes the Tx N-SDU. This N-SDU can consume meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.

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.

10.2.6 FrArTpTxSdu

SWS Item	**ECUC_FrArTp_00055 :**		
Container Name	FrArTpTxSdu		
Description	Describes the Tx N-SDU. This N-SDU can consume meta data items of type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_FrArTp_00041 :**		
Name	FrArTpSduTxId		
Parent Container	FrArTpTxSdu		
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	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_FrArTp_00052 :**		
Name	FrArTpTxSduRef		
Parent Container	FrArTpTxSdu		
Description	Reference to a PDU in the global PDU structure.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.7 FrArTpRxSdu

SWS Item	**ECUC_FrArTp_00038 :**
Container Name	FrArTpRxSdu
Description	Describes the Rx N-SDU. This N-SDU can produce meta data items of

	type SOURCE_ADDRESS_16 and TARGET_ADDRESS_16.		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

SWS Item	**ECUC_FrArTp_00040 :**		
Name	FrArTpSduRxId		
Parent Container	FrArTpRxSdu		
Description	This is a unique identifier for a received message. This Id is used in the CancelReceive and ChangeParameter API call. ImplementationType: PduIdType		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 65535		
Default value	--		
Post-Build Variant Value	false		
Value Configuration Class	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_FrArTp_00039 :**		
Name	FrArTpRxSduRef		
Parent Container	FrArTpRxSdu		
Description	Reference to a PDU in the global PDU structure.		
Multiplicity	1		
Type	Reference to [Pdu]		
Post-Build Variant Value	true		
Value Configuration Class	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

No Included Containers

10.2.8 FrArTpMultipleConfig

SWS Item	**ECUC_FrArTp_00028 :**
Container Name	FrArTpMultipleConfig
Description	This container contains the configuration parameters and sub containers of the AUTOSAR FrArTp module.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrArTpChannel	1..*	This container contains the configuration (parameters) of one FlexRay TP channel.

10.3 Published Information

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

10.4 Important Issues on Configuration

10.4.1 Start and Stop of the Timing Parameters

[SWS_FrArTp_00169] ⌈ 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).⌋ ()

[SWS_FrArTp_00170] ⌈ For 1:n connections only the parameters FrArTpTimeoutAs, FrArTpTimeCs (only CF) and FrArTpTimeoutCr (only CF) hold, since no flow control or acknowledgement is allowed in that case.⌋ ()

Timing Parameter	*Start*	*Stop*
FrArTpTimeoutAs	*FrIf_Transmit* (first PDU of the group used by the current connection)	*FrArTp_TxConfirmation* (for the last PDU of the group used by the current connection)
FrArTpTimeoutAr	*FrIf_Transmit* (FC or AF)	*FrArTp_TxConfirmation* (FC or AF)
FrArTpTimeoutBs	*FrArTp_TxConfirmation* (SF-x, FF-x or last CF of a block), *FrArTp_RxIndication* (FC or AF, both in case of FR_FS = WAIT)	*FrArTp_RxIndication* (FC or AF)
FrArTpTimeBr	*FrArTp_RxIndication* (FF-x, last CF of a block or SF-x), *FrArTp_TxConfirmation (FC or AF, both in case of FR_FS = WAIT)*	*FrIf_Transmit* (FC or AF)
FrArTpTimeoutCr	*FrArTp_RxIndication* (CF), *FrArTp_TxConfirmation* (FC or AF)	*FrArTp_RxIndication* (CF or SF-x, FF-x (the latter two in case of retry))
FrArTpTimeCs	*FrArTp_TxConfirmation* (CF), *FrArTp_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

[SWS_FrArTp_00171] ⌈ 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).⌋ ()

These and those are explained in the table below:

Parameter	*Allowed values*
FrArTpAckType （*）	‘FRARTP_NO’
FrArTpGrpSeg （*）	false

FrArTpTc （*）	false
FrArTpLm （*）	‘FRARTP_ISO’, ‘FRARTP_ISO6’
N-PDU length	9 [FrArTpAdrType == FRARTP_OB, FrArTpLm == FRARTP_ISO6], 10 [FrArTpAdrType == FRARTP_OB, FrArTpLm == FRARTP_ISO], 11 [FrArTpAdrType == FRARTP_TB, FrArTpLm == FRARTP_ISO6], 12 [FrArTpAdrType == FRARTP_TB, FrArTpLm == FRARTP_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

[SWS_FrArTp_00172] ⌈ There are several dependencies among the connection specific and channel specific configuration parameters:⌋ ()

- If FrArTpMultRec sets the connection to be a 1:1 connection, then the value of FrArTpGrpSeg does not play a role for this connection since it is only relevant for 1:n connections.
- If FrArTpMultRec sets the connection to be a 1:n connection, then the values of FrArTpAckType, FrArTpMaxBs, and FrArTpMaxRn do not play a role for this connection, since they are only relevant for 1:1 connections.
- If FrArTpMultRec sets the connection to be a 1:n connection or FrArTpAckType does not activate retry (FRARTP_NO, FRARTP_ACK_WITHOUT_RT) then the value of FrArTpMaxBs 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:

[SWS_FrArTp_00242] ⌈ V_E + FrArTpTimeBr + (FrArTpTimeoutAr * FrArTpMaxAr) + V_S < FrArTpTimeoutBs⌋ ()

[SWS_FrArTp_00243] ⌈ V_S + FrArTpTimeCs + (FrArTpTimeoutAs * FrArTpMaxAs)+ V_E < FrArTpTimeoutCr⌋ ()

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

[SWS_FrArTp_00180] ⌈ It has to be assured, that FrArTpStMin < FrArTpTimeoutCr since there will always be a timeout of the latter one otherwise.⌋ (SRS_BSW_00159)

[SWS_FrArTp_00181] ⌈ 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 FrArTpLa and FrArTpRa are swapped.⌋ (SRS_BSW_00159)

[SWS_FrArTp_00275] ⌈ 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.⌋ ()

[SWS_FrArTp_00276] ⌈ All PDUs of a PDU pool must have the same size, and all channels that reference these PDUs must have the same addressing type.⌋ ()

[SWS_FrArTp_00277] ⌈ 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.⌋ ()

[SWS_FrArTp_00278] ⌈ 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

[SWS_FrArTp_00174] ⌈ 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.

[SWS_FrArTp_00175] ⌈ For every FrArTp 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 FrArTp for in the current transfer not used N-PDUs or if e.g. in every 2nd FlexRay bus cycle an N-PDU is scheduled.

[SWS_FrArTp_00182] ⌈ For the last PDU (in temporal order) of each PDU pool, a TxConfirmation shall be configured.⌋ ()

11 Not applicable requirements

[SWS_FrArTp_00999] ⌈ These requirements are not applicable to this specification.⌋ (BSW00161, BSW00162, BSW00172, SRS_BSW_00301, SRS_BSW_00302, SRS_BSW_00307, SRS_BSW_00321, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00334, SRS_BSW_00335, SRS_BSW_00339, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00344, SRS_BSW_00347, SRS_BSW_00348, SRS_BSW_00375, SRS_BSW_00395, SRS_BSW_00400, SRS_BSW_00405, SRS_BSW_00409, SRS_BSW_00412, SRS_BSW_00415, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, BSW00431, SRS_BSW_00433, BSW00434, SRS_BSW_00005, SRS_BSW_00010, SRS_BSW_00164, SRS_BSW_00168, and SRS_BSW_00170)