

Document Title	Specification of FlexRay Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	27

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R25-11

Document Change History			
Date	Release	Changed by	Description
2025-11-27	R25-11	AUTOSAR Release Management	No content changes
2024-11-27	R24-11	AUTOSAR Release Management	Replaced upper communications layers by LSduR
2023-11-23	R23-11	AUTOSAR Release Management	No content changes
2022-11-24	R22-11	AUTOSAR Release Management	No content changes
2021-11-25	R21-11	AUTOSAR Release Management	Clarification on shortening of L-SduLength
2020-11-30	R20-11	AUTOSAR Release Management	Editorial changes
2019-11-28	R19-11	AUTOSAR Release Management	- Clarification on handling of dynamic length LSdus - Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	- Added bus mirroring support - Changed behavior for TxConflict - Minor corrections





2017-12-08	4.3.1	AUTOSAR Release Management	- Runtime error rollout - UL_TxConfirmation replaced with UL_TriggerTransmit in affected requirements
2016-11-30	4.3.0	AUTOSAR Release Management	- New feature to get the "TxConflictState" - Introduce reliable TxConfirmation - Unused bit handling reworked - Several bug fixes
2015-07-31	4.2.2	AUTOSAR Release Management	- Minor corrections - Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	- Support for GlobalTimeSynchronization added - Minor corrections
2014-03-31	4.1.3	AUTOSAR Release Management	- Added Chapter for Production Errors - Editorial Changes
2013-10-31	4.1.2	AUTOSAR Release Management	- Minor corrections - Editorial changes - Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Release Management	- Traceability requirements added - Several Bug fixes - Editorial Changes
2011-12-22	4.0.3	AUTOSAR Release Management	Added User-defined communication operations
2010-09-30	3.1.5	AUTOSAR Release Management	- API "FrIf_GetCycleLength" added - API "FrIf_ReadCCConfig" added - APIs FrIf_EnableTransceiverWakeup / FrIf_DisableTransceiverWakeup removed - Configuration parameter "FrIfByteOrder" added





2010-02-02	3.1.4	AUTOSAR Release Management	• Added support for FlexRay 3.0 hardware (CCs and transceivers) • Added functionalities to get detailed (error) information of the communications bus • Added support for single/key-slot mode • Added “cancel transmission” support • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Release Management	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Release Management	• Correction of Figure 5.1
2007-12-21	3.0.1	AUTOSAR Release Management	• Simplification of the FlexRay Interface State Machine due to the introduction of the new AUTOSAR SWS FlexRay State Manager • Cluster-based APIs were removed due to the introduced AUTOSAR SWS FlexRay State Manager • The FlexRay Interface does not initialize any other modules any more due to the introduction of the “flat initialization” for AUTOSAR release 3.0 Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Release Management	• “Advice” for users revised • Legal disclaimer added • “Revision Information” added • Release Notes added
2006-05-16	2.0	AUTOSAR Release Management	• Second Release
2005-05-31	1.0	AUTOSAR Release Management	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1	Introduction and functional overview	9
1.1	General Hints	10
2	Acronyms and Abbreviations	11
3	Related documentation	12
3.1	Input documents & related standards and norms	12
3.2	Related specification	12
4	Constraints and assumptions	13
4.1	Limitations	13
4.2	Applicability to car domains	13
5	Dependencies to other modules	14
5.1	AUTOSAR Operating System	14
5.2	All Upper Layer AUTOSAR BSW Modules	14
5.3	AUTOSAR LSdu-Router	14
5.4	AUTOSAR FlexRay Network Management	15
5.5	AUTOSAR FlexRay Transport Protocol	15
5.6	AUTOSAR Bus Mirroring	15
5.7	AUTOSAR FlexRay Driver	15
5.8	AUTOSAR FlexRay Transceiver Driver	15
5.9	File Structure	15
5.9.1	Header File Structure	15
6	Requirements Tracing	17
7	Functional specification	20
7.1	FlexRay BSW Stack	20
7.2	Indexing Scheme	20
7.2.1	Principle	20
7.2.2	Supported Indexed Resources	24
7.3	FlexRay Interface State Machine	24
7.3.1	FlexRay Interface Main Function	26
7.4	Implementation Requirements	28
7.5	Configuration description	29
7.6	Data Communication via FlexRay	29
7.6.1	PDU Packing, PDU update bits, and Frame Construction Plans	30
7.6.2	Dynamic PDU length	31
7.6.3	AlwaysTransmit	32
7.6.4	Realization of the Time-Driven FlexRay Schedule	33
7.6.4.1	FlexRay Job List	33
7.6.4.2	FlexRay Job List Execution Function	34
7.6.5	Communication Operations	36

7.6.5.1	TransmitWithDecoupledBufferAccess	36
7.6.5.2	ProvideTxConfirmation	38
7.6.5.3	ReceiveAndStore	39
7.6.5.4	ProvideRxIndication	40
7.6.5.5	ReceiveAndIndicate	41
7.6.5.6	PREPARE_LPDU	42
7.6.5.7	FREE_OP_A	43
7.6.5.8	FREE_OP_B	43
7.6.6	Transmission with Immediate Buffer Access	43
7.7	Error Classification	44
7.7.1	Development Errors	44
7.7.2	Runtime Errors	45
7.7.3	Production Errors	45
7.7.4	Extended Production Errors	48
8	API specification	49
8.1	Imported types	49
8.2	Type definitions	49
8.2.1	Frlf_ConfigType	50
8.2.2	Frlf_StateType	50
8.2.3	Frlf_StateTransitionType	50
8.3	Function definitions	51
8.3.1	Frlf_Init	51
8.3.2	Frlf_ControllerInit	52
8.3.3	Frlf_SetAbsoluteTimer	53
8.3.4	Frlf_EnableAbsoluteTimerIRQ	54
8.3.5	Frlf_AckAbsoluteTimerIRQ	55
8.3.6	Frlf_StartCommunication	56
8.3.7	Frlf_HaltCommunication	57
8.3.8	Frlf_AbortCommunication	58
8.3.9	Frlf_GetState	59
8.3.10	Frlf_SetState	59
8.3.11	Frlf_SetWakeupChannel	60
8.3.12	Frlf_SendWUP	62
8.3.13	Frlf_GetPOCStatus	63
8.3.14	Frlf_GetGlobalTime	64
8.3.15	Frlf_AllowColdstart	65
8.3.16	Frlf_GetMacroticksPerCycle	66
8.3.17	Frlf_GetMacrotickDuration	66
8.3.18	Frlf_Transmit	67
8.3.19	Frlf_SetTransceiverMode	69
8.3.20	Frlf_GetTransceiverMode	70
8.3.21	Frlf_GetTransceiverWUReason	71
8.3.22	Frlf_ClearTransceiverWakeup	72

8.3.23 Frlf_CancelAbsoluteTimer	73
8.3.24 Frlf_GetAbsoluteTimerIRQStatus	74
8.3.25 Frlf_DisableAbsoluteTimerIRQ	76
8.3.26 Frlf_GetCycleLength	76
8.4 Optional Function Definitions	77
8.4.1 Frlf_AllSlots	77
8.4.2 Frlf_GetChannelStatus	78
8.4.3 Frlf_GetClockCorrection	79
8.4.4 Frlf_GetSyncFrameList	80
8.4.5 Frlf_GetNumOfStartupFrames	81
8.4.6 Frlf_GetWakeupRxStatus	82
8.4.7 Frlf_CancelTransmit	82
8.4.8 Frlf_DisableLPdu	84
8.4.9 Frlf_GetTransceiverError	85
8.4.10 Frlf_EnableTransceiverBranch	86
8.4.11 Frlf_DisableTransceiverBranch	87
8.4.12 Frlf_ReconfigLPdu	89
8.4.13 Frlf_GetNmVector	90
8.4.14 Frlf_GetVersionInfo	91
8.4.15 Frlf_ReadCCConfig	91
8.4.16 Frlf_EnableBusMirroring	92
8.5 Interrupt Service Routines	93
8.5.1 Frlf_JobListExec_<FrlfCluster.ShortName>	93
8.6 Callback notifications	94
8.6.1 Frlf_CheckWakeupByTransceiver	94
8.7 Scheduled functions	95
8.7.1 Frlf_MainFunction_<FrlfCluster.ShortName>	95
8.8 Expected interfaces	96
8.8.1 Mandatory Interfaces	96
8.8.2 Optional Interfaces	97
8.8.3 Configurable Interfaces	98
8.8.3.1 <Free_Op_A>	98
8.8.3.2 <Free_Op_B>	99
8.8.3.3 <UL_TxConflictNotification>	99
9 Sequence diagrams	101
9.1 Data Transmission	101
9.1.1 TransmitWithImmediateBufferAccess	101
9.1.2 TransmitWithDecoupledBufferAccess	102
9.1.3 ProvideTxConfirmation	103
9.2 Data Reception	104
9.2.1 ReceiveAndIndicate	104
9.2.2 ReceiveAndStore	105
9.2.3 ProvideRxIndication	106

9.2.4 Cancel Transmission	107
9.3 Prepare LPDU	109
10 Configuration specification	110
10.1 How to read this chapter	110
10.2 Containers and configuration parameters	110
10.2.1 Frlf	110
10.2.2 FrlfGeneral	111
10.2.3 FrlfCluster	123
10.2.4 FrlfController	138
10.2.5 FrlfTransceiver	141
10.2.6 FrlfLPdu	142
10.2.7 FrlfFrameTriggering	143
10.2.8 FrlfJobList	148
10.2.9 FrlfJob	149
10.2.10 FrlfCommunicationOperation	151
10.2.11 FrlfFrameStructure	153
10.2.12 FrlfPdusInFrame	154
10.2.13 FrlfPdu	156
10.2.14 FrlfTxPdu	157
10.2.15 FrlfRxPdu	160
10.2.16 FrlfPduDirection	161
10.2.17 FrlfConfig	161
10.2.18 FrlfClusterDemEventParameterRefs	162
10.2.19 FrlfFrameTriggeringDemEventParameterRefs	166
10.3 Published Information	167
A Not applicable requirements	168
B Change history of AUTOSAR traceable items	169
B.1 Traceable item history of this document according to AUTOSAR Release R23-11	169
B.1.1 Added Specification Items in R23-11	169
B.1.2 Changed Specification Items in R23-11	169
B.1.3 Deleted Specification Items in R23-11	169
B.2 Traceable item history of this document according to AUTOSAR Release R24-11	169
B.2.1 Added Specification Items in R24-11	169
B.2.2 Changed Specification Items in R24-11	169
B.2.3 Deleted Specification Items in R24-11	170
B.3 Traceable item history of this document according to AUTOSAR Release R25-11	170
B.3.1 Added Specification Items in R25-11	170
B.3.2 Changed Specification Items in R25-11	170
B.3.3 Deleted Specification Items in R25-11	171

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module "FlexRay Interface".

In the AUTOSAR Layered Software Architecture Layered Software Architecture, the FlexRay Interface belongs to the ECU Abstraction Layer, or more precisely, to the Communication Hardware Abstraction. This indicates the main task of the FlexRay Interface:

Provide to upper layers an abstract interface to the FlexRay Communication System. At least as far as data transmission (i.e. data sending and reception) is concerned, this interface shall be uniform for all bus systems in Autosar (FlexRay, CAN, LIN). Thus, the upper layer (Communication Services like PDU Router, Transport Protocol, and Network Management and others) may access all underlying bus systems for data transmission in a uniform manner. The configuration of the FlexRay Interface however is bus-specific, since it takes into account the specific features of the communication system.

The FlexRay Interface does not directly access the FlexRay hardware (FlexRay Communication Controller and FlexRay Transceiver), but by means of one or more hardware-specific Driver modules.

In order to access the FlexRay Communication Controller(s), the FlexRay Interface uses one or multiple FlexRay Driver modules, which abstract the specific features and interfaces (CHI) of the respective FlexRay Communication Controller(s).

Likewise, in order to access the FlexRay Transceiver(s), the FlexRay Interface shall use one or multiple FlexRay Transceiver Driver module(s), which abstract the specific features and interfaces of the respective FlexRay Transceiver(s)

Therefore, the FlexRay Interface executable code (however, not the configuration used during runtime) shall be completely independent of the FlexRay Communication Controller(s) and the FlexRay Transceiver(s).

Note: The FlexRay Interface is specified in a way that allows for object code delivery of the code module, following the "one-fits-all" principle, i.e. the entire configuration of the FlexRay Interface can be carried out without modifying any source code. Thus, the configuration of the FlexRay Interface can be carried out largely without detailed knowledge of the underlying hardware.

The FlexRay Interface provides to upper layer AUTOSAR BSW modules the following groups of functions:

- initialization
- data transmission (sending and reception)
- start/halt/abort communication
- FlexRay specific functions (e.g. send wake-up pattern)

- set operation mode
- get status information
- various timer functions

1.1 General Hints

In general, the FlexRay Interface has no knowledge of the origin of a PDU passed to it in an API service call.

Therefore, throughout this document, the term "PDU" is being used for PDUs originating from or sent to:

- AUTOSAR Com (I-PDU) via the PDU-Router, or
- AUTOSAR FlexRay TP (N-PDU), or
- AUTOSAR FlexRay NM
- AUTOSAR XCP

In addition to the above-mentioned AUTOSAR BSW modules, the FrIf shall, with the functionality described within the specification in hand, also support other non-AUTOSAR upper layer software modules (Complex Drivers), provided that these modules interact with the FrIf in the same manner as the upper layer AUTOSAR BSW modules.

Throughout this document, several scenarios for changing configuration data are mentioned. They are being used as follows:

- "pre compile time" = carried out before compiling the code of the FlexRay Interface, since the code generation depends on this setting.
- "at system configuration time" = static configuration parameters stored in the FlexRay Interface; may be defined after compilation of the code of the FlexRay Interface ("link time" or "post build time"), but have to be defined before the first execution of the FlexRay Interface code.
- "during runtime" = dynamically switching (in POC:normal active state of the FlexRay CC, if supported) between different configuration parameter sets stored in the static configuration of the FlexRay Interface, or the FlexRay Driver, respectively.

Everything not explicitly mentioned in this document, should be considered as implementation-specific.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to this document that are not included in the AUTOSAR Glossary [1]:

Acronym:	Description:
BSW	(AUTOSAR) Basic Software
CAS	Collision Avoidance Symbol
CC	(FlexRay) Communication Controller
CDD	Complex Driver
CHI	Controller Host Interface of a FlexRay CC
COM	Communication (AUTOSAR BSW module)
ComM	Communication Manager (AUTOSAR BSW module)
DEM	Diagnostic Event Manager (AUTOSAR BSW module)
DET	Default Error Tracer (AUTOSAR BSW module)
FrIf	FlexRay Interface (AUTOSAR BSW module)
FrNm	FlexRay Network Management (AUTOSAR BSW module)
FrTp	FlexRay Transport Layer (AUTOSAR BSW module)
ISR	Interrupt Service Routine
LSduR	Data link layer service data unit Router (AUTOSAR BSW module)
MCG	Module Configuration Generator
PduR	PDU Router (AUTOSAR BSW module)
POC	Protocol Operation Control
WUDOP	Wake-Up During Operation
WUP	Wake-Up Pattern
WUS	Wake-Up Symbol
System Designer	The person responsible for the configuration of all system parameters that do not influence the executable code itself (i.e. the sequence of instructions executed during runtime), but the data used to configure which operations this executable code performs on which data and at which points in time.

Abbreviation:	Description:
i.e.	[lat.] id est = [eng.] that is
e.g.	[lat.] exempli gratia = [eng.] for example
N/A	not applicable

3 Related documentation

3.1 Input documents & related standards and norms

- [1] Glossary
AUTOSAR_FO_TR_Glossary
- [2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral
- [3] FlexRay Communications System Protocol Specification V3.0
- [4] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral
- [5] Requirements on FlexRay
AUTOSAR_CP_RS_FlexRay
- [6] Layered Software Architecture
AUTOSAR_CP_EXP_LayeredSoftwareArchitecture
- [7] FlexRay Communications System Protocol Specification V2.1
<http://www.flexray.com/>

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for FrIf.

Thus, the specification SWS BSW General shall be considered as additional and required specification for FrIf.

4 Constraints and assumptions

4.1 Limitations

The FlexRay BSW modules are only able to handle a single thread of execution per Cluster. The execution for a particular Cluster must not be pre-empted by itself for the same Cluster. The same applies to the execution of the FlexRay Job List Execution Function.

It is not possible to transmit signals, PDUs, and/or L-SDUs, which exceed the available buffer size of the used FlexRay CC during normal operation. Longer signals, PDUs, and/or L-SDUs have to be transmitted using the FlexRay Transport Protocol.

Note: The FlexRay Interface does not make any PDU payload-dependent routing decisions.

Note: In order for the AUTOSAR FlexRay BSW (FrIf and FlexRay Driver) modules to be able to control a FlexRay CC, this CC must allow for configuring its transmit/receive buffers to support the Cycle Counter Filter Criterion / (Support of Slot/Cycle Multiplexing)

For 2.1 FlexRay Hardware, the following Cycle Counter Filtering is possible

$$\text{Cycle Number} = (B + n * 2R) \bmod 64$$

with exactly one tuple of values for B and 2R, where:

- Base Cycle $B \in [0 \dots 63]$
- Cycle Repetition $2R$; $R \in [0 \dots 6]$
- Variable $n = 0 \dots 63$
- $B < 2R$

For 3.0 FlexRay Hardware, the Cycle Counter Filtering shall be possible as described in [3, FlexRay Communications System Protocol Specification V3.0]

4.2 Applicability to car domains

The FlexRay BSW Stack can be used wherever high data rates and fault tolerant communication (in conjunction with AUTOSAR COM) are required. Of course, it can also be used for less-demanding use cases, i.e. for low data rates or non-fault-tolerant communication. Furthermore, it enables the synchronized operation of several ECUs within a car.

5 Dependencies to other modules

5.1 AUTOSAR Operating System

[SWS_FrIf_05099]

Upstream requirements: [SRS_BSW_00432](#)

[There is one dedicated FlexRay Job List Execution Function for each FlexRay Cluster.]

[SWS_FrIf_05100]

Upstream requirements: [SRS_BSW_00432](#)

[The FlexRay Interface module shall execute the Flexray Job List Execution Function.]

Note: It is up to the implementer whether the FlexRay Job List Execution Functions run in a task context or in an ISR.

5.2 All Upper Layer AUTOSAR BSW Modules

[SWS_FrIf_05050]

Upstream requirements: [SRS_Fr_05000](#)

[The calling of the FlexRay Job List Execution Function by the FlexRay Interface module synchronously to the FlexRay Global Time shall ensure that both the indication (to an upper layer BSW module) of received data and the request (to an upper layer BSW module) for data to be sent occur synchronously to the FlexRay Global Time.]

[SWS_FrIf_05148]

Upstream requirements: [SRS_BSW_00426](#)

[The FlexRay Interface module shall ensure data consistency in its buffers.]

Rationale for [\[SWS_FrIf_05148\]](#): If the respective upper layer BSW module does not operate synchronously to the FlexRay Global Time, these occurrences are asynchronous to the code execution of this BSW module.

5.3 AUTOSAR LSdu-Router

The FrIf module declares and calls some callback functions of the LSdu-Router in order to confirm transmission and notify reception of L-SDUs.

5.4 AUTOSAR FlexRay Network Management

The FrIf module declares and calls some callback functions of the FlexRay Network Management in order to confirm transmission and notify reception of PDUs.

5.5 AUTOSAR FlexRay Transport Protocol

The FrIf module declares and calls some callback functions of the FlexRay Transport Protocol in order to confirm transmission and notify reception of PDUs.

5.6 AUTOSAR Bus Mirroring

The FrIf module calls a callback function of the Bus Mirroring module in order to report received and transmitted frames, which in turn calls some service functions of the FrIf module to acquire the network state.

5.7 AUTOSAR FlexRay Driver

The FrIf module has a tight relation to the FlexRay Driver since many of the FlexRay-related services offered by the FrIf module to upper layer BSW modules are actually carried out by the FlexRay Driver BSW module. For those services, the FrIf module mainly performs only an abstraction of the communication hardware specific information (e.g. the topology of the FlexRay Communication System) and then calls the respective FlexRay Driver with the appropriate parameters.

The FlexRay Driver module has to be the only BSW module which has to run necessarily synchronous to the FlexRay Interface.

5.8 AUTOSAR FlexRay Transceiver Driver

The FrIf module has a tight relation to the FlexRay Transceiver Driver since calls of API services of the FlexRay Transceiver Driver are also routed through the FrIf module in order to abstract the communication hardware specific information (e.g. the topology of the FlexRay Communication System).

5.9 File Structure

5.9.1 Header File Structure

Please refer to the chapter "Header file structure" in [2, General Specification of Basic Software Modules].

[SWS_FrIf_05087]

Upstream requirements: [SRS_BSW_00426](#)

[The FrIf module source code file(s) shall include SchM_FrIf.h if data consistency mechanisms of the BSW scheduler are required as described in [\[2, General Specification of Basic Software Modules\]](#).]

[SWS_FrIf_05090]

Upstream requirements: [SRS_BSW_00004](#)

[The header file FrIf.h shall contain a software and specification version number.]

[SWS_FrIf_05095]

Upstream requirements: [SRS_BSW_00004](#)

[Mirror.h contains the declaration of the API service the Bus Mirroring module offers to the FlexRay Interface. This header is only included if Bus Mirroring is enabled (see FrIfBusMirroringSupport).]

6 Requirements Tracing

The following tables reference the requirements specified in [4, SRS BSW General] and [5, SRS Flex Ray] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_Frlf_05090] [SWS_Frlf_05095]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_Frlf_05003]
[SRS_BSW_00162]	The AUTOSAR Basic Software shall provide a hardware abstraction layer	[SWS_Frlf_05107]
[SRS_BSW_00170]	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	[SWS_Frlf_05089]
[SRS_BSW_00171]	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	[SWS_Frlf_05089]
[SRS_BSW_00304]	All AUTOSAR Basic Software Modules shall use only AUTOSAR data types instead of native C data types	[SWS_Frlf_05001]
[SRS_BSW_00334]	Machine readable module description	[SWS_Frlf_05089]
[SRS_BSW_00336]	Basic SW module shall be able to shutdown	[SWS_Frlf_05006]
[SRS_BSW_00342]	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	[SWS_Frlf_05078]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_Frlf_05069]
[SRS_BSW_00348]	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	[SWS_Frlf_05001]
[SRS_BSW_00353]	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	[SWS_Frlf_05001]
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_Frlf_05003]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_Frlf_05283]
[SRS_BSW_00375]	Basic Software Modules shall report wake-up reasons	[SWS_Frlf_05036]





Requirement	Description	Satisfied by
[SRS_BSW_00378]	AUTOSAR shall provide a boolean type	[SWS_Frlf_05001]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_Frlf_05069]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_Frlf_05003]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_Frlf_05002]
[SRS_BSW_00411]	All AUTOSAR Basic Software Modules shall apply a naming rule for enabling/disabling the existence of the API	[SWS_Frlf_05002]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_Frlf_05003]
[SRS_BSW_00426]	BSW Modules shall ensure data consistency of data which is shared between BSW modules	[SWS_Frlf_05087] [SWS_Frlf_05148]
[SRS_BSW_00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	[SWS_Frlf_05099] [SWS_Frlf_05100] [SWS_Frlf_05119]
[SRS_BSW_00458]	Classification of production errors	[SWS_Frlf_05321] [SWS_Frlf_05322] [SWS_Frlf_05323] [SWS_Frlf_05324] [SWS_Frlf_05325] [SWS_Frlf_05326]
[SRS_Fr_05000]	Synchronous SW Modules shall be supported	[SWS_Frlf_05050]
[SRS_Fr_05007]	The FlexRay Interface shall be able to communicate with at least four Flex Ray CCs via the appropriate FlexRay Driver(s)	[SWS_Frlf_05053] [SWS_Frlf_05111] [SWS_Frlf_05112] [SWS_Frlf_05113]
[SRS_Fr_05010]	Each PDU shall have one PDU-ID	[SWS_Frlf_05052]
[SRS_Fr_05013]	The local Memory Space shall be initialized	[SWS_Frlf_05003]
[SRS_Fr_05015]	The FlexRay Interface shall provide a software interface to start-up a specific FlexRay CC	[SWS_Frlf_05005]
[SRS_Fr_05016]	A FlexRay CC Communication shall be aborted when wanted	[SWS_Frlf_05007]
[SRS_Fr_05018]	The FlexRay Interface shall provide a software interface to send a wake-up pattern on a channel or CC	[SWS_Frlf_05011]
[SRS_Fr_05022]	FlexRay CC POC Status shall be available	[SWS_Frlf_05014]
[SRS_Fr_05027]	A PDU shall be transmitted via the FlexRay communication system	[SWS_Frlf_05063]
[SRS_Fr_05031]	A FlexRay CC shall be initialized and configured	[SWS_Frlf_05004] [SWS_Frlf_05117]
[SRS_Fr_05039]	The Operation Mode of a FlexRay Transceiver shall be set	[SWS_Frlf_05034]
[SRS_Fr_05042]	The FlexRay Interface shall allow switching from one configuration to another one in Normal Active Mode	[SWS_Frlf_05061]





Requirement	Description	Satisfied by
[SRS_Fr_05056]	Configuration of the FlexRay Interface shall be done at System Configuration Time	[SWS_Frlf_05054]
[SRS_Fr_05063]	A FlexRay CC Communication shall be halted when wanted	[SWS_Frlf_05006]
[SRS_Fr_05096]	Communication controllers shall be assigned to FlexRay Driver.	[SWS_Frlf_05060]
[SRS_Fr_05097]	The FlexRay Interface shall be able to communicate with at least four Flex Ray Drivers	[SWS_Frlf_05057]
[SRS_Fr_05126]	PDU Update/Valid Information shall be handled	[SWS_Frlf_05056]
[SRS_Fr_05130]	The FlexRay Interface shall support PDU transmission buffer queues	[SWS_Frlf_05058]
[SRS_Fr_05157]	The Operation Mode of a FlexRay Transceiver shall be available	[SWS_Frlf_05035]
[SRS_Fr_05158]	The wake-up reason of a specific FlexRay Transceiver device shall be available	[SWS_Frlf_05036]
[SRS_Fr_05161]	Pending Wake-up Events of a Transceiver shall be cleared if necessary	[SWS_Frlf_05039]
[SRS_Fr_05170]	PDUs received via the FlexRay communication system shall be retrieved	[SWS_Frlf_05062]

Table 6.1: Requirements Tracing

7 Functional specification

7.1 FlexRay BSW Stack

As part of the AUTOSAR Layered Software Architecture according to [Figure 7.1](#), the FlexRay BSW modules also form a layered software stack. [6, AUTOSAR_EXP_LayeredSoftwareArchitecture] depicts the basic structure of this FlexRay BSW stack. The FrIf module accesses several CCs using the FlexRay Driver layer, which can be made up of several FlexRay Drivers modules. The FlexRay Transceivers are not shown in this figure; however, the structure that applies to the FlexRay Drivers and the FlexRay CCs analogously applies to the FlexRay Transceiver Drivers and the FlexRay Transceivers.

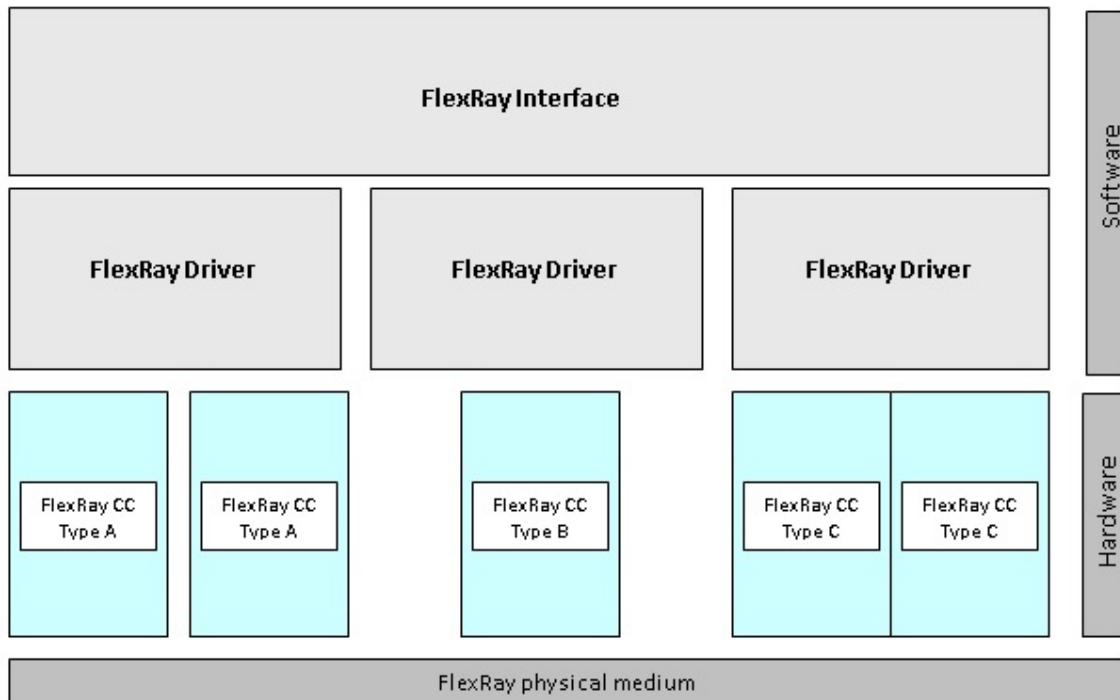


Figure 7.1: Basic Structure of the FlexRay Bsw Stack

7.2 Indexing Scheme

7.2.1 Principle

Most of the FrIf module's API services used for accessing the numerous (hardware and software) resources ¹ map to corresponding API services of the underlying FlexRay Driver(s), or FlexRay Transceiver Driver(s), respectively.

¹E.g. timers, configuration data sets, etc.

In order to select those resources spread over the various entities ² accessed via the FrIf module, the FlexRay-related AUTOSAR BSW modules use an indexing scheme that is exemplarily described in [Figure 7.2](#) and [Figure 7.3](#).

Definition ControllerIndex: The ControllerIndex is an abstract, unique, zero-based consecutive index to achieve the abstraction of the FlexRay Communication Controllers, independent of their type, location, and access method.

Definition ClusterIndex: The ClusterIndex is an abstract, unique, zero-based consecutive index to achieve the abstraction of the FlexRay Clusters, independent of their type, location, and access method.

Definition ChannelIndex: The ChannelIndex has either the value FR_CHANNEL_A or FR_CHANNEL_B. In combination with the ControllerIndex, the corresponding FlexRay Transceiver is identified.

[SWS_FrIf_05052]

Upstream requirements: [SRS_Fr_05010](#)

[The FrIf module shall achieve the abstraction (of the CCs and Drivers) by providing to the upper layer BSW modules an abstract, unique, zero-based consecutive index for each sort of resource, independent of their type, location, and access method.]

Rationale: The FrIf module achieves the abstraction (of the CCs and Drivers) by providing these abstract indices to the upper layer BSW modules.

The FrIf module API service uses the abstract index passed to it by the upper layer BSW module to retrieve:

- the function pointer to a corresponding lower layer BSW module's API service from a static configuration data table containing function pointers to all API services of all lower layer BSW modules called by the FrIf module, and
- the translated index used in the call to the lower layer BSW module's API service from a static configuration data table.

Since this static configuration data table contains function pointers to the lower layer BSW module's API services, it obviously has to be linked against the linked and located code of the lower layer BSW modules.

The FrIf module then calls the corresponding lower layer BSW module's API service via the function pointer and passes the translated index in the API call.

The function descriptions in [Chapter 8](#) specify the required calls of corresponding lower layer BSW module's API services in detail.

²FlexRay Drivers, FlexRay Communication Controllers, FlexRay Transceiver Drivers, and FlexRay Transceivers

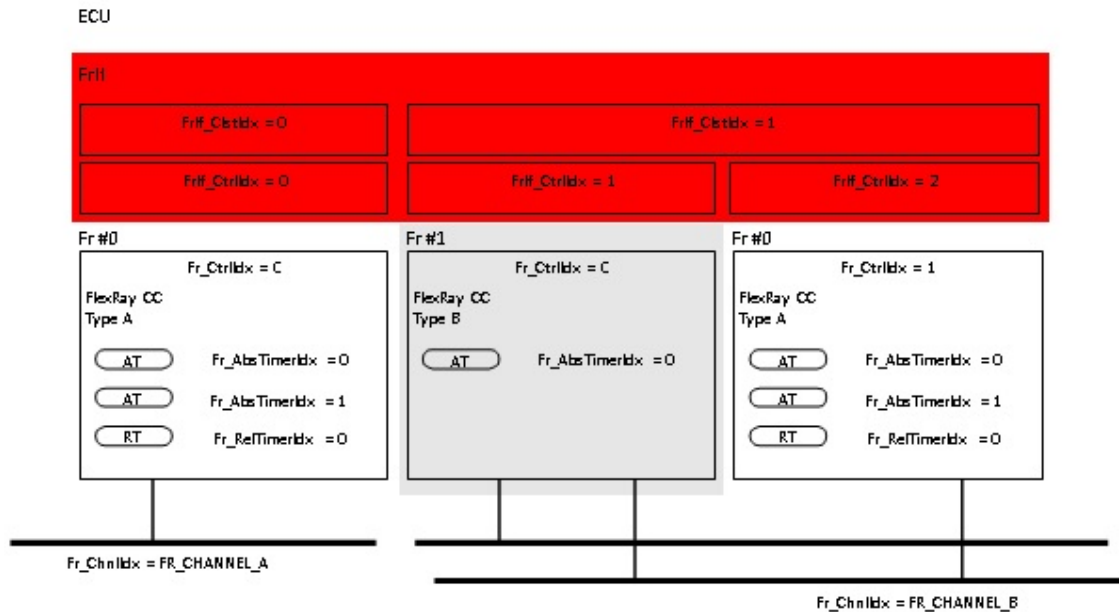


Figure 7.2: CC Indexing Scheme of the FlexRay Interface

[SWS_FrIf_05060]

Upstream requirements: [SRS_Fr_05096](#)

[In order to abstract for upper layer BSW modules the various CCs, which the FrIf module controls via the FlexRay Driver modules, the FrIf module offers an abstract, unique, zero-based consecutive index FrIfCtrlIdx as configuration parameter, which maps to a tuple of FlexRay Driver API Service function pointer and CC index Fr_CtrlIdx.]

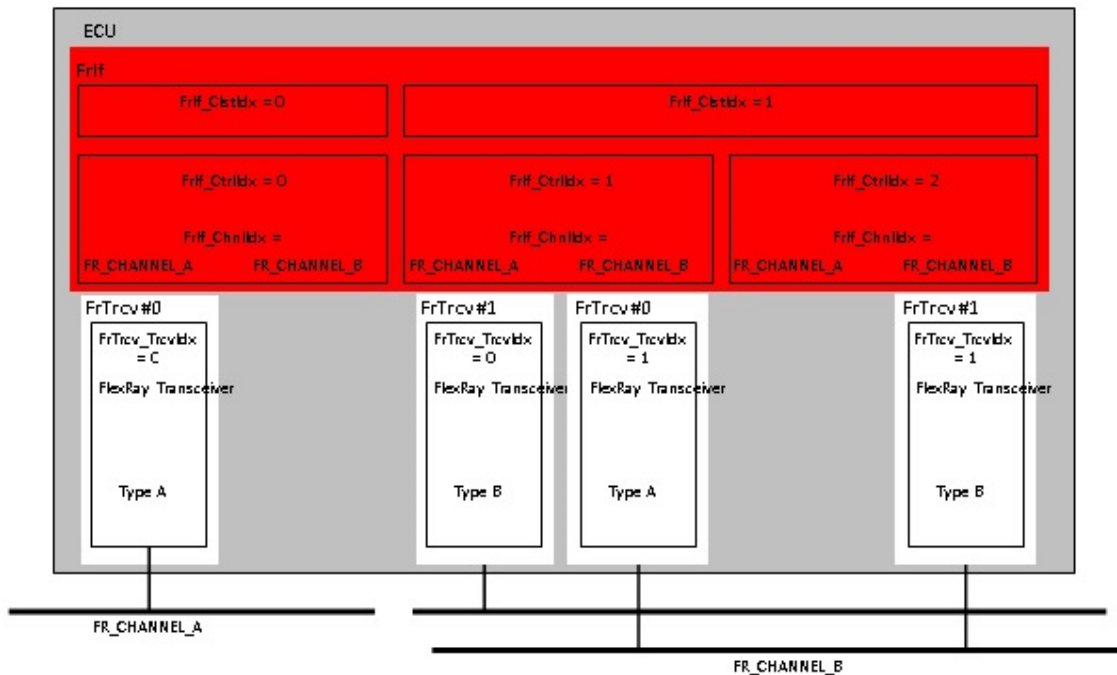


Figure 7.3: Flexray Transceiver Indexing Scheme of the FlexRay Interface

In order to abstract for upper layer BSW modules the various FlexRay Transceiver modules, which the Frlf module accesses via the FlexRay Transceiver Driver modules, the Frlf module takes advantage of the fact that each FlexRay Transceiver module is unambiguously assigned to a specific Channel on a specific FlexRay CC.

Therefore, the Frlf module abstracts the various FlexRay Transceivers by a combination of the two indices **FrIf_CtrlIdx** (Controller Index) and **FrIf_ChnlIdx** (Channel Index) and maps this to a tuple of FlexRay Transceiver Driver API Service function pointer and FlexRay Transceiver index **FrTrcv_TrcvIdx**. (Transceiver Index)

The function descriptions in [Chapter 8](#) specify the required mapping of upper layer BSW module's parameters to corresponding lower layer BSW module's API services in detail."

[SWS_Frlf_05107]

Upstream requirements: [SRS_BSW_00162](#)

[Besides hardware and software resources, the Frlf module also numbers the logical structure elements presented by FlexRay with an abstract, unique, zero-based consecutive index.

The static configuration data of the Frlf module contains a data structure that specifies which FlexRay CC modules and which FlexRay Transceiver modules are connected to which Clusters, or in other words, that maps each value of **FrIf_CtrlIdx** to (one, or in general) a set of values for **FrIf_CtrlIdx** and tuples of (**FrIf_CtrlIdx**, **FrIf_ChnlIdx**).]

[SWS_FrIf_05110] [The FrIf module shall number all PDUs to be transmitted with an abstract, unique, zero-based consecutive index TxPduld.]

Note: This index is used in the FrIf API service FrIf_Transmit() and allows the FrIf module to quickly identify (e.g. by a table look-up) the PDU that is passed to it by an upper layer BSW module, and to process it accordingly.

7.2.2 Supported Indexed Resources

[SWS_FrIf_05057]

Upstream requirements: [SRS_Fr_05097](#)

[It shall be possible that the FrIf module can be configured to support at least four (possibly different) FlexRay Drivers to access the FlexRay Communication Controllers.]

[SWS_FrIf_05053]

Upstream requirements: [SRS_Fr_05007](#)

[It shall be possible that the FrIf module can be configured using the parameter FRIF_CTRL_IDX to support at least four (possibly different) FlexRay CCs.]

[SWS_FrIf_05111]

Upstream requirements: [SRS_Fr_05007](#)

[It shall be possible that the FrIf module can be configured to support one of both or both FlexRay Channels as specified in [\[7, FlexRay Communications System Protocol Specification V2.1\]](#).]

[SWS_FrIf_05112]

Upstream requirements: [SRS_Fr_05007](#)

[It shall be possible that the FrIf module can be configured using the parameter FRIF_CLST_IDX to support at least four FlexRay Clusters.]

[SWS_FrIf_05113]

Upstream requirements: [SRS_Fr_05007](#)

[It shall be possible that the FrIf module can be configured using the parameter FRIF_ABS_TIMER_IDX to support at least one absolute timer per FlexRay CCs.]

7.3 FlexRay Interface State Machine

[SWS_FrIf_05115] [In order to allow to control the communication operations of the FlexRay system, the FrIf module shall implement a behavior, which is defined using a simple state machine (one per FlexRay cluster), called FlexRay Interface State Machine]

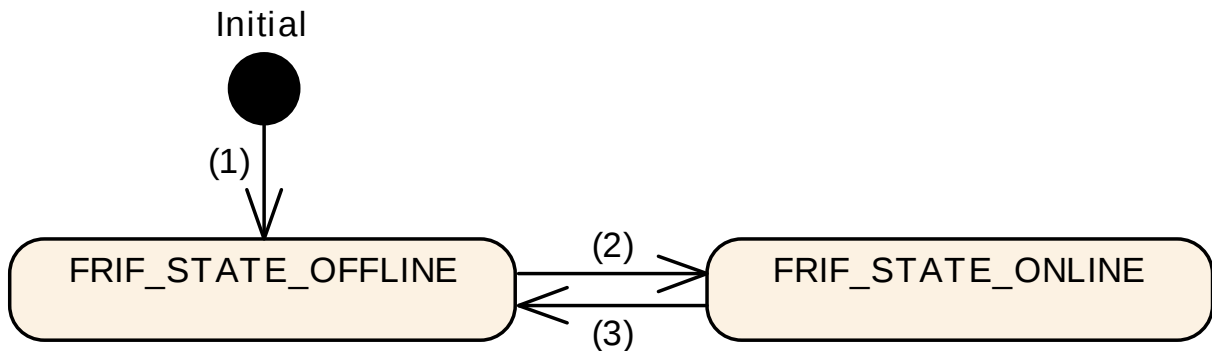


Figure 7.4: FlexRay Interface State Machine

State	Description
FRIF_STATE_OFFLINE	No communication services are executed (see Chapter 7.6 for details)
FRIF_STATE_ONLINE	All communication services (reception, transmission, transmission confirmation) are executed (see Chapter 7.6 for details).

Table 7.1: FlexRay State Machine Transitions (1)

[SWS_FrIf_05117]

Upstream requirements: [SRS_Fr_05031](#)

[During initialization of the FrIf by executing FrIf_Init() the FrIf_State for each cluster shall be initialized with state 'FRIF_STATE_OFFLINE'.

The transitions are requested by an API service FrIf_SetState() which takes the Cluster to process on and the Transition name to invoke.]

[SWS_FrIf_05118] [If the FrIf module's environment calls the function FrIf_SetState with parameter FrIf_StateTransition = FRIF_GOTO_ONLINE and if the current state for the requested cluster is FRIF_STATE_OFFLINE, the FrIf module shall take the current state of the requested cluster to FRIF_STATE_ONLINE.".

If the FrIf module's environment calls the function FrIf_SetState with parameter FrIf_StateTransition = FRIF_GOTO_OFFLINE and if the current state for the requested cluster is FRIF_STATE_ONLINE, the FrIf module shall take the current state of the requested cluster to FRIF_STATE_OFFLINE.".

Otherwise, do not perform a state transition.]

For details see [Figure 7.4](#) and [Table 7.2](#)

Transition Name	Transitions (see Figure 7.4)	Description
FRIF_GOTO_ONLINE	(2)	Transition resulting in FrIf_State FRIF_STATE_ONLINE
FRIF_GOTO_OFFLINE	(3)	Transition resulting in FrIf_State FRIF_STATE_OFFLINE

Table 7.2: FlexRay State Machine Transitions (2)

[SWS_FrIf_05501] [If the API `FrIf_SetState` with parameter `FRIF_STATE_OFFLINE` is called, the FlexRay Interface module shall check the parameter "TxConfCounter" for every PDU. If the value for the corresponding PDU is greater than 0, the FlexRay Interface shall call the upper layer using the API `LSduR_FrIfConfirmation(id, E_NOT_OK)`.]

Note: It has to be ensured that the FlexRay Interface does not lose the TxConfCounter values at the point in time the API `FrIf_SetState` with parameter `FRIF_STATE_OFFLINE` is called.

7.3.1 FlexRay Interface Main Function

The FlexRay Interface Main Function needs to be called cyclically from a task body provided by the BSW Scheduler with a calling period (`FRIF_MAINFUNCTION_PERIOD`) depending on the FlexRay Cycle length and configurable at system configuration time.

Since the Cycle length of each Cluster is independent, the desired calling period of the FlexRay Interface Main Function might differ from Cluster to Cluster, except for "Transmission with Immediate Buffer Access".

[SWS_FrIf_05119]

Upstream requirements: [SRS_BSW_00432](#)

[The `FrIf` module shall provide one dedicated FlexRay Interface Main Function for each FlexRay Cluster that is controlled by that `FrIf` module.]

[SWS_FrIf_05283]

Upstream requirements: [SRS_BSW_00373](#)

[The API names of the FlexRay Interface Main Functions shall obey the following pattern:

`FrIf_MainFunction_<FrIfCluster.ShortName>` where `FrIfCluster.ShortName` is the Short Name of the corresponding `FrIfCluster`.]

[SWS_FrIf_15120] [The Main Function monitors and controls the continuous execution of the FlexRay Job List Execution Function including the (re)synchronization if the current FlexRay Interface State Machine is `FRIF_STATE_ONLINE`.]

[SWS_FrIf_01124] [If Bus Mirroring is enabled globally (see `FrIfBusMirroringSupport`), then call `Fr_GetChannelStatus` for all controllers of each FlexRay cluster for which mirroring has been activated with a call to `FrIf_EnableBusMirroring()`, merge the states reported for the controllers of one cluster with a binary OR, and then call `Mirror_ReportFlexRayChannelStatus()` with the cluster, `Fr_ChannelAStatusPtr`, and `Fr_ChannelBStatusPtr` to report the aggregated channel states to the Bus Mirroring module.]

[SWS_FrIf_25120] [If one of the optional cluster-specific configuration parameters `FRIF_E_NIT_CH_A`, `FRIF_E_NIT_CH_B`, `FRIF_E_SW_CH_A`, `FRIF_E_SW_CH_B` or

FRIF_E_ACS_CH_A, FRIF_E_ACS_CH_B exists, then call FrIf_GetChannelStatus for each FlexRay controller of the cluster and report the status to DEM as described below.]

[SWS_FrIf_35120] [If the optional configuration parameter FRIF_E_NIT_CH_A exists, then the channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_NIT_CH_A, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel A NIT status data vSS!SyntaxError, vSS!Bviolation) is set or as Dem_SetEventStatus (FRIF_E_NIT_CH_A, DEM_EVENT_STATUS_PREPASSED) when none of these error bits is set.]

[SWS_FrIf_45120] [If the optional configuration parameter FRIF_E_NIT_CH_B exists, then the channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_NIT_CH_B, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel B NIT status data vSS!SyntaxError, vSS!Bviolation) is set or as Dem_SetEventStatus (FRIF_E_NIT_CH_B, DEM_EVENT_STATUS_PREPASSED) when none of these error bits is set.]

[SWS_FrIf_55120] [If the optional configuration parameter FRIF_E_SW_CH_A exists, then the channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_SW_CH_A, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel A symbol window status data vSS!SyntaxError, vSS!Bviolation, vSS!TxConflict) is set or as Dem_SetEventStatus (FRIF_E_SW_CH_A, DEM_EVENT_STATUS_PREPASSED) when none of these error bits is set.]

[SWS_FrIf_65120] [If the optional configuration parameter FRIF_E_SW_CH_B exists, then the channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_SW_CH_B, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel B symbol window status data vSS!SyntaxError, vSS!Bviolation, vSS!TxConflict) is set or as Dem_SetEventStatus (FRIF_E_SW_CH_B, DEM_EVENT_STATUS_PREPASSED) when none of these error bits is set.]

[SWS_FrIf_75120] [If the optional configuration parameter FRIF_E_ACS_CH_A exists, then the channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_ACS_CH_A, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel A aggregated channel status vSS!SyntaxError, vSS!ContentError, vSS!Bviolation, vSS!TxConflict) is set or as Dem_SetEventStatus (FRIF_E_ACS_CH_A, DEM_EVENT_STATUS_PREPASSED) when none of these error bits is set.]

[SWS_FrIf_85120] [If the optional configuration parameter FRIF_E_ACS_CH_B exists, then the channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_ACS_CH_B, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel B aggregated channel status vSS!SyntaxError, vSS!ContentError, vSS!Bviolation, vSS!TxConflict) is set or as Dem_SetEventStatus (FRIF_E_ACS_CH_B, DEM_EVENT_STATUS_PREPASSED) when none of these error bits is set.]

[SWS_FrIf_95120] [If a loss of the JobList's synchronization (see JobListAsyncFlag) or a miss of execution was detected, the following steps shall be performed:

1. Get the global time (FrIf_GetGlobalTime())
 - If FrIf_GetGlobalTime() returns E_NOT_OK, stop here
 - If FrIf_GetGlobalTime() returns E_OK, continue with step 2
2. add some 'time buffer' (i.e. some timespan which takes jitter into account)
3. search the FlexRay Job List for the next job, i.e. that job with an invocation time greater than the current global time + 'time buffer'.
4. set the JobListPointer to that job and program the absolute timer with this job's invocation time (now the FlexRay Job List is synchronized again)
5. clear the JobListAsyncFlag
6. Enable the absolute timer interrupt

]

7.4 Implementation Requirements

[SWS_FrIf_05096] [The FlexRay Interface executable code (however, not the configuration used during runtime) shall be completely independent of the FlexRay Communication Controller(s) and the FlexRay Transceiver(s).]

[SWS_FrIf_05069]

Upstream requirements: [SRS_BSW_00404](#), [SRS_BSW_00345](#)

[The FrIf module shall support pre-compile time, link-time and post-build-time configuration.]

[SWS_FrIf_05284] [The FrIf module shall implement link-time and post-build-time configuration data as read-only data structures.]

[SWS_FrIf_05285] [The FrIf module shall immediately reference link-time configuration data by the implementation,]

[SWS_FrIf_05078]

Upstream requirements: [SRS_BSW_00342](#)

[The FrIf module shall implement the API functions specified by the FrIf SWS as real C code functions and shall not implement the API functions as macros.]

Note: The rationale of SWS_FrIf_05078 is to allow object code module integration.

[SWS_Frlf_05244] [The Frlf module shall pad transmitted PDUs that are located on a Frlf L-Sdu where FrlfAllowDynamicLSduLength is set to false, if the size is smaller than the configured size of the PDU. Padding shall be done with the configured FrlfUnused BitValue.]

7.5 Configuration description

[SWS_Frlf_05089]

Upstream requirements: [SRS_BSW_00171](#), [SRS_BSW_00170](#), [SRS_BSW_00334](#)

[The Frlf module shall provide an XML file that contains the data which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.]

The description of the configuration and initialization data itself is not part of this specification but very implementation specific.]

7.6 Data Communication via FlexRay

FlexRay in general is a deterministic time-driven communication system.

Each datum that should be transmitted or received has to be scheduled at system configuration time.

This even holds true for data that - from the application's point of view - are considered event-driven.

Note: When looking only at specific instances of the AUTOSAR FlexRay software modules running on a specific ECU it is not possible to "anticipate" the exact point in time when a certain FlexRay frame is being sent (or received, respectively) in the Dynamic Segment of the FlexRay Cycle.

[SWS_Frlf_05054]

Upstream requirements: [SRS_Fr_05056](#)

[The Frlf module shall define the resources (e.g. a buffer in the FlexRay Communication Controller or FlexRay Driver) needed for data transmission (or reception, respectively) at system configuration time specifically for data transmission (or reception, respectively).]

Note: There is no true spontaneous event-driven data communication on FlexRay. Even application data that occur at unpredictable points in time (i.e. "event-driven"), and that should be transmitted via FlexRay, have to be scheduled for transmission at system configuration time.

7.6.1 PDU Packing, PDU update bits, and Frame Construction Plans

In accordance with basic AUTOSAR rules, the API services that the FrIf module provides to upper layer BSW modules for data transmission and data reception are PDU-based.

[SWS_FrIf_05121] [The FrIf module shall be capable of packing multiple PDUs into one FlexRay Frame.]

Rationale for SWS_FrIf_05121: Bus-independent AUTOSAR PDUs have a maximal length of 8 bytes, but according to [7, FlexRay Communications System Protocol Specification V2.1] a FlexRay Frame can contain as many as 254 bytes of payload data.

Note: It is also allowed to define PDUs which are larger than 8 bytes. Please be aware that PDUs greater than 8 bytes are not bus independent any more!

[SWS_FrIf_05122] [The FrIf module shall take the information on how to pack PDUs into FlexRay Frames from the so-called Frame Construction Plans. The rules defining how to pack PDUs into FlexRay Frames are defined at system configuration time]

[SWS_FrIf_05123] [The Frame Construction Plan shall be stored in the static configuration of the FrIf module (configuration parameter FrIfFrameStructure, see FrIf05370).]

[SWS_FrIf_05124] [If multiple PDUs are packed into a single FlexRay Frame and if the FrIf module recognizes the update of at least one of the contained PDUs, then the FrIf module shall transmit this FlexRay Frame.]

Note: As a result, the space associated with PDUs in this FlexRay Frame that have not been updated by the upper layer BSW module will also be transmitted. This does not necessarily mean that the previous values of those PDUs are transmitted. On the contrary, in case the parameter 'FrIfUnusedBitValue' does not exist, arbitrary values for those PDUs will be transmitted.

[SWS_FrIf_05723] [In case the parameter 'FrIfUnusedBitValue' exists, all the unused bits within the Frame Construction Plan shall be set to the configured value 'FrIfUnusedBitValue' while assembling the frame on sender side.]

[SWS_FrIf_05725] [The FlexRayInterface shall ensure that unused spaces within the frame construction plan only contain deterministic values (instead of possible random data).

For this purpose, the value given by the parameter 'FrIfUnusedBitValue' shall be used to fill unused spaces with this value.]

[SWS_FrIf_05125] [It shall be possible to configure (configuration parameter FrIfPduUpdateBitOffset, see FrIf06071) for each PDU a dedicated PDU update bits in the FlexRay Frame. The FrIf module shall identify the position of the PDU update bits for each PDU using the information stored in configuration parameter FrIfPduUpdateBitOffset.]

[SWS_FrIf_05056]

Upstream requirements: [SRS_Fr_05126](#)

[The receiving FrIf module shall evaluate the PDU Update-bit (if configured) to recognize the update of the PDU associated with this PDU update bits]

Rationale: In order for the receiving FrIf module to be able to determine which of the PDUs in a received FlexRay Frame have actually been updated by the upper layer BSW module (by a call of FrIf_Transmit()) on the transmitter side, additional update information, so called PDU update bits within the FlexRay Frame, shall be transmitted to the receiving FrIf module.

Note: A details description of the update bits handling is described in the Communication Operation, [Chapter 7.6.3](#) "TransmitWithDecoupledBufferAccess"

[SWS_FrIf_05126] [This PDU update bits shall be located at an arbitrary bit position in the Frame Construction Plan that is not occupied by any PDU.]

[SWS_FrIf_05127] [The configuration of update bits for the PDUs and the definition of the location of the update bits within the FlexRay Frame are performed at system configuration time [Configuration Parameter FrIfPduUpdateBitOffset, see FrIf06071]]

[SWS_FrIf_05128] [If no update bit is configured for a specific PDU, the FrIf module shall assume this PDU to be always valid and the FrIf module shall always indicate its reception to the upper layer BSW module on the receiver side.]

[SWS_FrIf_05758] [In case the parameter 'FrIfAllowDynamicLSduLength' exists and is set to TRUE for the associated frame triggering for reception, PDUs in non-received areas (PDU offset > actual L-SDU length) shall not be indicated to upper layer(s).]

[SWS_FrIf_05129] [If Transmission with Immediate Buffer Access is used, only one PDU is allowed per FlexRay Frame (L-SDU).]

Note: Therefore, PDU update bits can be omitted for Transmission with Immediate Buffer Access.

7.6.2 Dynamic PDU length

[SWS_FrIf_05093] [In case the parameter 'FrIfAllowDynamicLSduLength' (see FrIf06049) is set to true for the associated frame triggering, the FrIf module passes the actual used L-PDU length to the driver (Fr_TransmitTxLPdu()), taking into account the following parameters for each PDU:

- the position of the PDU within the L-PDU
- the position of the update-bit information (if configured)

If `FrIfImmediate` equals `TRUE`, the actual length of the respective PDU shall be as passed via `FrIf_Transmit()`.

If `FrIfImmediate` equals `FALSE`, the actual length of the respective PDU shall be as passed via `LSduR_FrIfTriggerTransmit().`]

Note: If `FrIfAllowDynamicLSduLength` is set to `false`, the `FrIf` module just passes the length information according to the frame construction plan to the FlexRay driver.

[SWS_FrIf_05094] [The `FrIf` shall only indicate PDUs in received areas (PDU offset $\leq$ actual L-PDU length) to upper layer(s).]

7.6.3 AlwaysTransmit

Note: According to [7, FlexRay Communications System Protocol Specification V2.1], a FlexRay CC might only support the so-called "continuous transmission mode" where a message is transmitted continuously until the host explicitly invalidates the transmit buffer. If such a FlexRay CC is being used for transmission, and the receiving `FrIf` should still be able to determine which of the PDUs in a received FlexRay Frame have actually been updated by an upper layer BSW module on the transmitter side, a special mechanism is needed in the transmitting `FrIf`, called `AlwaysTransmit` (configuration parameter `FrIfAlwaysTransmit`, see `ECUC_FrIf_06050`). If `AlwaysTransmit` is enabled for an L-PDU that is transmitted using the Communication Operation `DECOUPLED_TRANSMISSION`, the FlexRay Driver's API service `Fr_TransmitTxLPdu()` is always called for this L-PDU, independent from any PDUs in this L-PDU having been updated by an upper layer BSW module. This enables resetting the PDU update bits in the FlexRay CC's transmit buffer, even if none of the PDUs in the FlexRay Frame have actually been updated by an upper layer BSW module, and thus ensures the correct interpretation of the received Frame contents by the receiving `FrIf`.

Note: Since:

- in general, the transmit mode of a FlexRay CC can be configured ("continuous mode" / "single shot mode"), and
- `AlwaysTransmit` can be configured independently per L-PDU, and
- update bits can be configured independently per PDU,

the `FrIf` module can be tailored to exhibit exactly the behavior required by a certain use case,

however, it is the responsibility of the System Designer to select the correct configuration of all these parameters. An incorrect configuration will lead to undesired results.

7.6.4 Realization of the Time-Driven FlexRay Schedule

According to [7, FlexRay Communications System Protocol Specification V2.1], a FlexRay CC is not required to provide mechanisms in hardware to ensure asynchronous access to its transmit and receive buffers e.g. by providing shadow buffers that may be accessed asynchronously by the AUTOSAR FlexRay software modules.

[SWS_FrIf_05130] [The FrIf module shall call all functions accessing the transmit and receive buffers (i.e. performing data transmission or reception, respectively) synchronously (i.e. synchronized to the FlexRay Global Time)]

Rationale for SWS_FrIf_05130: The access of FrIf module functions to transmit and receive buffers only at well-defined points in time ³ avoids concurrent access to the buffers by the hardware and the software.

Note: In order to provide this necessary synchronicity, the FrIf module defines for each Cluster a FlexRay Job List [Configuration Parameter FrIfJobList, see FrIf05367].

The Cluster's FlexRay Job List is executed by its Job List Execution Function (see [Chapter 8.5.1](#)) using an absolute timer [Configuration Parameter FrIfAbsTimerRef, see FrIf06063] of a FlexRay CC connected to the respective Cluster.

7.6.4.1 FlexRay Job List

[SWS_FrIf_05131] [Definition: A FlexRay Job List is a list of (maybe different) Communication Jobs sorted according to their respective execution start time.

Each Communication Job [Configuration Parameter FrIfJob, see FrIf05368] contains the following properties:

- Job start time by means of
 - FlexRay Communication Cycle [Configuration Parameter FrIfCycle, see FrIf06064]
 - Macrotick Offset within the Communication Cycle [Configuration Parameter FrIfMacrotick, see FrIf06065].
- A list of Communication Operations [Configuration Parameter FrIfCommunicationOperation, see FrIf05369] sorted according to a configurable Communication operation index [Configuration Parameter FrIfCommunicationOperationIdx, see FrIf06068]. The sorting order defines the order of execution of the Communication Operations within a FlexRay Communication Job.

]

³In FlexRay Global Time

[SWS_Frlf_05133] [The Frlf module shall call the respective Cluster's FlexRay Job List Execution Function to execute each FlexRay Communication Job at the execution start time assigned to that Communication Job]

[SWS_Frlf_05134] [The Frlf module shall process the actions determined by the Communication Operations assigned to each FlexRay Communication Job]

Each Communication Operation (see Frlf05369) contains the following properties:

- Communication Operation Index [Configuration Parameter FrlfCommunicationOperationIdx, see ECUC_Frlf_06068], which determines the execution order of the Communication Operations.
- Communication Action [Configuration Parameter FrlfCommunicationAction, see Frlf06067], which specifies the actual action to perform
 - DECOUPLED_TRANSMISSION
 - TX_CONFIRMATION
 - RECEIVE_AND_STORE
 - RX_INDICATION
 - RECEIVE_AND_INDICATE
 - PREPARE_LPDU
- A reference to a frame triggering (L-PDU) which is associated with the Communication Action to perform [Configuration parameter FrlfLPduldx, see Frlf06058]⁴.

]

7.6.4.2 FlexRay Job List Execution Function

Since the Communication Schedule of each FlexRay Cluster is independent, there is one dedicated FlexRay Job List and one dedicated FlexRay Job List Execution Function for each FlexRay Cluster that is controlled by the FlexRay Interface.

The Copy Operation into/from the FlexRay CCs are scheduled within the FlexRay Job Lists' communication operations

[SWS_Frlf_05136] [The API names of the FlexRay Job List Execution Functions shall obey the following pattern:

⁴The LPDU is identified by a LPdu Index, which has a 1:1 association to a frame triggering for historical reasons. To obtain compatibility this configuration structure is not changed here. The L-PDU index is identified with a zero-based and dense index, which shall be used as the parameter Fr_LPduldx passed to the AUTOSAR FlexRay Driver when processing LPdus.

FrIf_JobListExec_<FrIfCluster.ShortName> where FrIfCluster.ShortName is the Short Name of the corresponding FrIfCluster.]

[SWS_FrIf_05137] [The FlexRay Job List Execution Function shall execute the Cluster's FlexRay Job List Jobs synchronously to the Cluster's global time (i.e. at well-defined points in time).]

[SWS_FrIf_05138] [Upon invocation, the FlexRay Job List Execution Function shall perform the following steps:

1. Retrieve the FlexRay Global Time from the FlexRay CC providing the Cluster's absolute timer interrupt.
2. If the FlexRay Global Time cannot be retrieved or the global time delay compared to the jobs start time is larger than a maximum delay [Configuration Parameter FrIfMaxIsrDelay, see FrIf06004], the execution of the FlexRay Job List is considered to be asynchronous to the FlexRay Global Time and thus the following actions are performed:
 - Either set a flag (JobListAsyncFlag) indicating that the execution of the FlexRay Job List of this Cluster is asynchronous or directly resynchronize the Joblist as described in SWS_FrIf_95120
 - If the JobListAsyncFlag was set, call the Runtime error FRIF_E_JLE_SYNC
 - Disable absolute Timer Interrupt
 - Terminate the execution of this FlexRay Job.

Otherwise, the FlexRay Job List Execution Function continues with step 3.

3. Retrieve the ordered list of Communication Operations of the current Job pointed to by the current job-pointer.
4. Forward the current job-pointer to the next job-list entry. If the job-pointer was pointed at the end of the job-list, wrap around and set it to the first job-list entry.
5. Retrieve the execution start time of the job marked by the job-pointer and set the absolute timer to this job's start time in order to invoke the FlexRay Job List Execution Function again.
6. Execute the retrieved Communication Operations.

]

Note: In order to keep the runtime of the JLEF short, it is acceptable to implement the described functionality of the JLEF into a separate, high priority task which has to be activated immediately in the JLEF.

7.6.5 Communication Operations

This chapter describes each Communication Operation that is executed within the Job List Execution Function.

7.6.5.1 TransmitWithDecoupledBufferAccess

[SWS_FrIf_05058]

Upstream requirements: [SRS_Fr_05130](#)

[The FrIf module shall be capable of Transmit Request queuing by using the TrigTx Counter.]

Note: Only the amount of transmit requests are stored, not the data itself.

[SWS_FrIf_05063]

Upstream requirements: [SRS_Fr_05027](#)

[If the related CC is in FrIf_State FRIF_STATE_ONLINE for a Communication Operation DECOUPLED_TRANSMISSION, then the Job List Execution Function shall execute this Communication Operation. Otherwise, the Job List Execution Function shall ignore this Communication Operation.]

[SWS_FrIf_05287] [For a Communication Operation DECOUPLED_TRANSMISSION the Job List Execution Function shall perform the following steps

1. Iterate over all PDUs contained in the FrIfFrameStructure (see FrIf05370) of the associated frame triggering of this Communication Operation and
 - (a) Check whether TrigTxCounter is > 0 or FrIfNoneMode == true for the PDU. If not, clear the update-bit for this PDU [Configuration Parameter FrIfPdu UpdateBitOffset, see FrIf06071] and proceed with the next PDU, otherwise continue with the following steps:
 - i. Decrement TrigTxCounter only if TrigTxCounter > 0. If the value of TrigTxCounter = 0, do not decrement.
 - ii. Call the upper layer's function LSduR_FrIfTriggerTransmit() with the associated PDUId (defined by the upper layer) and pass a pointer to a temporary buffer within the FrIf that assembles the L-SDU. The pointer shall consider the byte offset [Configuration Parameter FrIfPduOffset, see FrIf06070]] of the PDU within the frame. If LSduR_FrIfTriggerTransmit() returns E_NOT_OK, the TrigTxCounter value has to be rolled back to the previous value.
 - iii. Remember that a transmission for this PDU is pending if a transmission confirmation is needed for this PDU [Configuration Parameter FrIfConfirm, see FrIf06075] increment TxConfCounter, where the maximum value is limited by static configuration [Configuration Parameter FrIf

CounterLimit, see FrIf06076]. If the FrIfCounterLimit has been reached, the FrIfCounterLimit value is kept and not incremented any more.

- iv. Set the update-bit if configured for this PDU [Configuration Parameter FrIfPduUpdateBitOffset, see FrIf06071]. In case the API LSduR_FrIfTriggerTransmit() does not return E_OK, or the API FrIf_CancelTransmit() for the corresponding PDU has been called, reset the update-bit to "not updated".
2. If at least one PDU was requested for transmission or for at least one PDU FrIfNoneMode == true and LSduR_FrIfTriggerTransmit() returned E_OK or the frame is configured to be always transmitted [Configuration Parameter FrIfAlwaysTransmit == true] then the FlexRay Driver's API service Fr_TransmitTxLPdu() is called:
 - (a) Fr_CtrlIdx is derived according to the indexing scheme described in chapter of indexing.
 - (b) Fr_LPdulIdx is set to the configured L-PDU index [Configuration Parameter
 - (c) FrIfLPdulIdx, see FrIf06058] associated with the Communication Operation
 - (d) Fr_LSduPtr is set to the temporary FrIf L-SDU assembling buffer.
 - (e) Fr_LSduLength is set to the L-SDU length [Configuration Parameter FrIfLSduLength, see FrIf06054]
 - (f) Fr_SlotAssignmentPtr is set to a temporary slot assignment buffer if Bus Mirroring is enabled globally (see FrIfBusMirroringSupport), otherwise to the NULL_PTR.
 3. If Bus Mirroring is enabled globally (see FrIfBusMirroringSupport) and has been activated with a call to FrIf_EnableBusMirroring() for the Fr_CtrlIdx and Fr_TransmitTxLPdu() returned E_OK (indicating that the transmission succeeded), call Mirror_ReportFlexRayFrame() with "controllerId" set to Fr_CtrlIdx, "slotId", "cycle", and "channel" taken from Fr_SlotAssignmentPtr, "frame" constructed from Fr_LSduPtr and Fr_LSduLength, and "txConflict" set to false.
 4. In case the Driver's API Fr_TransmitTxLPdu() returned E_NOT_OK (indicating that the transmission failed) changes on TrigTxCounter and TxConfCounter must be rolled back (see 4. and 5.) for each PDU contained in the FlexRay L-SDU.

]

Note: All described actions in SWS_FrIf_05287 are depicted in detail in the sequence chart in [Chapter 9.1.2](#).

[SWS_FrIf_05435] [If FrIfAllowDynamicLSduLength exists and is set to TRUE for the associated frame triggering, the actual L-SDU length, that is passed to the driver by calling Fr_TransmitTxLPdu(), shall be determined (i.e. shortened as much as possi-

ble) by taking only those PDUs into account, which have been indicated via LSduR_FrlfTriggerTransmit() and consider the following points:

- the position of the respective PDU within the L-SDU
- the actual length of the respective PDU as passed via LSduR_FrlfTriggerTransmit()

」

[SWS_Frlf_05436] 「A shortened L-Sdu (see [SWS_Frlf_05435]) shall always contain all configured update bits.」

Note: [SWS_Frlf_05435] and [SWS_Frlf_05436] ensure that on one hand all the needed information for disassembling the L-SDU is available on receiver side (PDU(s) itself and the corresponding update-bit(s) if configured), and on the other hand that the payload can be reduced as much as possible by taking the position of all the required data for disassembling contained in the frame construction plan into account when shortening the L-SDU to be passed to the driver.

7.6.5.2 ProvideTxConfirmation

This Communication Operation provides a Tx confirmation and optionally checks the occurrence of a Tx conflict.

[SWS_Frlf_05064] 「If the related CC is in Frlf_State FRIF_STATE_ONLINE for a Communication Operation TX_CONFIRMATION, then the Job List Execution Function shall execute this Communication Operation. Otherwise, the Job List Execution Function shall ignore this Communication Operation.」

[SWS_Frlf_05288] 「"For a Communication Operation TX_CONFIRMATION the Job List Execution Function shall perform the following steps:

1. Call the FlexRay Driver's API function Fr_CheckTxLPduStatus():
 - Fr_CtrlIdx is derived according to the indexing scheme described in chapter of indexing
 - Fr_LPdulIdx is set to the configured L-PDU buffer index [Configuration Parameter FrlfLPdulIdx, see Frlf06058] associated with the Communication Operation.
 - Fr_SlotAssignmentPtr is set to a temporary slot assignment buffer if Bus Mirroring is enabled globally (see FrlfBusMirroringSupport), otherwise to the NULL_PTR.
2. If the transmission was performed (output parameter *Fr_TxLPduStatusPtr is set to FR_TRANSMITTED) then iterate over all PDUs contained in the FrlfFrame

Structure (see Frlf05370) of the associated frame triggering. If TxConfCounter for a PDU is 0 proceed with the next PDU, otherwise

- If FrlfConfirm == true, call the upper layer's function LSduR_FrlfTxConfirmation(E_OK) with the associated PDUID (defined by the upper layer).
 - If FrlfConfirm == true , decrement TxConfCounter.
3. If the transmission was performed but a TxConflict occurred (output parameter *Fr_TxLPduStatusPtr is set to FR_TRANSMITTED_CONFLICT) then iterate over all PDUs contained in the FrlfFrameStructure (see Frlf05370) of the associated frame triggering. If TxConfCounter for a PDU is 0 proceed with the next PDU, otherwise
 - If FrlfConfirm == true, call the upper layer's function <UL_TxConfirmation(E_NOT_OK)> with the associated PDUID (defined by the upper layer).
 - If FrlfConfirm == true , decrement TxConfCounter.
 4. If Bus Mirroring is enabled globally (see FrlfBusMirroringSupport) and has been activated with a call to Frlf_EnableBusMirroring() for the Fr_CtrlIdx and the API Fr_CheckTxLpduStatus() returns "FR_TRANSMITTED_CONFLICT", call Mirror_ReportFlexRayFrame() with "controllerId" set to Fr_CtrlIdx, "slotId", "cycle", and "channel" taken from Fr_SlotAssignmentPtr, "frame" set to the NULL_PTR, and "txConflict" set to true.

If the API Fr_CheckTxLpduStatus() returns "FR_TRANSMITTED_CONFLICT" and the <UL_TxConflictNotification> is configured via FrlfTxConflictNotificationName (ECUC_Frlf_06122), call this function for the same LPduIdx.]

7.6.5.3 ReceiveAndStore

[SWS_Frlf_05289] [If the related CC is in Frlf_State FRIF_STATE_ONLINE for a Communication Operation RECEIVE_AND_STORE, then the Job List Execution Function shall execute this Communication Operation. Otherwise, the Job List Execution Function shall ignore this Communication Operation.]

[SWS_Frlf_05290] [For a Communication Operation RECEIVE_AND_STORE the Job List Execution Function shall perform the following steps:

1. Call the FlexRay Driver's API function Fr_ReceiveRxLPdu():
 - (a) Fr_CtrlIdx is derived according to the indexing scheme described in chapter of indexing.
 - (b) Fr_LPduIdx is set to the configured L-PDU index [Configuration Parameter FrlfLPduIdx, see Frlf06058] associated with the Communication Operation.
 - (c) Fr_LSduPtr is set to a temporary buffer.

- (d) Fr_SlotAssignmentPtr is set to a temporary slot assignment buffer if Bus Mirroring is enabled globally (see FrIfBusMirroringSupport), otherwise to the NULL_PTR.
- 2. If Bus Mirroring is enabled globally (see FrIfBusMirroringSupport) and has been activated with a call to FrIf_EnableBusMirroring() for the Fr_CtrlIdx and an L-PDU was received (Output parameter *Fr_LPduStatusPtr != FR_NOT_RECEIVED), call Mirror_ReportFlexRayFrame() with "controllerId" set to Fr_CtrlIdx, "slotId", "cycle", and "channel" taken from Fr_SlotAssignmentPtr, "frame" constructed from Fr_LSduPtr and Fr_LSduLengthPtr, and "txConflict" set to false.
- 3. If a L-PDU was received (Output parameter *Fr_LPduStatusPtr != FR_NOT_RECEIVED) iterate over all PDUs contained in the FrIfFrameStructure (see FrIf05370) of the associated frame triggering and:
 - (a) If an update bit was configured for the PDU [Configuration Parameter FrIfPduUpdateBitOffset, see FrIf06071] and the update bit for the PDU is not set, continue with the next PDU. Otherwise,
 - (b) Copy the PDU Payload from the temporary buffer considering the PDU offset within the L-SDU [Configuration Parameter FrIfPduOffset, see FrIf06070] into a FrIf PDU-related static buffer.
 - (c) Store the actual received PDU length
 - (d) Mark the PDU-related static buffer as up-to-date.
- 4. if *Fr_LPduStatusPtr == FR_RECEIVED_MORE_DATA_AVAILABLE restart at number 1 again. Otherwise the communication operation has finished.

]

7.6.5.4 ProvideRxIndication

[SWS_FrIf_05062]

Upstream requirements: [SRS_Fr_05170](#)

[If the related CC is in FrIf_State FRIF_STATE_ONLINE for a Communication Operation RX_INDICATION, then the Job List Execution Function shall execute this Communication Operation. Otherwise, the Job List Execution Function shall ignore this Communication Operation.]

[SWS_FrIf_05291] [For a Communication Operation RX_INDICATION the Job List Execution Function shall perform the following steps:

1. Iterate over all PDU-related static buffers of PDUs contained in the FrIfFrameStructure (see FrIf05370) of the associated frame triggering
2. If the PDU-related static buffer is marked as outdated, continue with the next PDU. Otherwise if the buffer is marked up-to-date,

- (a) Call the upper layer's function `LSduR_FrlfRxIndication()` with the PDU Id the receiving module expects and `PduInfoPtr` which contains the received data address and received data length.
- (b) Mark the PDU-related static buffer as outdated.

]

7.6.5.5 ReceiveAndIndicate

[SWS_Frlf_05292] [If the related CC is in `Frlf_State FRIF_STATE_ONLINE` for a Communication Operation `RECEIVE_AND_INDICATE`, then the Job List Execution Function shall execute this Communication Operation. Otherwise, the Job List Execution Function shall ignore this Communication Operation.]

[SWS_Frlf_05293] [For a Communication Operation `RECEIVE_AND_INDICATE` the Job List Execution Function shall perform the following steps:

1. Calculate values for input parameters:
 - (a) `Fr_CtrlIdx` is derived according to the indexing scheme described in chapter of indexing.
 - (b) `Fr_LPduIdx` is set to the configured L-PDU index [Configuration Parameter `FrlfLPduIdx`, see `Frlf06058`] associated with the Communication Operation.
 - (c) `Fr_LSduPtr` is set to a temporary buffer.
 - (d) `Fr_SlotAssignmentPtr` is set to a temporary slot assignment buffer if Bus Mirroring is enabled globally (see `FrlfBusMirroringSupport`), otherwise to the `NULL_PTR`.
2. Initialize `ComOpLoopCounter` to 0.
3. As long as `ComOpLoopCounter < FrlfRxComOpMaxLoop` do
 - (a) Call `Fr_ReceiveRxLPdu` with the parameters calculated in 1)
 - (b) If `*Fr_LPduStatusPtr != FR_NOT_RECEIVED` then continue at 3)c), otherwise the communication operation has finished.
 - (c) If Bus Mirroring is enabled globally (see `FrlfBusMirroringSupport`) and has been activated with a call to `Frlf_EnableBusMirroring()` for the `Fr_CtrlIdx`, call `Mirror_ReportFlexRayFrame()` with "controllerId" set to `Fr_CtrlIdx`, "slotId", "cycle", and "channel" taken from `Fr_SlotAssignmentPtr`, "frame" constructed from `Fr_LSduPtr` and `Fr_LSduLengthPtr`, and "txConflict" set to false.

Otherwise, continue at 3)d).

- (d) For each Pdu contained in the `FrIfFrameStructure` (see `FrIf05370`) of the associated frame triggering do

-If an update bit was configured for the PDU [Configuration Parameter `FrIfPduUpdateBitOffset`, see `FrIf06071`] and the update bit for the PDU is not set, continue with the next PDU. Otherwise

-Call the upper layer's function `LSduR_FrIfRxIndication()` with the PDU Id the receiving module expects and a pointer to the Pdu-Info structure containing the Pdu length and a reference to the temporary buffer considering the PDU offset within the L-SDU [Configuration Parameter `FrIfPduOffset`, see `FrIf06070`] as parameters.

- (e) if `*Fr_LPduStatusPtr == FR_RECEIVED_MORE_DATA_AVAILABLE` then increment `ComOpLoopCounter` and restart at 3)a), otherwise the communication operation has finished.

]

7.6.5.6 PREPARE_LPDU

The Communication Operation `PREPARE_LPDU` enables hardware optimization purposes (hardware buffer re-configuration)

[SWS_FrIf_05294] [The Communication Operation `PREPARE_LPDU` performs the following steps:

1. Call the FlexRay Driver's API function `Fr_PrepareLPdu()`:

- `Fr_CtrlIdx` is derived according to the indexing scheme described in chapter of indexing.
- `Fr_LPdulIdx` is set to the configured L-PDU index [Configuration Parameter `FrIfLPdulIdx`, see `FrIf06058`] associated with the Communication Operation.

]

[SWS_FrIf_05061]

Upstream requirements: [SRS_Fr_05042](#)

[The Communication Operation `PREPARE_LPDU` enables hardware optimization purposes. Its purpose is to enable certain FlexRay CC hardware resources (e.g. a CC's message buffer) to be prepared (configured) for the transmission/reception of a certain L-PDU.

This Communication Operation enables the FlexRay Driver to optimize the usage of hardware resources if available at appropriate point of times. However, it is the responsibility of the FlexRay Driver to decide and validate resource allocation optimizations

based on the PREPARE_LPDU Communication Operations. Practically the usage of this Communication Operation will introduce some runtime-overhead even if the Flex Ray Driver does not use the opportunity for reconfiguration.]

7.6.5.7 FREE_OP_A

User-defined communication operation in order to support hardware specific or additional communication controller features to increase performance. Use cases are communication controllers with serial connection or DMA-transfers.

7.6.5.8 FREE_OP_B

User-defined communication operation in order to support hardware specific or additional communication controller features to increase performance. Use cases are communication controllers with serial connection or DMA-transfers.

7.6.6 Transmission with Immediate Buffer Access

[SWS_Frlf_15295] [The FlexRay Job List Execution Function does not initiate transmission with immediate buffer access. Instead, the actions described here are carried out in the context of the Frlf_Transmit() API service, which in turn is called by an upper layer BSW module.]

[SWS_Frlf_05295] [The FlexRay Interface shall perform a PDU transmission with immediate buffer access (see 9.1), only if the following restriction regarding static configuration apply:

- The PDU must be the only PDU in a FlexRay Frame (L-SDU). It is not packed into a FlexRay Frame together with other PDUs (i.e., the mapping between this PDU and the respective L-SDU is a 1:1 association).
- The PDU must be located at the beginning of the L-SDU.
- There is no update-bit for immediate PDUs configured.

]

[SWS_Frlf_05296] [If an upper layer module calls Frlf_Transmit() with TxPduId being configured for an immediate PDU, the AUTOSAR module FlexRay Interface shall perform the following steps for an immediate PDU transmission within the context of the Frlf_Transmit() API service Driver's API function Fr_TransmitTxLPdu():

- Fr_CtrlIdx is derived according to the indexing scheme described in chapter of indexing.

- Fr_LPdulIdx is set to the configured L-PDU index [Configuration Parameter FrLf_LPdulIdx, see FrLf06058] associated with the TxPdul.
- Fr_LSduPtr is set to the Pdu Payload pointer contained in the PduInfoPtr passed as parameter to FrLf_Transmit.
- If the parameter FrLfAllowDynamicLSduLength=TRUE, the actual length of the respective PDU shall be as passed via FrLf_Transmit().
- Fr_SlotAssignmentPtr is set to a temporary slot assignment buffer if Bus Mirroring is enabled globally (see FrLfBusMirroringSupport), otherwise to the NULL_PTR.

In case the Driver's API Fr_TransmitTxLPdu() returned E_OK (indicating that the transmission request succeeded) the TxConfCounter is incremented for the respective PDU. The maximum value of TxConfCounter is limited by static configuration [Configuration Parameter FrLfCounterLimit, see FrLf06076]. If Bus Mirroring is enabled globally (see FrLfBusMirroringSupport) and has been activated with a call to FrLf_EnableBusMirroring() for the Fr_CtrlIdx, call Mirror_ReportFlexRayFrame() with "controllerId" set to Fr_CtrlIdx, "slotId", "cycle", and "channel" taken from Fr_SlotAssignmentPtr, "frame" constructed from Fr_LSduPtr and Fr_LSduLength, and "txConflict" set to false.

In case the Driver's API Fr_TransmitTxLPdu() returned E_NOT_OK do not modify the current counter value of TxConfCounter.]

7.7 Error Classification

Chapter [2, General Specification of Basic Software Modules] 7.2 “Error Handling” describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

7.7.1 Development Errors

[SWS_FrLf_05145] Definition of development errors in module FrLf [

Type of error	Related error code	Error value
Invalid pointer	FRIF_E_PARAM_POINTER	0x01
Invalid Controller index	FRIF_E_INV_CTRL_IDX	0x02
Invalid Cluster index	FRIF_E_INV_CLST_IDX	0x03
Invalid Channel index	FRIF_E_INV_CHNL_IDX	0x04
Invalid timer index	FRIF_E_INV_TIMER_IDX	0x05
Invalid FrLf_TxPdu Index	FRIF_E_INV_TXPDUID	0x06
Invalid LPdu Index	FRIF_E_INV_LPDU_IDX	0x07





Type of error	Related error code	Error value
Frlf not initialized	FRIF_E_UNINIT	0x08
Invalid state requested	FRIF_E_INV_FRIF_STATE	0x0A
Invalid Frame ID	FRIF_E_INV_FRAME_ID	0x0B
Initialization failed	FRIF_E_INIT_FAILED	0x0C
Invalid Pdu length	FRIF_E_INV_PDULENGTH	0x0D

]

7.7.2 Runtime Errors

[SWS_Frlf_05432] Definition of runtime errors in module Frlf [

Type of error	Related error code	Error value
Job List Execution lost synchronization to the Flex Ray Global Time	FRIF_E_JLE_SYNC	0x01

]

7.7.3 Production Errors

[SWS_Frlf_05146] [For "Definition of Production Errors" see [\[SWS_Frlf_05320\]](#)]

[SWS_Frlf_05320] Definition of Production Errors [

Type or error	Related error code	Value [hex]
error detection in NIT on channel A	FRIF_E_NIT_CH_A	Assigned by DEM
error detection in NIT on channel B	FRIF_E_NIT_CH_B	Assigned by DEM
error detection in SW on channel A	FRIF_E_SW_CH_A	Assigned by DEM
error detection in SW on channel B	FRIF_E_SW_CH_B	Assigned by DEM
error detection in ACS on channel A	FRIF_E_ACS_CH_A	Assigned by DEM
error detection in ACS on channel B	FRIF_E_ACS_CH_B	Assigned by DEM

]

[SWS_Frlf_05426] [For "Error Detection in NIT channel A" see [\[SWS_Frlf_05321\]](#)]

[SWS_Frlf_05321] Error detection in NIT on channel A

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	FRIF_E_NIT_CH_A
Description	This production error shall be issued when an error in NIT on channel A was detected.
Failed condition	The channel status information shall be reported to DEM as Dem_SetEvent Status (FRIF_E_NIT_CH_A, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel A NIT status data vSSISyntaxError, vSSIBviolation) is set [SWS_Frlf_35120] .
Passed condition	The channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_NIT_CH_A, DEM_EVENT_STATUS_PREPASSED) when none of the error bits of a single controller (Channel A NIT status data vSSISyntaxError, vSSIBviolation) is set [SWS_Frlf_35120] .

]

[SWS_Frlf_05427] [For "Error Detection in NIT channel B" see [\[SWS_Frlf_05322\]](#)]

[SWS_Frlf_05322] Error detection in NIT on channel B

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	FRIF_E_NIT_CH_B
Description	This production error shall be issued when an error in NIT on channel B was detected.
Failed condition	The channel status information shall be reported to DEM as Dem_SetEvent Status (FRIF_E_NIT_CH_B, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel B NIT status data vSSISyntaxError, vSSIBviolation) is set [SWS_Frlf_45120] .
Passed condition	The channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_NIT_CH_B, DEM_EVENT_STATUS_PREPASSED) when none of the error bits of a single controller (Channel B NIT status data vSSISyntaxError, vSSIBviolation) is set [SWS_Frlf_45120] .

]

[SWS_Frlf_05428] [For "Error detection in SW on channel A" see [\[SWS_Frlf_05323\]](#)]

[SWS_Frlf_05323] Error detection in SW channel A

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	FRIF_E_SW_CH_A
Description	This production error shall be issued when an error in SW on channel A was detected.
Failed condition	The channel status information shall be reported to DEM as Dem_SetEvent Status (FRIF_E_SW_CH_A, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel A symbol window status data vSSISyntaxError, vSSIBviolation) is set [SWS_Frlf_55120] .
Passed condition	The channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_SW_CH_A, DEM_EVENT_STATUS_PREPASSED) when none of the error bits of a single controller (Channel A symbol window status data vSSISyntaxError, vSSIBviolation) is set [SWS_Frlf_55120] .

]

[SWS_Frlf_05429] [For "Error detection in SW on channel B" see [SWS_Frlf_05324]]

[SWS_Frlf_05324] Error detection in SW channel B

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	FRIF_E_SW_CH_B
Description	This production error shall be issued when an error in SW on channel B was detected.
Failed condition	The channel status information shall be reported to DEM as Dem_SetEvent Status (FRIF_E_SW_CH_B, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel B symbol window status data vSS!SyntaxError, vSS!Bviolation) is set [SWS_Frlf_65120].
Passed condition	The channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_SW_CH_B, DEM_EVENT_STATUS_PREPASSED) when none of the error bits of a single controller (Channel B symbol window status data vSS!SyntaxError, vSS!Bviolation) is set [SWS_Frlf_65120].

]

[SWS_Frlf_05431] [For "Error detection in ACS on channel A" see [SWS_Frlf_05325]]

[SWS_Frlf_05325] Error detection in ACS on channel A

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	FRIF_E_ACS_CH_A
Description	This production error shall be issued when an error in ACS on channel A was detected.
Failed condition	The channel status information shall be reported to DEM as Dem_SetEvent Status (FRIF_E_ACS_CH_A, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel A aggregated channel status data vSS!SyntaxError, vSS!ContentError, vSS!Bviolation, vSS!TxConflict) is set [SWS_Frlf_75120].
Passed condition	The channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_ACS_CH_A, DEM_EVENT_STATUS_PREPASSED) when none of the error bits of a single controller (Channel A aggregated channel status data vSS!SyntaxError, vSS!ContentError, vSS!Bviolation, vSS!TxConflict) is set [SWS_Frlf_75120].

]

[SWS_Frlf_05430] [For "Error detection in ACS on channel B" see [SWS_Frlf_05326]]

[SWS_Frlf_05326] Error detection in ACS on channel B

Upstream requirements: [SRS_BSW_00458](#)

[

Diagnostic Event (Error Name)	FRIF_E_ACS_CH_B
Description	This production error shall be issued when an error in ACS on channel B was detected.
Failed condition	The channel status information shall be reported to DEM as Dem_SetEvent Status (FRIF_E_ACS_CH_B, DEM_EVENT_STATUS_PREFAILED) when any of the error bits of a single controller (Channel B aggregated channel status data vSS!SyntaxError, vSS!ContentError, vSS!Bviolation, vSS!TxConflict) is set [SWS_Frlf_85120].





Passed condition	The channel status information shall be reported to DEM as Dem_SetEventStatus (FRIF_E_ACS_CH_B, DEM_EVENT_STATUS_PREPASSED) when none of the error bits of a single controller (Channel B aggregated channel status data vSS!Syntax Error, vSS!ContentError, vSS!Bviolation, vSS!TxConflict) is set [SWS_FrIf_85120].
-------------------------	---

」

7.7.4 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

In this chapter all types included from the following files are listed.

[SWS_FrIf_05001] Definition of imported datatypes of module FrIf

Upstream requirements: [SRS_BSW_00348](#), [SRS_BSW_00353](#), [SRS_BSW_00304](#), [SRS_BSW_00378](#)

Module	Header File	Imported Type
Comtype	ComStack_Types.h	PduIdType
	ComStack_Types.h	PduInfoType
	ComStack_Types.h	PduLengthType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Fr	Fr_GeneralTypes.h	Fr_ChannelType
	Fr_GeneralTypes.h	Fr_ErrorModeType
	Fr_GeneralTypes.h	Fr_POCSStateType
	Fr_GeneralTypes.h	Fr_POCSStatusType
	Fr_GeneralTypes.h	Fr_RxLPduStatusType
	Fr_GeneralTypes.h	Fr_SlotAssignmentType
	Fr_GeneralTypes.h	Fr_SlotModeType
	Fr_GeneralTypes.h	Fr_StartupStateType
	Fr_GeneralTypes.h	Fr_TxLPduStatusType
	Fr_GeneralTypes.h	Fr_WakeupStatusType
FrTrcv	Fr_GeneralTypes.h	FrTrcv_TrvcModeType
	Fr_GeneralTypes.h	FrTrcv_TrvcWUReasonType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

8.2 Type definitions

This chapter lists the data types that the FlexRay Interface defines.

8.2.1 Frlf_ConfigType

[SWS_Frlf_05301] Definition of datatype Frlf_ConfigType [

Name	Frlf_ConfigType		
Kind	Structure		
Elements	Implementation specific		
	Type	–	
	Comment	–	
Description	This type contains the implementation-specific post build time configuration structure. Only pointers of this type are allowed.		
Available via	Frlf.h		

]

8.2.2 Frlf_StateType

[SWS_Frlf_05755] Definition of datatype Frlf_StateType [

Name	Frlf_StateType		
Kind	Enumeration		
Range	FRIF_STATE_OFFLINE	–	The FlexRay CC is not ready for communication, the FlexRay cluster is not synchronized.
	FRIF_STATE_ONLINE	–	The FlexRay CC is ready for communication, the FlexRay cluster is synchronized.
Description	Variables of this type are used to represent the Frlf_State of a FlexRay CC.		
Available via	Frlf.h		

]

8.2.3 Frlf_StateTransitionType

[SWS_Frlf_05303] Definition of datatype Frlf_StateTransitionType [

Name	Frlf_StateTransitionType		
Kind	Enumeration		
Range	FRIF_GOTO_OFFLINE	–	Literal for requesting transition into FRIF_STATE_OFFLINE
	FRIF_GOTO_ONLINE	–	Literal for requesting transition into FRIF_STATE_ONLINE state.
Description	Variables of this type are used to represent the Frlf_State of a FlexRay CC.		
Available via	Frlf.h		

]

8.3 Function definitions

This is a list of API services (functions) the FrIf module provides to upper layer BSW modules.

8.3.1 FrIf_Init

[SWS_FrIf_05003] Definition of API function FrIf_Init

Upstream requirements: [SRS_BSW_00405](#), [SRS_BSW_00101](#), [SRS_BSW_00358](#), [SRS_BSW_00414](#), [SRS_Fr_05013](#)

Service Name	FrIf_Init	
Syntax	<pre>void FrIf_Init (const FrIf_ConfigType* FrIf_ConfigPtr)</pre>	
Service ID [hex]	0x02	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	FrIf_ConfigPtr	Base pointer to the configuration structure of the FlexRay Interface.
Parameters (inout)	None	
Parameters (out)	None	
Return value	void	–
Description	Initializes the FlexRay Interface.	
Available via	FrIf.h	

Note:

The AUTOSAR ECU StateManager calls this FlexRay Interface API service with the address of the static configuration structure of the FrIf module in parameter FrIf_Config Ptr.

[SWS_FrIf_05156] [The function FrIf_Init shall carry out the following actions:

1. Configure the FlexRay Interface module: initialize the local memory space used to store the PDU data and the PDU properties and state variables and the Flex Ray Interface State Machine.
2. The initialization of the memory space has to make sure that the PDU-related static buffer status is set to "outdated"

8.3.2 FrIf_ControllerInit

[SWS_FrIf_05004] Definition of API function FrIf_ControllerInit

Upstream requirements: [SRS_Fr_05031](#)

[

Service Name	FrIf_ControllerInit	
Syntax	<pre>Std_ReturnType FrIf_ControllerInit (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x03	
Sync/Async	Synchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Initialized a FlexRay CC.	
Available via	FrIf.h	

]

[SWS_FrIf_05158] [If parameter FrIf_CtrlIdx of FrIf_ControllerInit has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_ControllerInit shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05159] [The function FrIf_ControllerInit shall wrap the FlexRay Driver API function Fr_ControllerInit() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Calling Fr_ControllerInit() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05160] [Caveats of FrIf_ControllerInit: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#)]

8.3.3 FrIf_SetAbsoluteTimer

[SWS_FrIf_05021] Definition of API function FrIf_SetAbsoluteTimer [

Service Name	FrIf_SetAbsoluteTimer	
Syntax	<pre>Std_ReturnType FrIf_SetAbsoluteTimer (uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx, uint8 FrIf_Cycle, uint16 FrIf_Offset)</pre>	
Service ID [hex]	0x19	
Sync/Async	Synchronous	
Reentrancy	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_AbsTimerIdx	Index of the absolute timer to address.
	FrIf_Cycle	FlexRay Cycle number to be set.
	FrIf_Offset	Number of Macroticks to be set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_SetAbsoluteTimer().	
Available via	FrIf.h	

]

[SWS_FrIf_05234] [If parameter FrIf_CtrlIdx of FrIf_SetAbsoluteTimer has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetAbsoluteTimer shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05235] [The function FrIf_SetAbsoluteTimer shall wrap This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_SetAbsoluteTimer() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters
3. Fr_AbsTimerIdx to FrIf_AbsTimerIdx
4. Fr_Cycle to FrIf_Cycle
5. Fr_Offset to FrIf_Offset
6. Calling Fr_SetAbsoluteTimer() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_Frlf_05236] [Caveats of Frlf_SetAbsoluteTimer: The FlexRay Interface module has to be initialized with a call of Frlf_Init() before this API service may be called, see [\[SWS_Frlf_05003\]](#).]

8.3.4 Frlf_EnableAbsoluteTimerIRQ

[SWS_Frlf_05025] Definition of API function Frlf_EnableAbsoluteTimerIRQ [

Service Name	Frlf_EnableAbsoluteTimerIRQ	
Syntax	<pre>Std_ReturnType Frlf_EnableAbsoluteTimerIRQ (uint8 Frlf_CtrlIdx, uint8 Frlf_AbsTimerIdx)</pre>	
Service ID [hex]	0x1d	
Sync/Async	Synchronous	
Reentrancy	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	Frlf_CtrlIdx	Index of the FlexRay CC to address.
	Frlf_AbsTimerIdx	Index of the absolute timer to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_EnableAbsoluteTimerIRQ().	
Available via	Frlf.h	

]

[SWS_Frlf_05246] [If parameter Frlf_CtrlIdx of Frlf_EnableAbsoluteTimerIRQ has an invalid value and if development error detection is enabled (i.e. FrlfDevErrorDetect equals ON), the function Frlf_EnableAbsoluteTimerIRQ shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_Frlf_05247] [The function Frlf_EnableAbsoluteTimerIRQ shall wrap the FlexRay Driver API function Fr_EnableAbsoluteTimerIRQ() by:

1. Translating (based on static Frlf module configuration) the FlexRay CC index Frlf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters
 - (a) Fr_AbsTimerIdx to Frlf_AbsTimerIdx
3. Calling Fr_EnableAbsoluteTimerIRQ() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_Frlf_05248] [Caveats of Frlf_EnableAbsoluteTimerIRQ: The FlexRay Interface module has to be initialized with a call of Frlf_Init() before this API service may be called, see [SWS_Frlf_05003].]

8.3.5 Frlf_AckAbsoluteTimerIRQ

[SWS_Frlf_05029] Definition of API function Frlf_AckAbsoluteTimerIRQ [

Service Name	Frlf_AckAbsoluteTimerIRQ	
Syntax	<pre>Std_ReturnType Frlf_AckAbsoluteTimerIRQ (uint8 Frlf_CtrlIdx, uint8 Frlf_AbsTimerIdx)</pre>	
Service ID [hex]	0x21	
Sync/Async	Synchronous	
Reentrancy	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	Frlf_CtrlIdx	Index of the FlexRay CC to address.
	Frlf_AbsTimerIdx	Index of the absolute timer to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_AckAbsoluteTimerIRQ()	
Available via	Frlf.h	

]

[SWS_Frlf_05258] [If parameter Frlf_CtrlIdx of Frlf_AckAbsoluteTimerIRQ has an invalid value and if development error detection is enabled (i.e. FrlfDevErrorDetect equals ON), the function Frlf_AckAbsoluteTimerIRQ shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_Frlf_05259] [The function Frlf_AckAbsoluteTimerIRQ shall wrap the FlexRay Driver API function Fr_AckAbsoluteTimerIRQ() by:

1. Translating (based on static Frlf module configuration) the FlexRay CC index Frlf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters
 - (a) Fr_AbsTimerIdx to Frlf_AbsTimerIdx
3. Calling Fr_AckAbsoluteTimerIRQ() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05260] [Caveats of FrIf_AckAbsoluteTimerIRQ: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.6 FrIf_StartCommunication

[SWS_FrIf_05005] Definition of API function FrIf_StartCommunication

Upstream requirements: [SRS_Fr_05015](#)

[

Service Name	FrIf_StartCommunication	
Syntax	Std_ReturnType FrIf_StartCommunication (uint8 FrIf_CtrlIdx)	
Service ID [hex]	0x04	
Sync/Async	Asynchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_StartCommunication().	
Available via	FrIf.h	

]

[SWS_FrIf_05161] [If parameter FrIf_CtrlIdx of FrIf_StartCommunication has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_StartCommunication shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05162] [The function FrIf_StartCommunication shall wrap the FlexRay Driver API function Fr_StartCommunication() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Calling Fr_StartCommunication() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05163] [Caveats of FrIf_StartCommunication: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#)]

8.3.7 FrIf_HaltCommunication

[SWS_FrIf_05006] Definition of API function FrIf_HaltCommunication

Upstream requirements: [SRS_BSW_00336](#), [SRS_Fr_05063](#)

[

Service Name	FrIf_HaltCommunication	
Syntax	Std_ReturnType FrIf_HaltCommunication (uint8 FrIf_CtrlIdx)	
Service ID [hex]	0x05	
Sync/Async	Asynchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_HaltCommunication().	
Available via	FrIf.h	

]

[SWS_FrIf_05164] [If parameter FrIf_CtrlIdx of FrIf_HaltCommunication has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_HaltCommunication shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05165] [The function FrIf_HaltCommunication shall wrap the FlexRay Driver API function Fr_HaltCommunication() by:

- Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
- Calling Fr_HaltCommunication() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05166] [Caveats of FrIf_HaltCommunication: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#)]

8.3.8 FrIf_AbortCommunication

[SWS_FrIf_05007] Definition of API function FrIf_AbortCommunication

Upstream requirements: [SRS_Fr_05016](#)

[

Service Name	FrIf_AbortCommunication	
Syntax	<pre>Std_ReturnType FrIf_AbortCommunication (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_AbortCommunication().	
Available via	FrIf.h	

]

[SWS_FrIf_05167] [If parameter FrIf_CtrlIdx of FrIf_AbortCommunication has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_AbortCommunication shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05168] [The function FrIf_AbortCommunication shall wrap the FlexRay Driver API function Fr_AbortCommunication() by:

- Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
- Calling Fr_AbortCommunication() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05169] [Caveats of FrIf_AbortCommunication: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#)]

8.3.9 FrIf_GetState

[SWS_FrIf_05170] Definition of API function FrIf_GetState [

Service Name	FrIf_GetState	
Syntax	<pre>Std_ReturnType FrIf_GetState (uint8 FrIf_ClstIdx, FrIf_StateType* FrIf_StatePtr)</pre>	
Service ID [hex]	0x07	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_ClstIdx	Index of the cluster addressed.
Parameters (inout)	None	
Parameters (out)	FrIf_StatePtr	Pointer to a memory location where the retrieved FrIfState will be stored
Return value	Std_ReturnType	E_OK: Function was successfully executed. State transition request was accepted. E_NOT_OK: Function execution failed due to detected errors. State transition request was not accepted.
Description	Get current FrIf state.	
Available via	FrIf.h	

]

[SWS_FrIf_05171] [If parameter FrIf_ClstIdx of FrIf_GetState has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetState shall report development error code FRIF_E_INV_CLST_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05172] [If parameter FrIf_StatePtr of FrIf_GetState equals NULL_PTR and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetState shall report development error code FRIF_E_PARAM_POINTER to the Det_ReportError service of the DET module.]

[SWS_FrIf_05173] [Caveats of FrIf_GetState: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [SWS_FrIf_05003]]

8.3.10 FrIf_SetState

[SWS_FrIf_05174] Definition of API function FrIf_SetState [

Service Name	FrIf_SetState	
Syntax	<pre>Std_ReturnType FrIf_SetState (uint8 FrIf_ClstIdx, FrIf_StateTransitionType FrIf_StateTransition)</pre>	
Service ID [hex]	0x08	





Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	FrIf_ClstIdx	Index of the cluster addressed.
	FrIf_StateTransition	Requested FrIf state transition.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Function was successfully executed. State transition request was accepted.
		E_NOT_OK: Function execution failed due to detected errors. State transition request was not accepted.
Description	Requests FrIf state machine transition.	
Available via	FrIf.h	

]

[SWS_FrIf_05175] [If parameter FrIf_ClstIdx of FrIf_SetState has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetState shall report development error code FRIF_E_INV_CLST_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05037] [If parameter FrIf_StateTransition of FrIf_SetState has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetState shall report development error code FRIF_E_INV_FRIF_STATE to the Det_ReportError service of the DET module.]

[SWS_FrIf_05176] [Caveats of FrIf_SetState: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#)]

8.3.11 FrIf_SetWakeupChannel

[SWS_FrIf_05010] Definition of callback function FrIf_SetWakeupChannel [

Service Name	FrIf_SetWakeupChannel	
Syntax	<pre>Std_ReturnType FrIf_SetWakeupChannel (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx)</pre>	
Service ID [hex]	0x09	
Sync/Async	Synchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout)	None	
Parameters (out)	None	





Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_SetWakeupChannel(). The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

└

[SWS_FrIf_05500] [If parameter FrIf_CtrlIdx of FrIf_SetWakeupChannel has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetWakeupChannel shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05177] [If parameter FrIf_ChnlIdx of FrIf_SetWakeupChannel has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetWakeupChannel shall report development error code FRIF_E_INV_CHNL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05178] [The function FrIf_SetWakeupChannel shall wrap the FlexRay Driver API function Fr_SetWakeupChannel() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters Fr_ChnlIdx to FrIf_ChnlIdx
3. Calling Fr_SetWakeupChannel() of the determined FlexRay Driver module with the parameters determined as described above.

└

[SWS_FrIf_05179] [Caveats of FrIf_SetWakeupChannel: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.12 FrIf_SendWUP

[SWS_FrIf_05011] Definition of API function FrIf_SendWUP

Upstream requirements: [SRS_Fr_05018](#)

[

Service Name	FrIf_SendWUP	
Syntax	<pre>Std_ReturnType FrIf_SendWUP (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x0a	
Sync/Async	Asynchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_SendWUP().	
Available via	FrIf.h	

]

[SWS_FrIf_05180] [If parameter FrIf_CtrlIdx of FrIf_SendWUP has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SendWUP shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05181] [The function FrIf_SendWUP shall wrap the FlexRay Driver API function Fr_SendWUP() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Calling Fr_SendWUP() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05182] [Caveats of FrIf_SendWUP: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.13 FrIf_GetPOCStatus

[SWS_FrIf_05014] Definition of API function FrIf_GetPOCStatus

Upstream requirements: [SRS_Fr_05022](#)

Service Name	FrIf_GetPOCStatus	
Syntax	<pre>Std_ReturnType FrIf_GetPOCStatus (uint8 FrIf_CtrlIdx, Fr_POCStatusType* FrIf_POCStatusPtr)</pre>	
Service ID [hex]	0x0d	
Sync/Async	Synchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	FrIf_POCStatusPtr	Pointer to a memory location where output value will be stored.
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_GetPOCStatus().	
Available via	FrIf.h	

[SWS_FrIf_05190] [If parameter FrIf_CtrlIdx of FrIf_GetPOCStatus has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetPOCStatus shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05192] [The function FrIf_GetPOCStatus shall wrap the FlexRay Driver API function Fr_GetPOCStatus() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters Fr_POCStatusPtr to FrIf_POCStatusPtr
3. Calling Fr_GetPOCStatus() of the determined FlexRay Driver module with the parameters determined as described above.

[SWS_FrIf_05193] [Caveats of FrIf_GetPOCStatus: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.14 FrIf_GetGlobalTime

[SWS_FrIf_05015] Definition of API function FrIf_GetGlobalTime [

Service Name	FrIf_GetGlobalTime	
Syntax	<pre>Std_ReturnType FrIf_GetGlobalTime (uint8 FrIf_CtrlIdx, uint8* FrIf_CyclePtr, uint16* FrIf_MacroTickPtr)</pre>	
Service ID [hex]	0x0e	
Sync/Async	Synchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	FrIf_CyclePtr	Pointer to a memory location where output value will be stored.
	FrIf_MacroTickPtr	Pointer to a memory location where output value will be stored.
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_GetGlobalTime(). Important Note: FrIf_GetGlobalTime may be called within an exclusive area.	
Available via	FrIf.h	

]

[SWS_FrIf_05194] [If parameter FrIf_CtrlIdx of FrIf_GetGlobalTime has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetGlobalTime shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05195] [The function FrIf_GetGlobalTime shall wrap the FlexRay Driver API function Fr_GetGlobalTime() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters
3. Fr_CylcePtr to FrIf_CyclePtr
Fr_MacroTickPtr to FrIf_MacroTickPtr
4. Calling Fr_GetGlobalTime() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05196] [Caveats of FrIf_GetGlobalTime: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.15 FrIf_AllowColdstart

[SWS_FrIf_05017] Definition of API function FrIf_AllowColdstart [

Service Name	FrIf_AllowColdstart	
Syntax	<pre>Std_ReturnType FrIf_AllowColdstart (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x10	
Sync/Async	Asynchronous	
Reentrancy	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_AllowColdstart().	
Available via	FrIf.h	

]

[SWS_FrIf_05200] [If parameter FrIf_CtrlIdx of FrIf_AllowColdstart has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_AllowColdstart shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05201] [The function FrIf_AllowColdstart shall wrap the FlexRay Driver API function Fr_AllowColdstart() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Calling Fr_AllowColdstart() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05202] [Caveats: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.16 FrIf_GetMacroticksPerCycle

[SWS_FrIf_05018] Definition of API function FrIf_GetMacroticksPerCycle [

Service Name	FrIf_GetMacroticksPerCycle	
Syntax	<pre>uint16 FrIf_GetMacroticksPerCycle (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x11	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint16	Number of Macroticks per Cycle
Description	Retrieves the amount of Macroticks per Cycle	
Available via	FrIf.h	

]

[SWS_FrIf_05203] [If parameter FrIf_CtrlIdx of FrIf_GetMacroticksPerCycle has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetMacroticksPerCycle shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.

This API service of the FlexRay Interface retrieves the number of Macroticks per Flex Ray Cycle of the FlexRay Cluster with index FrIf_CtrlIdx out of the static configuration.]

[SWS_FrIf_05204] [Caveats of FrIf_GetMacroticksPerCycle: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.17 FrIf_GetMacrotickDuration

[SWS_FrIf_05019] Definition of API function FrIf_GetMacrotickDuration [

Service Name	FrIf_GetMacrotickDuration	
Syntax	<pre>uint16 FrIf_GetMacrotickDuration (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x31	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	uint16	Duration of one Macrotick in ns
Description	Retrieves the Duration of a Macrotick in ns	





Available via	FrIf.h
---------------	--------

]

[SWS_FrIf_05191] [If parameter `FrIf_CtrlIdx` of `FrIf_GetMacrotickDuration`: has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_GetMacrotickDuration`: shall report development error code `FRIF_E_INV_CTRL_IDX` to the `Det_ReportError` service of the DET module.]

This API service of the FlexRay Interface retrieves duration of one Macrotick in nanoseconds of the FlexRay Cluster with index `FrIf_CtrlIdx` out of the static configuration.]

[SWS_FrIf_05754] [Caveats of `FrIf_GetMacrotickDuration`: The FlexRay Interface module has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.18 FrIf_Transmit

[SWS_FrIf_05033] Definition of API function `FrIf_Transmit` [

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

]

[SWS_FrIf_05318] [

`FrIf_Transmit()` shall return `E_NOT_OK` in case the FrIf's state is `FRIF_STATE_OFFLINE`.]

[SWS_FrIf_05205] [If parameter `TxPduId` of `FrIf_Transmit` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_Transmit` shall report development error code `FRIF_E_INV_TXPDUID` to the `Det_ReportError` service of the DET module.]

[SWS_Frlf_05207] [If the parameter `FrlfAllowedDynamicSduLength` is set to false and/or if the parameter `FrlfImmediate` is set to true for the passed `TxPduId`, the passed `SduDataPtr` in parameter `PduInfoPtr` of `Frlf_Transmit` shall be checked for `NULL_PTR` in case development error detection is enabled (i.e. `FrlfDevErrorDetect` equals ON). If in this case the passed `SduDataPtr` equals `NULL_PTR`, the function `Frlf_Transmit` shall report the development error code `FRIF_E_PARAM_POINTER` to the `Det_ReportError` service of the DET module.

In case of decoupled transmission the PDU with index `TxPduId` is not yet passed to the underlying FlexRay Driver module for transmission. `Frlf` only remembers the PDU's transmission request (increment `TrigTxCounter` ⁵) This decoupling mechanism between the call of `Frlf_Transmit()` and the execution of the `FrlfCommunicationAction` (see [\[ECUC_Frlf_06067\]](#)) has some implications:

- The upper layer BSW module may operate asynchronously to the FlexRay Communication System and thus may call `Frlf_Transmit()` at any point in time.
- The upper layer BSW module must permanently buffer the PDU's payload data and must be able to handle a call of its `LSduR_FrlfTriggerTransmit()` API service at (from the BSW's point of view) any arbitrary point in time.

]

[SWS_Frlf_05208] [In case of immediate transmission the function `Frlf_Transmit` shall pass the PDU (single PDU, no Update bit) to the underlying FlexRay Driver module immediately for transmission.]

[SWS_Frlf_05757] ["If parameter `TxPduId` is configured for an immediate PDU, and if configuration parameter `FrlfAllowDynamicLSduLength` is set to FALSE, the provided length in `PduInfoPtr` shall be compared with the static configured length (see [\[ECUC_Frlf_06054\]](#)).

If the length information does not match, `Frlf_Transmit()` shall return `E_NOT_OK` and shall not perform the immediate PDU transmission. If development error detection is enabled (i.e. `FrlfDevErrorDetect` equals ON), the function `Frlf_Transmit()` shall report development error code `FRIF_E_INV_PDULENGTH` to the `Det_ReportError` service of the DET module.]

[SWS_Frlf_05209] [Caveats of `Frlf_Transmit`: The FlexRay Interface module has to be initialized with a call of `Frlf_Init()` before this API service may be called, see [\[SWS_Frlf_05003\]](#)]

⁵Limited by static configuration see [FrlfCounterLimit](#)

8.3.19 FrIf_SetTransceiverMode

[SWS_FrIf_05034] Definition of API function FrIf_SetTransceiverMode

Upstream requirements: [SRS_Fr_05039](#)

Service Name	FrIf_SetTransceiverMode	
Syntax	<pre>Std_ReturnType FrIf_SetTransceiverMode (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcevModeType FrIf_TrcevMode)</pre>	
Service ID [hex]	0x13	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
	FrIf_TrcevMode	Transceiver mode to be set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned E_NOT_OK.
Description	Wraps the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode(). The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

[SWS_FrIf_05210] [If parameter FrIf_CtrlIdx of FrIf_SetTransceiverMode has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetTransceiverMode shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05211] [If parameter FrIf_ChnlIdx of FrIf_SetTransceiverMode has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_SetTransceiverMode shall report development error code FRIF_E_INV_CHNL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05212] [The function FrIf_SetTransceiverMode shall wrap the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode() by:

1. Translating (based on static FrIf module configuration) the tuple (FlexRay CC index FrIf_CtrlIdx | FlexRay Channel index FrIf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_TrcevIdx).
2. Setting parameters
 - FrTrcv_TrcevMode to FrIf_TrcevMode

3. Calling `FrTrcv_SetTransceiverMode()` of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05213] [Caveats of `FrIf_SetTransceiverMode`: The FlexRay Interface module has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.20 `FrIf_GetTransceiverMode`

[SWS_FrIf_05035] Definition of API function `FrIf_GetTransceiverMode`

Upstream requirements: [SRS_Fr_05157](#)

[

Service Name	<code>FrIf_GetTransceiverMode</code>	
Syntax	<pre>Std_ReturnType FrIf_GetTransceiverMode (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcevModeType* FrIf_TrcevModePtr)</pre>	
Service ID [hex]	0x14	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>FrIf_ChnlIdx</code>	Index of the FlexRay Channel to address in scope of the FlexRay controller <code>FrIf_CtrlIdx</code> .
Parameters (inout)	None	
Parameters (out)	<code>FrIf_TrcevModePtr</code>	Pointer to a memory location where output value will be stored.
Return value	<code>Std_ReturnType</code>	<code>E_OK</code> : The call of the FlexRay Transceiver Driver's API service has returned <code>E_OK</code> . <code>E_NOT_OK</code> : The call of the FlexRay Transceiver Driver's API service has returned <code>E_NOT_OK</code> .
Description	Wraps the FlexRay Transceiver Driver API function <code>FrTrcv_GetTransceiverMode()</code> . The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	<code>FrIf.h</code>	

]

[SWS_FrIf_05214] [If parameter `FrIf_CtrlIdx` of `FrIf_GetTransceiverMode` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_GetTransceiverMode` shall report development error code `FRIF_E_INV_CTRL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_FrIf_05215] [If parameter `FrIf_ChnlIdx` of `FrIf_GetTransceiverMode` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_GetTransceiverMode` shall report development error code `FRIF_E_INV_CHNL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_FrIf_05216] [The function `FrIf_GetTransceiverMode` shall wrap the FlexRay Transceiver Driver API function `FrTrcv_GetTransceiverMode()` by:

1. Translating (based on static FrIf module configuration) the tuple (FlexRay CC index FrIf_CtrlIdx | FlexRay Channel index FrIf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_TrvcIdx).
2. Setting parameters
 - FrTrcv_TrcvModePtr to FrIf_TrvcModePtr
3. Calling FrTrcv_GetTransceiverMode() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05217] [Caveats of FrIf_GetTransceiverMode: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.21 FrIf_GetTransceiverWUReason

[SWS_FrIf_05036] Definition of API function FrIf_GetTransceiverWUReason

Upstream requirements: [SRS_BSW_00375](#), [SRS_Fr_05158](#)

[

Service Name	FrIf_GetTransceiverWUReason	
Syntax	<pre>Std_ReturnType FrIf_GetTransceiverWUReason (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrvcWUReasonType* FrIf_TrvcWUReasonPtr)</pre>	
Service ID [hex]	0x15	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout)	None	
Parameters (out)	FrIf_TrvcWUReasonPtr	Pointer to a memory location where output value will be stored.
Return value	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned E_NOT_OK.
Description	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverWUReason(). The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

]

[SWS_FrIf_05218] [If parameter FrIf_CtrlIdx of FrIf_GetTransceiverWUReason has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetTransceiverWUReason shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_Frlf_05219] [If parameter `Frlf_ChnlIdx` of `Frlf_GetTransceiverWUReason` has an invalid value and if development error detection is enabled (i.e. `FrlfDevErrorDetect` equals ON), the function `Frlf_GetTransceiverWUReason` shall report development error code `FRIF_E_INV_CHNL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_Frlf_05220] [The function `Frlf_GetTransceiverWUReason` shall wrap the FlexRay Transceiver Driver API function `FrTrcv_GetTransceiverWUReason()` by:

1. Translating (based on static `Frlf` module configuration) the tuple (FlexRay CC index `Frlf_CtrlIdx` | FlexRay Channel index `Frlf_ChnlIdx`) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index `FrTrcv_TrvcIdx`).
2. Setting parameters
 - `FrTrcv_TrcvWUReasonPtr` to `Frlf_WUReasonPtr`
3. Calling `FrTrcv_GetTransceiverWUReason()` of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_Frlf_05221] [Caveats of `Frlf_GetTransceiverWUReason`: The FlexRay Interface module has to be initialized with a call of `Frlf_Init()` before this API service may be called, see [\[SWS_Frlf_05003\]](#).]

8.3.22 Frlf_ClearTransceiverWakeup

[SWS_Frlf_05039] Definition of API function `Frlf_ClearTransceiverWakeup`

Upstream requirements: [SRS_Fr_05161](#)

[

Service Name	Frlf_ClearTransceiverWakeup	
Syntax	<pre>Std_ReturnType FrIf_ClearTransceiverWakeup (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx)</pre>	
Service ID [hex]	0x18	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	<code>Frlf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>Frlf_ChnlIdx</code>	Index of the FlexRay Channel to address in scope of the FlexRay controller <code>Frlf_CtrlIdx</code> .
Parameters (inout)	None	
Parameters (out)	None	





Return value	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned E_NOT_OK.
Description	Wraps the FlexRay Transceiver Driver API function FrTrcv_ClearTransceiverWakeup(). The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

]

[SWS_FrIf_05230] [If parameter FrIf_CtrlIdx of FrIf_ClearTransceiverWakeup has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_ClearTransceiverWakeup shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05231] [If parameter FrIf_ChnlIdx of FrIf_ClearTransceiverWakeup has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_ClearTransceiverWakeup shall report development error code FRIF_E_INV_CHNL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05232] [The function FrIf_ClearTransceiverWakeup shall wrap the FlexRay Transceiver Driver API function FrTrcv_ClearTransceiverWakeup() by:

1. Translating (based on static FrIf module configuration) the tuple (FlexRay CC index FrIf_CtrlIdx | FlexRay Channel index FrIf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_TrcvIdx).
2. Calling FrTrcv_ClearTransceiverWakeup() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05233] [Caveats of FrIf_ClearTransceiverWakeup: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.23 FrIf_CancelAbsoluteTimer

[SWS_FrIf_05023] Definition of API function FrIf_CancelAbsoluteTimer [

Service Name	FrIf_CancelAbsoluteTimer
Syntax	Std_ReturnType FrIf_CancelAbsoluteTimer (uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx)
Service ID [hex]	0x1b
Sync/Async	Synchronous
Reentrancy	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs





Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_AbsTimerIdx	Index of the absolute timer to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_CancelAbsoluteTimer() .	
Available via	FrIf.h	

]

[SWS_FrIf_05240] [If parameter FrIf_CtrlIdx of FrIf_CancelAbsoluteTimer has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_CancelAbsoluteTimer shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05241] [The function FrIf_CancelAbsoluteTimer shall wrap the FlexRay Driver API function Fr_CancelAbsoluteTimer() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters Fr_AbsTimerIdx to FrIf_AbsTimerIdx
3. Calling Fr_CancelAbsoluteTimer() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05242] [Caveats of FrIf_CancelAbsoluteTimer: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.24 FrIf_GetAbsoluteTimerIRQStatus

[SWS_FrIf_05027] Definition of API function FrIf_GetAbsoluteTimerIRQStatus [

Service Name	FrIf_GetAbsoluteTimerIRQStatus
Syntax	Std_ReturnType FrIf_GetAbsoluteTimerIRQStatus (uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx, boolean* FrIf_IRQStatusPtr)
Service ID [hex]	0x1f
Sync/Async	Synchronous





Reentrancy	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_AbsTimerIdx	Index of the absolute timer to address.
Parameters (inout)	None	
Parameters (out)	FrIf_IRQStatusPtr	Pointer to a memory location where output value will be stored.
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_GetAbsoluteTimerIRQStatus()	
Available via	FrIf.h	

]

[SWS_FrIf_05252] [If parameter FrIf_CtrlIdx of FrIf_GetAbsoluteTimerIRQStatus has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetAbsoluteTimerIRQStatus shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05253] [The function FrIf_GetAbsoluteTimerIRQStatus shall wrap the FlexRay Driver API function Fr_GetAbsoluteTimerIRQStatus() by:

1. Translating (based on static FrIf module configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).
2. Setting parameters
 - Fr_AbsTimerIdx to FrIf_AbsTimerIdx
 - Fr_IRQStatusPtr to FrIf_IRQStatusPtr
3. Calling Fr_GetAbsoluteTimerIRQStatus() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05254] [Caveats of FrIf_GetAbsoluteTimerIRQStatus: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.3.25 FrIf_DisableAbsoluteTimerIRQ

[SWS_FrIf_05031] Definition of API function FrIf_DisableAbsoluteTimerIRQ [

Service Name	FrIf_DisableAbsoluteTimerIRQ	
Syntax	<pre>Std_ReturnType FrIf_DisableAbsoluteTimerIRQ (uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx)</pre>	
Service ID [hex]	0x23	
Sync/Async	Synchronous	
Reentrancy	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_AbsTimerIdx	Index of the absolute timer to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_DisableAbsoluteTimerIRQ().	
Available via	FrIf.h	

]

[SWS_FrIf_05264] [If parameter FrIf_CtrlIdx of FrIf_DisableAbsoluteTimerIRQ has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_DisableAbsoluteTimerIRQ shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05266] [Caveats of FrIf_DisableAbsoluteTimerIRQ: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [SWS_FrIf_05003].]

8.3.26 FrIf_GetCycleLength

[SWS_FrIf_05239] Definition of API function FrIf_GetCycleLength [

Service Name	FrIf_GetCycleLength	
Syntax	<pre>uint32 FrIf_GetCycleLength (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x3a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	





Return value	uint32	Time in unit of nanoseconds
Description	This API returns the configured time of the configuration parameter "GdCycle" in nanoseconds for the FlexRay controller with index <code>FrIf_CtrlIdx</code> .	
Available via	FrIf.h	

]

[SWS_FrIf_05237] [If parameter `FrIf_CtrlIdx` of `FrIf_GetCycleLength` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_GetCycleLength` shall report development error code `FRIF_E_INV_CTRL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_FrIf_05238] [Caveats of `FrIf_GetCycleLength`: The FlexRay Interface module has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.4 Optional Function Definitions

8.4.1 FrIf_AllSlots

[SWS_FrIf_05020] Definition of API function `FrIf_AllSlots` [

Service Name	<code>FrIf_AllSlots</code>	
Syntax	<pre>Std_ReturnType FrIf_AllSlots (uint8 FrIf_CtrlIdx)</pre>	
Service ID [hex]	0x33	
Sync/Async	Synchronous	
Reentrancy	non reentrant	
Parameters (in)	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	None	
Return value	<code>Std_ReturnType</code>	<code>E_OK</code> : The call of the FlexRay Driver's API service has returned <code>E_OK</code> . <code>E_NOT_OK</code> : The call of the FlexRay Driver's API service has returned <code>E_NOT_OK</code> , or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function <code>Fr_AllSlots</code>	
Available via	FrIf.h	

]

[SWS_FrIf_05412] [The function `FrIf_AllSlots` shall be pre compile time configurable ON/OFF by the configuration parameter `FrIfAllSlotsSupport` (derived from configuration parameter `FrIfAllSlotsSupport`, see [ECUC_FrIf_06108](#))]

[SWS_FrIf_05706] [If development error detection for the FrIf module is enabled: if the function FrIf_AllSlots is called before the FrIf was initialized successfully, the function FrIf_AllSlots shall raise the development error FRIF_E_UNINIT and return E_NOT_OK.]

[SWS_FrIf_05707] [If development error detection for the Fr module is enabled: the function FrIf_AllSlots shall check the parameter FrIf_CtrlIdx for being valid. If FrIf_CtrlIdx is invalid, the function FrIf_AllSlots shall raise the development error FRIF_E_INV_CTRL_IDX and return E_NOT_OK.]

8.4.2 FrIf_GetChannelStatus

[SWS_FrIf_05030] Definition of API function FrIf_GetChannelStatus [

Service Name	FrIf_GetChannelStatus	
Syntax	<pre>Std_ReturnType FrIf_GetChannelStatus (uint8 FrIf_CtrlIdx, uint16* FrIf_ChannelAStatusPtr, uint16* FrIf_ChannelBStatusPtr)</pre>	
Service ID [hex]	0x26	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same device	
Parameters (in)	FrIf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Interface.
Parameters (inout)	None	
Parameters (out)	FrIf_ChannelAStatusPtr	Address where the bitcoded channel A status information shall be stored.
	FrIf_ChannelBStatusPtr	Address where the bitcoded channel B status information shall be stored.
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description	Wraps the FlexRay Driver API function Fr_GetChannelStatus() and gets the channel status information.	
Available via	FrIf.h	

]

[SWS_FrIf_05413] [The function FrIf_GetChannelStatus shall be pre compile time configurable ON/OFF by the configuration parameter FrIfGetGetChannelStatusSupport (derived from configuration parameter FrIfGetGetChannelStatusSupport, see ECUC_FrIf_06105)]

[SWS_FrIf_05708] [If development error detection for the FrIf module is enabled: if the function FrIf_GetChannelStatus is called before the FrIf module was initialized successfully, the function FrIf_GetChannelStatus shall raise the development error FRIF_E_UNINIT and return E_NOT_OK.]

[SWS_FrIf_05709] [If development error detection for the FrIf module is enabled: the function FrIf_GetChannelStatus shall check the parameter FrIf_CtrlIdx for being valid. If FrIf_CtrlIdx is invalid, the function FrIf_GetChannelStatus shall raise the development error FRIF_E_INV_CTRL_IDX and return E_NOT_OK.]

8.4.3 FrIf_GetClockCorrection

[SWS_FrIf_05071] Definition of API function FrIf_GetClockCorrection [

Service Name	FrIf_GetClockCorrection	
Syntax	<pre>Std_ReturnType FrIf_GetClockCorrection (uint8 FrIf_CtrlIdx, sint16* FrIf_RateCorrectionPtr, sint32* FrIf_OffsetCorrectionPtr)</pre>	
Service ID [hex]	0x29	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same device	
Parameters (in)	FrIf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Interface.
Parameters (inout)	None	
Parameters (out)	FrIf_RateCorrectionPtr	Address where the current rate correction value shall be stored.
	FrIf_OffsetCorrectionPtr	Address where the current offset correction value shall be stored.
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description	Wraps the FlexRay Driver API function Fr_GetClockCorrection () and gets the current clock correction values.	
Available via	FrIf.h	

]

[SWS_FrIf_05414] [The function FrIf_GetClockCorrection shall be pre compile time configurable ON/OFF by the configuration parameter FrIfGetClockCorrectionSupport (derived from configuration parameter FrIfGetClockCorrectionSupport, see ECUC_FrIf_06106)]

[SWS_FrIf_05711] [If development error detection for the FrIf module is enabled: if the function FrIf_GetClockCorrection is called before the FrIf was initialized successfully, the function FrIf_GetClockCorrection shall raise the development error FRIF_E_UNINIT and return E_NOT_OK.]

[SWS_FrIf_05712] [If development error detection for the FrIf module is enabled: the function FrIf_GetClockCorrection shall check the parameter FrIf_CtrlIdx for being valid. If FrIf_CtrlIdx is invalid, the function FrIf_GetClockCorrection shall raise the development error FRIF_E_INV_CTRL_IDX and return E_NOT_OK.]

8.4.4 FrIf_GetSyncFrameList

[SWS_FrIf_05072] Definition of API function FrIf_GetSyncFrameList [

Service Name	FrIf_GetSyncFrameList	
Syntax	<pre>Std_ReturnType FrIf_GetSyncFrameList (uint8 FrIf_CtrlIdx, uint8 FrIf_ListSize, uint16* FrIf_ChannelAEvenListPtr, uint16* FrIf_ChannelBEvenListPtr, uint16* FrIf_ChannelAOddListPtr, uint16* FrIf_ChannelBOddListPtr)</pre>	
Service ID [hex]	0x2a	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same device	
Parameters (in)	FrIf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Interface.
	FrIf_ListSize	Size of the arrays passed via parameters: FrIf_ChannelAEvenListPtr FrIf_ChannelBEvenListPtr FrIf_ChannelAOddListPtr FrIf_ChannelBOddListPtr. The service must ensure to not write more entries into those arrays than granted by this parameter.
Parameters (inout)	None	
Parameters (out)	FrIf_ChannelAEvenListPtr	Address the list of syncframes on channel A within the even communication cycle is written to. The exact number of elements written to the list is limited by parameter FrIf_ListSize. Unused list elements are filled with the value '0' to indicate that no more syncframe has been seen.
	FrIf_ChannelBEvenListPtr	Address the list of syncframes on channel B within the even communication cycle is written to. The exact number of elements written to the list is limited by parameter FrIf_ListSize. Unused list elements are filled with the value '0' to indicate that no more syncframe has been seen.
	FrIf_ChannelAOddListPtr	Address the list of syncframes on channel A within the odd communication cycle is written to. The exact number of elements written to the list is limited by parameter FrIf_ListSize. Unused list elements are filled with the value '0' to indicate that no more syncframe has been seen.
	FrIf_ChannelBOddListPtr	Address the list of syncframes on channel B within the odd communication cycle is written to. The exact number of elements written to the list is limited by parameter FrIf_ListSize. Unused list elements are filled with the value '0' to indicate that no more syncframe has been seen.
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description	Wraps the FlexRay Driver API function Fr_GetSyncFrameList and gets a list of syncframes received or transmitted on channel A and channel B via the even and odd communication cycle.	
Available via	FrIf.h	

]

[SWS_FrIf_05415] [The function FrIf_GetSyncFrameList shall be pre compile time configurable ON/OFF by the configuration parameter FrIfGetSyncFrameListSupport (derived from configuration parameter FrIfGetSyncFrameListSupport, see ECUC_FrIf_06107)]

[SWS_FrIf_05715] [If development error detection for the FrIf module is enabled: if the function FrIf_GetSyncFrameList is called before the Fr was initialized successfully,

the function `FrIf_GetSyncFrameList` shall raise the development error `FRIF_E_UNINIT` and return `E_NOT_OK`.]

[SWS_FrIf_05716] [If development error detection for the `FrIf` module is enabled: the function `FrIf_GetSyncFrameList` shall check the parameter `FrIf_CtrlIdx` for being valid. If `FrIf_CtrlIdx` is invalid, the function `FrIf_GetSyncFrameList` shall raise the development error `FRIF_E_INV_CTRL_IDX` and return `E_NOT_OK`.]

8.4.5 `FrIf_GetNumOfStartupFrames`

[SWS_FrIf_05073] Definition of API function `FrIf_GetNumOfStartupFrames` [

Service Name	<code>FrIf_GetNumOfStartupFrames</code>	
Syntax	<pre>Std_ReturnType FrIf_GetNumOfStartupFrames (uint8 FrIf_CtrlIdx, uint8* FrIf_NumOfStartupFramesPtr)</pre>	
Service ID [hex]	0x34	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same device	
Parameters (in)	<code>FrIf_CtrlIdx</code>	Index of FlexRay CC within the context of the FlexRay Interface.
Parameters (inout)	None	
Parameters (out)	<code>FrIf_NumOfStartupFramesPtr</code>	Address where the number of startup frames seen within the last even/odd cycle pair shall be stored.
Return value	<code>Std_ReturnType</code>	<code>E_OK</code> : API call finished successfully. <code>E_NOT_OK</code> : API call aborted due to errors.
Description	Wraps the FlexRay Driver API function <code>Fr_GetNumOfStartupFrames</code> and gets a list of the current number of startup frames seen on the cluster. See variable <code>vStartupPairs</code> of [12] for details.	
Available via	<code>FrIf.h</code>	

]

[SWS_FrIf_05416] [The function `FrIf_GetNumOfStartupFrames` shall be pre compile time configurable ON/OFF by the configuration parameter `FrIfGetNumOfStartupFramesSupport` (derived from configuration parameter `FrIfGetNumOfStartupFramesSupport`, see `ECUC_FrIf_06104`).]

[SWS_FrIf_05721] [If development error detection for the `FrIf` module is enabled: if the function `FrIf_GetNumOfStartupFrames` is called before the `FrIf` was initialized successfully, the function `FrIf_GetNumOfStartupFrames` shall raise the development error `FRIF_E_UNINIT` and return `E_NOT_OK`.]

[SWS_FrIf_05722] [If development error detection for the `FrIf` module is enabled: the function `FrIf_GetNumOfStartupFrames` shall check the parameter `FrIf_CtrlIdx` for being valid. If `FrIf_CtrlIdx` is invalid, the function `FrIf_GetNumOfStartupFrames` shall raise the development error `FRIF_E_INV_CTRL_IDX` and return `E_NOT_OK`.]

8.4.6 FrIf_GetWakeupRxStatus

[SWS_FrIf_05102] Definition of API function FrIf_GetWakeupRxStatus [

Service Name	FrIf_GetWakeupRxStatus	
Syntax	<pre>Std_ReturnType FrIf_GetWakeupRxStatus (uint8 FrIf_CtrlIdx, uint8* FrIf_WakeupRxStatusPtr)</pre>	
Service ID [hex]	0x2b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same device	
Parameters (in)	FrIf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout)	None	
Parameters (out)	FrIf_WakeupRxStatusPtr	Address where bitcoded wakeup reception status shall be stored. Bit 0: Wakeup received on channel A indicator Bit 1: Wakeup received on channel B indicator Bit 2-7: Unused
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description	Wraps the FlexRay Driver API function Fr_GetWakeupRxStatus and gets the wakeup received information from the FlexRay controller.	
Available via	FrIf.h	

]

[SWS_FrIf_05417] [The function FrIf_GetWakeupRxStatus shall be pre compile time configurable ON/OFF by the configuration parameter FrIfGetWakeupRxStatusSupport (derived from configuration parameter FrIfGetWakeupRxStatusSupport, see ECUC_FrIf_06111)]

[SWS_FrIf_05700] [If development error detection for the FrIf module is enabled: if the function FrIf_GetWakeupRxStatus is called before the Fr was initialized successfully, the function FrIf_GetWakeupRxStatus shall raise the development error FRIF_E_UNINIT and return E_NOT_OK.]

[SWS_FrIf_05701] [If development error detection for the FrIf module is enabled: the function FrIf_GetWakeupRxStatus shall check the parameter FrIf_CtrlIdx for being valid. If FrIf_CtrlIdx is invalid, the function FrIf_GetWakeupRxStatus shall raise the development error FRIF_E_INV_CTRL_IDX and return E_NOT_OK.]

8.4.7 FrIf_CancelTransmit

[SWS_FrIf_05070] Definition of API function FrIf_CancelTransmit [

Service Name	FrIf_CancelTransmit	
Syntax	<pre>Std_ReturnType FrIf_CancelTransmit (PduIdType TxPduId)</pre>	
Service ID [hex]	0x4a	





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

]

[SWS_FrIf_05713] [The function `FrIf_CancelTransmit` shall be pre compile time configurable ON/OFF by the configuration parameter `FrIfCancelTransmitSupport` (derived from configuration parameter `FrIfCancelTransmitSupport`, see [ECUC_FrIf_00002](#))]

[SWS_FrIf_05703] [If development error detection for the `FrIf` module is enabled: if the function `FrIf_CancelTransmit` is called before the `FrIf` was initialized successfully, the function `FrIf_CancelTransmit` shall raise the development error `FRIF_E_UNINIT` and return `E_NOT_OK`.]

[SWS_FrIf_05704] [If development error detection for the `FrIf` module is enabled: the function `FrIf_CancelTransmit` shall check the parameter `TxPduId` for being valid. If `TxPduId` is invalid, the function `FrIf_CancelTransmit` shall raise the development error `FRIF_E_INV_TXPDUID` and return `E_NOT_OK`.]

[SWS_FrIf_05705] [For Transmit Cancellation, the following steps are performed:

1. Decrement `TrigTxCounter` for the IPDU that shall be canceled.
2. If `TxConfCounter > 0` for this PDU, continue with step 3). Else, stop here.
3. Call FlexRay Driver's API function `Fr_CancelTxLPdu()`:
 - (a) `Fr_CtrlIdx` is derived according to the indexing scheme described in chapter of indexing.
 - (b) `Fr_LPduIdx` is set to the configured L-PDU buffer index [Configuration Parameter `FrIfLPduIdx`, see [ECUC_FrIf_06058](#)] associated with the Communication Operation.
4. Increment `TrigTxCounter` (limited by `FrIfCounterLimit`) for all other I-PDUs within that L-PDU that have a `TxConfCounter > 0`.
5. Decrement `TxConfCounter` for all other I-PDUs within that L-PDU that have a `TxConfCounter > 0`.
6. Decrement the `TxConfCounter` for the IPDU that has been initiated by the `CancelTransmit` API call.

]

8.4.8 FrIf_DisableLPdu

[SWS_FrIf_05710] Definition of API function FrIf_DisableLPdu [

Service Name	FrIf_DisableLPdu	
Syntax	<pre>Std_ReturnType FrIf_DisableLPdu (uint8 FrIf_CtrlIdx, uint16 FrIf_LPduIdx)</pre>	
Service ID [hex]	0x28	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same device	
Parameters (in)	FrIf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Interface.
	FrIf_LPduIdx	This index is used to uniquely identify a FlexRay frame
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description	Wraps the FlexRay Driver Function Fr_DisableLPdu. It disables the hardware resource of an LPdu for transmission/reception.	
Available via	FrIf.h	

]

[SWS_FrIf_05418] [The function FrIf_DisableLPdu shall be pre compile time configurable ON/OFF by the configuration parameter FrIfDisableLPduSupport (derived from configuration parameter FrIfDisableLPduSupport, see ECUC_FrIf_06110)]

[SWS_FrIf_05717] [If development error detection for the FrIf module is enabled: if the function FrIf_DisableLPdu is called before the FrIf was initialized successfully, the function FrIf_DisableLPdu shall raise the development error FRIF_E_UNINIT and return E_NOT_OK.]

[SWS_FrIf_05714] [If development error detection for the FrIf module is enabled: the function FrIf_DisableLPdu shall check the parameter FrIf_CtrlIdx for being valid. If FrIf_CtrlIdx is invalid, the function FrIf_DisableLPdu shall raise the development error FRIF_E_INV_CTRL_IDX and return E_NOT_OK.]

8.4.9 FrIf_GetTransceiverError

[SWS_FrIf_05032] Definition of API function FrIf_GetTransceiverError [

Service Name	FrIf_GetTransceiverError	
Syntax	<pre>Std_ReturnType FrIf_GetTransceiverError (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, uint8 FrIf_BranchIdx, uint32* FrIf_BusErrorState)</pre>	
Service ID [hex]	0x35	
Sync/Async	Synchronous	
Reentrancy	Function is non reentrant for the same channel of the same controller.	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
	FrIf_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout)	None	
Parameters (out)	FrIf_BusErrorState	Address where the transceiver error state is stored.
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors
Description	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverError. The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

]

[SWS_FrIf_05419] [The function FrIf_GetTransceiverError shall be pre compile time configurable ON/OFF by the configuration parameter FrIfGetTransceiverErrorSupport (derived from configuration parameter FrIfGetTransceiverErrorSupport, see ECUC_FrIf_06101)]

[SWS_FrIf_05718] [If development error detection for the FrIf module is enabled: if the function FrIf_GetTransceiverError is called before the FrIf was initialized successfully, the function FrIf_GetTransceiverError shall raise the development error FRIF_E_UNINIT and return E_NOT_OK.]

[SWS_FrIf_05719] [If development error detection for the FrIf module is enabled: the function FrIf_GetTransceiverError shall check the parameter FrIf_CtrlIdx for being valid. If FrIf_CtrlIdx is invalid, the function FrIf_GetTransceiverError shall raise the development error FRIF_E_INV_CTRL_IDX and return E_NOT_OK.]

[SWS_FrIf_05720] [If parameter FrIf_ChnlIdx of FrIf_GetTransceiverError has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_GetTransceiverError shall report development error code FRIF_E_INV_CHNL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05728] [The function FrIf_GetTransceiverError shall wrap the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverError by:

1. Translating (based on static FrIf module configuration) the tuple (FlexRay CC index FrIf_CtrlIdx | FlexRay Channel index FrIf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_TrvcIdx).
2. Setting parameters
 - FrTrcv_BranchIdx to FrIf_BranchIdx
 - FrTrcv_BusErrorState to FrIf_BusErrorState
3. Calling FrTrcv_GetTransceiverError of the determined FlexRay Transceiver module with the parameters determined as described above.

8.4.10 FrIf_EnableTransceiverBranch

[SWS_FrIf_05085] Definition of API function FrIf_EnableTransceiverBranch [

Service Name	FrIf_EnableTransceiverBranch	
Syntax	<pre>Std_ReturnType FrIf_EnableTransceiverBranch (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, uint8 FrIf_BranchIdx)</pre>	
Service ID [hex]	0x36	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
	FrIf_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned E_NOT_OK.
Description	Wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverBranch. The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

[SWS_FrIf_05420] [The function FrIf_EnableTransceiverBranch shall be pre compile time configurable ON/OFF by the configuration parameter FrIfEnableTransceiverBranchSupport (derived from configuration parameter FrIfEnableTransceiverBranchSupport, see ECUC_FrIf_06103)]

[SWS_FrIf_05302] [If parameter FrIf_CtrlIdx of FrIf_EnableTransceiverBranch has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect

equals ON), the function `FrIf_EnableTransceiverBranch` shall report development error code `FRIF_E_INV_CTRL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_FrIf_05304] [If parameter `FrIf_ChnlIdx` of `FrIf_EnableTransceiverBranch` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_EnableTransceiverBranch` shall report development error code `FRIF_E_INV_CHNL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_FrIf_05306] [The function `FrIf_EnableTransceiverBranch` shall wrap the FlexRay Transceiver Driver API function `FrIf_EnableTransceiverBranch` by:

1. Translating (based on static `FrIf` module configuration) the tuple (FlexRay CC index `FrIf_CtrlIdx` | FlexRay Channel index `FrIf_ChnlIdx`) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index `FrTrcv_TrcvIdx`).
2. Setting parameter: `FrTrcv_BranchIdx` to `FrIf_BranchIdx`
3. Calling `FrTrcv_EnableTransceiverBranch` of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05307] [If development error detection for the `FrIf` module is enabled: if the function `FrIf_EnableTransceiverBranch` is called before the `Fr` was initialized successfully, the function `FrIf_EnableTransceiverBranch` shall raise the development error `FRIF_E_UNINIT` and return `E_NOT_OK`.]

8.4.11 `FrIf_DisableTransceiverBranch`

[SWS_FrIf_05028] Definition of API function `FrIf_DisableTransceiverBranch` [

Service Name	<code>FrIf_DisableTransceiverBranch</code>	
Syntax	<pre>Std_ReturnType FrIf_DisableTransceiverBranch (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, uint8 FrIf_BranchIdx)</pre>	
Service ID [hex]	0x37	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>FrIf_ChnlIdx</code>	Index of the FlexRay Channel to address in scope of the FlexRay controller <code>FrIf_CtrlIdx</code> .
	<code>FrIf_BranchIdx</code>	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout)	None	
Parameters (out)	None	





Return value	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned E_NOT_OK.
Description	Wraps the FlexRay Transceiver Driver API function FrTrcv_DisableTransceiverBranch. The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

]

[SWS_FrIf_05421] [The function FrIf_DisableTransceiverBranch shall be pre compile time configurable ON/OFF by the configuration parameter FrIfDisableTransceiverBranchSupport (derived from configuration parameter FrIfDisableTransceiverBranchSupport, see ECUC_FrIf_06102)]

[SWS_FrIf_05425] [The function FrIf_DisableTransceiverBranch shall be pre compile time configurable ON/OFF by the configuration parameter FrIfDisableTransceiverBranchSupport (derived from configuration parameter FrIfDisableTransceiverBranchSupport, see ECUC_FrIf_06102)]

[SWS_FrIf_05756] [If parameter FrIf_CtrlIdx of FrIf_DisableTransceiverBranch has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_DisableTransceiverBranch shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05243] [If parameter FrIf_ChnlIdx of FrIf_DisableTransceiverBranch has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_DisableTransceiverBranch shall report development error code FRIF_E_INV_CHNL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05305] [The function FrIf_DisableTransceiverBranch shall wrap the FlexRay Transceiver Driver API function FrIf_DisableTransceiverBranch by:

1. Translating (based on static FrIf module configuration) the tuple (FlexRay CC index FrIf_CtrlIdx | FlexRay Channel index FrIf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_TrcvIdx)
2. Setting parameter: FrTrcv_BranchIdx to FrIf_BranchIdx
3. Calling FrTrcv_DisableTransceiverBranch() of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_FrIf_05308] [Caveats of FrIf_DisableTransceiverBranch: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.4.12 FrIf_ReconfigLPdu

[SWS_FrIf_05048] Definition of API function FrIf_ReconfigLPdu [

Service Name	FrIf_ReconfigLPdu	
Syntax	<pre>Std_ReturnType FrIf_ReconfigLPdu (uint8 FrIf_CtrlIdx, uint16 FrIf_LPduIdx, uint16 FrIf_FrameId, Fr_ChannelType FrIf_ChnlIdx, uint8 FrIf_CycleRepetition, uint8 FrIf_CycleOffset, uint8 FrIf_PayloadLength, uint16 FrIf_HeaderCRC)</pre>	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	FrIf_LPduIdx	This index is used to uniquely identify a FlexRay frame.
	FrIf_FrameId	FlexRay Frame ID the FrIf_LPdu shall be configured to.
	FrIf_ChnlIdx	FlexRay Channel the FrIf_LPdu shall be configured to.
	FrIf_CycleRepetition	Cycle Repetition part of the cycle filter mechanism FrIf_LPdu shall be configured to.
	FrIf_CycleOffset	Cycle Offset part of the cycle filter mechanism FrIf_LPdu shall be configured to.
	FrIf_PayloadLength	Payloadlength in units of bytes the FrIf_LPduIdx shall be configured to.
	FrIf_HeaderCRC	Header CRC the FrIf_LPdu shall be configured to.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description	Calls the FlexRay Driver's API Fr_ReconfigLPdu. The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

]

[SWS_FrIf_05422] [The function FrIf_ReconfigLPdu shall be pre compile time configurable ON/OFF by the configuration parameter FrIfReconfigLPduSupport (derived from configuration parameter FrIfReconfigLPduSupport, see ECUC_FrIf_06109)]

[SWS_FrIf_05309] [If parameter FrIf_CtrlIdx of FrIf_ReconfigLPdu has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_ReconfigLPdu shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

[SWS_FrIf_05310] [If parameter FrIf_ChnlIdx of FrIf_ReconfigLPdu has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_ReconfigLPdu shall report development error code FRIF_E_INV_CHNL_IDX to the Det_ReportError service of the DET module.]

[SWS_Frlf_05311] [If parameter `Frlf_LPduldx` of `Frlf_ReconfigLPdu` has an invalid value (i.e. outside of LPdu range or if `FrlfReconfigurable` of this LPdu is not set to TRUE) and development error detection is enabled (i.e. `FrlfDevErrorDetect` equals ON), the `Frlf_ReconfigLPdu` shall report development error code `FRIF_E_INV_LPDU_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_Frlf_05312] [If parameter `Frlf_Frameld` of `Frlf_ReconfigLPdu` has an invalid value and if development error detection is enabled (i.e. `FrlfDevErrorDetect` equals ON), the `Frlf_ReconfigLPdu` shall report development error code `FRIF_E_INV_FRAME_ID` to the `Det_ReportError` service of the DET module.]

8.4.13 Frlf_GetNmVector

[SWS_Frlf_05016] Definition of API function `Frlf_GetNmVector` [

Service Name	<code>Frlf_GetNmVector</code>	
Syntax	<pre>Std_ReturnType Frlf_GetNmVector (uint8 Frlf_CtrlIdx, uint8* Frlf_NmVectorPtr)</pre>	
Service ID [hex]	0x0f	
Sync/Async	Synchronous	
Reentrancy	non reentrant for identical values of <code>Frlf_CtrlIdx</code> , reentrant for different values of <code>Frlf_CtrlIdx</code>	
Parameters (in)	<code>Frlf_CtrlIdx</code>	Index of the FlexRay CC to address.
Parameters (inout)	None	
Parameters (out)	<code>Frlf_NmVectorPtr</code>	Pointer to a memory location where output value will be stored.
Return value	<code>Std_ReturnType</code>	<code>E_OK</code> : The call of the FlexRay Driver's API service has returned <code>E_OK</code> . <code>E_NOT_OK</code> : The call of the FlexRay Driver's API service has returned <code>E_NOT_OK</code> , or an error has been detected in development mode.
Description	Derives the FlexRay NM Vector.	
Available via	<code>Frlf.h</code>	

]

[SWS_Frlf_05423] [The function `Frlf_GetNmVector` shall be pre compile time configurable ON/OFF by the configuration parameter `FrlfGetNmVectorSupport` (derived from configuration parameter `FrlfGetNmVectorSupport`, see [\[ECUC_Frlf_06114\]](#))]

[SWS_Frlf_05197] [If parameter `Frlf_CtrlIdx` of `Frlf_GetNmVector` has an invalid value and if development error detection is enabled (i.e. `FrlfDevErrorDetect` equals ON), the function `Frlf_GetNmVector` shall report development error code `FRIF_E_INV_CTRL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_Frlf_05198] [The function `Frlf_GetNmVector` wraps the FlexRay Driver API `Fr_GetNmVector` function.]

[SWS_Frlf_05199] [Caveats of Frlf_GetNmVector: The FlexRay Interface module has to be initialized with a call of Frlf_Init() before this API service may be called, see [SWS_Frlf_05003]]

8.4.14 Frlf_GetVersionInfo

[SWS_Frlf_05002] Definition of API function Frlf_GetVersionInfo

Upstream requirements: [SRS_BSW_00407](#), [SRS_BSW_00411](#)

[

Service Name	Frlf_GetVersionInfo	
Syntax	<pre>void Frlf_GetVersionInfo (Std_VersionInfoType* Frlf_VersionInfoPtr)</pre>	
Service ID [hex]	0x01	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	Frlf_VersionInfoPtr	Pointer to a memory location where the FlexRay Interface version information shall be stored.
Return value	void	–
Description	Returns the version information of this module.	
Available via	Frlf.h	

]

[SWS_Frlf_05424] [The function Frlf_GetVersionInfo shall be pre compile time configurable ON/OFF by the configuration parameter FrlfVersionInfoApi (derived from configuration parameter FrlfVersionInfoApi, see ECUC_Frlf_06083)]

[SWS_Frlf_05151] [If parameter Frlf_VersionInfoPtr of Frlf_GetVersionInfo equals NULL_PTR and if development error detection is enabled (i.e. FrlfDevErrorDetect equals ON), the function Frlf_GetVersionInfo shall report development error code FRIF_E_PARAM_POINTER to the Det_ReportError service of the DET module.]

8.4.15 Frlf_ReadCCConfig

[SWS_Frlf_05313] Definition of API function Frlf_ReadCCConfig [

Service Name	Frlf_ReadCCConfig	
Syntax	<pre>Std_ReturnType Frlf_ReadCCConfig (uint8 Frlf_CtrlIdx, uint8 Frlf_ConfigParamIdx, uint32* Frlf_ConfigParamValuePtr)</pre>	





Service ID [hex]	0x3b	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ConfigParamIdx	Index of the configuration parameter to read.
Parameters (inout)	None	
Parameters (out)	FrIf_ConfigParamValuePtr	Pointer to a memory location where output value will be stored.
Return value	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description	Wraps the FlexRay Driver API function Fr_ReadCCConfig().	
Available via	FrIf.h	

]

[SWS_FrIf_05314] [The function FrIf_ReadCCConfig wraps the FlexRay Driver API Fr_ReadCCConfig function.]

[SWS_FrIf_05315] [If parameter FrIf_CtrlIdx of FrIf_ReadCCConfig has an invalid value and if development error detection is enabled (i.e. FrIfDevErrorDetect equals ON), the function FrIf_ReadCCConfig shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.]

8.4.16 FrIf_EnableBusMirroring

[SWS_FrIf_05726] Definition of API function FrIf_EnableBusMirroring [

Service Name	FrIf_EnableBusMirroring	
Syntax	Std_ReturnType FrIf_EnableBusMirroring (uint8 FrIf_ClstIdx, boolean FrIf_MirroringActive)	
Service ID [hex]	0x4b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_ClstIdx	Index of the FlexRay cluster to address.
	FrIf_MirroringActive	TRUE: Mirror_ReportFlexRayFrame will be called for each frame received or transmitted on the addressed FlexRay CC. FALSE: Mirror_ReportFlexRayFrame will not be called for the addressed FlexRay CC.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Mirroring mode was changed. E_NOT_OK: Wrong FrIf_CtrlIdx, or mirroring is globally disabled (see FrIfBusMirroringSupport).





Description	Enables or disables mirroring for all FlexRay controllers connected to the addressed FlexRay cluster.
Available via	FrIf.h

]

[SWS_FrIf_05727] [The function `FrIf_EnableBusMirroring` shall be pre compile time configurable ON/OFF by the configuration parameter `FrIfBusMirroringSupport` (see `ECUC_FrIf_06124`).]

8.5 Interrupt Service Routines

8.5.1 `FrIf_JobListExec_<FrIfCluster.ShortName>`

[SWS_FrIf_05040] Definition of API function `FrIf_JobListExec_<FrIfCluster.ShortName>` [

Service Name	<code>FrIf_JobListExec_<FrIfCluster.ShortName></code>
Syntax	<pre>void FrIf_JobListExec_<FrIfCluster.ShortName> (void)</pre>
Service ID [hex]	0x32
Sync/Async	Synchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None
Parameters (out)	None
Return value	None
Description	Processes the FlexRay Job List of the FlexRay Cluster with index <code>CstIdx</code> .
Available via	FrIf.h

]

Note:

For a detailed description of this API service, please refer to chapter [Chapter 7.6.4.2](#).

[SWS_FrIf_05270] [The function `FrIf_JobListExec_<FrIfCluster.ShortName>` shall exist once per FlexRay Cluster of a FlexRay Interface module.]

[SWS_FrIf_05271] [The function name of each instance of `FrIf_JobListExec_<FrIfCluster.ShortName>` shall contain the short name of the respective FlexRay Cluster (`FrIfCluster`).

For each FlexRay Cluster (identified by index `CstIdx`), the respective API service `FrIf_JobListExec_<FrIfCluster.ShortName>` must be registered in the AUTOSAR OS as the ISR of an absolute timer of a FlexRay CC connected to the FlexRay Cluster with index `CstIdx`, if the CC does not guarantee asynchronous buffer access.]

Note: If the CC guarantees asynchronous buffer access, the execution of `FrIf_JobListExec<FrIfCluster.ShortName>` can run in a regular OS task.

[SWS_FrIf_05272] [Caveats of `FrIf_JobListExec_<FrIfCluster.ShortName>`: The FlexRay Interface module has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[SWS_FrIf_05003\]](#).]

8.6 Callback notifications

This is a list of functions provided for other modules.

8.6.1 FrIf_CheckWakeupByTransceiver

[SWS_FrIf_05041] Definition of API function `FrIf_CheckWakeupByTransceiver` [

Service Name	FrIf_CheckWakeupByTransceiver	
Syntax	<pre>void FrIf_CheckWakeupByTransceiver (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx)</pre>	
Service ID [hex]	0x39	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Wraps the FlexRay Transceiver Driver API function <code>FrTrcv_CheckWakeupByTransceiver()</code> . The enum value "FR_CHANNEL_AB" shall not be used.	
Available via	FrIf.h	

]

[SWS_FrIf_05274] [If parameter `FrIf_CtrlIdx` of `FrIf_CheckWakeupByTransceiver` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_CheckWakeupByTransceiver` shall report development error code `FRIF_E_INV_CTRL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_FrIf_05275] [If parameter `FrIf_ChnlIdx` of `FrIf_CheckWakeupByTransceiver` has an invalid value and if development error detection is enabled (i.e. `FrIfDevErrorDetect` equals ON), the function `FrIf_CheckWakeupByTransceiver` shall report development error code `FRIF_E_INV_CHNL_IDX` to the `Det_ReportError` service of the DET module.]

[SWS_Frlf_05276] [The function `Frlf_CheckWakeupByTransceiver` shall wrap the FlexRay Transceiver Driver API function `FrTrcv_CheckWakeupByTransceiver()` by:

1. Translating (based on static `Frlf` module configuration) the tuple (FlexRay CC index `Frlf_CtrlIdx` | FlexRay Channel index `Frlf_ChnlIdx`) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index `FrTrcv_TrvcIdx`).
2. Calling `FrTrcv_CheckWakeupByTransceiver()` of the determined FlexRay Driver module with the parameters determined as described above.

]

[SWS_Frlf_05277] [Caveats of `Frlf_CheckWakeupByTransceiver`: The FlexRay Interface module has to be initialized with a call of `Frlf_Init()` before this API service may be called, see `SWS_Frlf_05003`.]

8.7 Scheduled functions

8.7.1 `Frlf_MainFunction_<FrlfCluster.ShortName>`

[SWS_Frlf_05042] Definition of scheduled function `Frlf_MainFunction_<FrlfCluster.ShortName>` [

Service Name	<code>Frlf_MainFunction_<FrlfCluster.ShortName></code>
Syntax	<pre>void Frlf_MainFunction_<FrlfCluster.ShortName> (void)</pre>
Service ID [hex]	0x27
Description	This function will be called cyclically by a task body provided by the BSW Scheduler.
Available via	<code>SchM_Frlf.h</code>

]

Note:

This cyclically executed API service of the FlexRay Interface serves the following purposes:

- Program the absolute timer interrupt in order to start the execution of `Frlf_JobListExec_<FrlfCluster.ShortName>()` if the CC does not support asynchronous buffer access.
- Monitoring the proper (in time) execution of the `Frlf_JobListExec_<FrlfCluster.ShortName>()` and resynchronize the Joblist if necessary.

Please refer to chapter [Chapter 7.3](#) for a detailed description.

Pre condition: The function `Frlf_MainFunction_<FrlfCluster.ShortName>` is cyclically called from a task body provided by the BSW Scheduler module.

Since the duration of a FlexRay Cycle may be different for two Clusters of an ECU, the calling period (parameter `FrIfMainFunctionPeriod`) of this API service shall be configurable independently for each Cluster at system configuration time.

The parameter `FrIfMainFunctionPeriod` determines for each FlexRay cluster of a FlexRay Interface module the calling period, which is provided for the BSW scheduler module.

[SWS_FrIf_05278] [The function `FrIf_MainFunction_<FrIfCluster.ShortName>` shall exist once per FlexRay Cluster of a FlexRay Interface module.]

[SWS_FrIf_05279] [The function name of each instance of `FrIf_MainFunction_<FrIfCluster.ShortName>` shall contain the short name of the respective FlexRay Cluster (`FrIfCluster`).]

8.8 Expected interfaces

This chapter lists all API services required from other BSW modules.

8.8.1 Mandatory Interfaces

This chapter defines all API services which are required from other BSW modules to fulfill the core functionality of the FlexRay Interface.

[SWS_FrIf_05043] Definition of mandatory interfaces required by module FrIf [

API Function	Header File	Description
<code>Det_ReportRuntimeError</code>	<code>Det.h</code>	Service to report runtime errors. If a callout has been configured then this callout shall be called.
<code>Fr_AbortCommunication</code>	<code>Fr.h</code>	Invokes the CC CHI command 'FREEZE'.
<code>Fr_AckAbsoluteTimerIRQ</code>	<code>Fr.h</code>	Resets the interrupt condition of an absolute timer.
<code>Fr_AllowColdstart</code>	<code>Fr.h</code>	Invokes the CC CHI command 'ALLOW_COLDSTART'.
<code>Fr_CancelAbsoluteTimer</code>	<code>Fr.h</code>	Stops an absolute timer.
<code>Fr_CheckTxLPduStatus</code>	<code>Fr.h</code>	Checks the transmit status of the LSdu.
<code>Fr_ControllerInit</code>	<code>Fr.h</code>	Initializes a FlexRay CC.
<code>Fr_DisableAbsoluteTimerIRQ</code>	<code>Fr.h</code>	Disables the interrupt line of an absolute timer.
<code>Fr_EnableAbsoluteTimerIRQ</code>	<code>Fr.h</code>	Enables the interrupt line of an absolute timer.
<code>Fr_GetAbsoluteTimerIRQStatus</code>	<code>Fr.h</code>	Gets IRQ status of an absolute timer.
<code>Fr_GetGlobalTime</code>	<code>Fr.h</code>	Gets the current global FlexRay time. Important Note: <code>Fr_GetGlobalTime</code> may be called within an exclusive area.
<code>Fr_GetPOCStatus</code>	<code>Fr.h</code>	Gets the POC status.
<code>Fr_HaltCommunication</code>	<code>Fr.h</code>	Invokes the CC CHI command 'DEFERRED_HALT'.
<code>Fr_ReceiveRxLPdu</code>	<code>Fr.h</code>	Receives data from the FlexRay network.





API Function	Header File	Description
Fr_SendWUP	Fr.h	Invokes the CC CHI command 'WAKEUP'.
Fr_SetAbsoluteTimer	Fr.h	Sets the absolute FlexRay timer.
Fr_SetWakeupChannel	Fr.h	Sets a wakeup channel.
Fr_StartCommunication	Fr.h	Starts communication.
Fr_TransmitTxLPdu	Fr.h	Transmits data on the FlexRay network.
FrTrcv_CheckWakeupByTransceiver	FrTrcv.h	--
FrTrcv_ClearTransceiverWakeup	FrTrcv.h	This function clears a pending wake up event.
FrTrcv_GetTransceiverMode	FrTrcv.h	This function returns the actual state of the transceiver.
FrTrcv_GetTransceiverWUReason	FrTrcv.h	This function returns the wakeup reason.
FrTrcv_SetTransceiverMode	FrTrcv.h	This service sets the transceiver mode.

]

8.8.2 Optional Interfaces

This chapter defines all API services which are required from other BSW modules to fulfill an optional functionality of the FlexRay Interface

[SWS_FrIf_05044] Definition of optional interfaces requested by module FrIf [

API Function	Header File	Description
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
Fr_AllSlots	Fr.h	Invokes the CC CHI command 'ALL_SLOTS'.
Fr_CancelTxLPdu	Fr.h	Cancels the already pending transmission of a LPdu contained in a controllers physical transmit resource (e.g. message buffer).
Fr_DisableLPdu	Fr.h	Disables the hardware resource of a LPdu for transmission/reception.
Fr_GetChannelStatus	Fr.h	Gets the channel status information.
Fr_GetClockCorrection	Fr.h	Gets the current clock correction values. See variables vInterimRateCorrection and vInterimOffset Correction of [12] for details.
Fr_GetNmVector	Fr.h	Gets the network management vector of the last communication cycle.
Fr_GetNumOfStartupFrames	Fr.h	Gets the current number of startup frames seen on the cluster. See variable vStartupPairs of [12] for details.
Fr_GetSyncFrameList	Fr.h	Gets a list of syncframes received or transmitted on channel A and channel B via the even and odd communication cycle. See variables vsSyncIdListA and vsSyncIdListB of [12] for details.
Fr_GetWakeupRxStatus	Fr.h	Gets the wakeup received information from the Flex Ray controller.





API Function	Header File	Description
Fr_PrepareLPdu	Fr.h	Prepares a LPdu.
Fr_ReadCCConfig	Fr.h	Reads a FlexRay protocol configuration parameter for a particular FlexRay controller out of the module's configuration.
Fr_ReconfigLPdu	Fr.h	Reconfigures a given LPdu according to the parameters (FrameId, Channel, CycleRepetition, CycleOffset, PayloadLength, HeaderCRC) at runtime.
FrTrcv_DisableTransceiverBranch	FrTrcv.h	This function disables the specified branch on the addressed (active star) transceiver.
FrTrcv_EnableTransceiverBranch	FrTrcv.h	This function enables the specified branch on the addressed (active star) transceiver.
FrTrcv_GetTransceiverError	FrTrcv.h	All mandatory errors defined by the FlexRay EPL [5] which are supported by the FlexRay transceiver hardware can be accessed via this API. In addition to errors on the physical layer and local to the ECU hardware, a global error flag is provided.
LSduR_FrlfRxIndication (draft)	LSduR_Frlf.h	Indication of a received PDU from a lower layer communication interface module.
LSduR_FrlfTriggerTransmit (draft)	LSduR_Frlf.h	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.
LSduR_FrlfTxConfirmation (draft)	LSduR_Frlf.h	The lower layer communication interface module confirms the transmission of a PDU, or the failure to transmit a PDU.
Mirror_ReportFlexRayFrame	Mirror.h	Reports a received or transmitted FlexRay frame or a Tx conflict.

8.8.3 Configurable Interfaces

8.8.3.1 <Free_Op_A>

[SWS_Frlf_05316] Definition of configurable interface <Free_Op_A> [

Service Name	<Free_Op_A>	
Syntax	<pre>void <Free_Op_A> (uint8 Frlf_CtrlIdx, uint16 Frlf_LPduIdx)</pre>	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different Frlf_LPduIdx, non reentrant for same Frlf_LPduIdx	
Parameters (in)	Frlf_CtrlIdx	Index of the FlexRay CC to address.
	Frlf_LPduIdx	This index is used to uniquely identify a FlexRay frame.
Parameters (inout)	None	





Parameters (out)	None
Return value	None
Description	User defined communication operation in order to support hardware specific or additional communication controller features to increase performance.
Available via	FrIf_Externals.h

]

Caveats of <Free_Op_A>: This API service is called during the execution of the Flex Ray Job List Execution Function.

8.8.3.2 <Free_Op_B>

[SWS_FrIf_05317] Definition of configurable interface <Free_Op_B> [

Service Name	<Free_Op_B>	
Syntax	<pre>void <Free_Op_B> (uint8 FrIf_CtrlIdx, uint16 FrIf_LPduIdx)</pre>	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different FrIf_LPduIdx, non reentrant for same FrIf_LPduIdx	
Parameters (in)	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_LPduIdx	This index is used to uniquely identify a FlexRay frame.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	User defined communication operation in order to support hardware specific or additional communication controller features to increase performance.	
Available via	FrIf_Externals.h	

]

Caveats of <Free_Op_B>: This API service is called during the execution of the Flex Ray Job List Execution Function.

8.8.3.3 <UL_TxConflictNotification>

[SWS_FrIf_91001] Definition of configurable interface <UL_TxConflictNotification> [

Service Name	<UL_TxConflictNotification>	
Syntax	<pre>void <UL_TxConflictNotification> (uint8 FrIf_CtrlIdx, uint16 FrIf_LPduIdx)</pre>	
Sync/Async	Synchronous	





Reentrancy	Reentrant for different FrLf_LPdIdx. Non reentrant for the same FrLf_LPdIdx.	
Parameters (in)	FrLf_CtrlIdx	ID of the addressed FlexRay CC
	FrLf_LPdIdx	ID of the transmitted FlexRay frame
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notification in case a TxConflict has been detected.	
Available via	FrLf_Externals.h	

└

9 Sequence diagrams

The sequence diagrams in this chapter show the basic operations carried out in a FlexRay Cluster's FlexRay Job List Execution Function when executing the various Communication Operations. They also show the interaction of the FrIf with the upper layer BSW module and with the underlying FlexRay Driver.

Please note that the sequence diagrams are an extension for illustrational purposes to ease understanding of the specification.

9.1 Data Transmission

9.1.1 TransmitWithImmediateBufferAccess

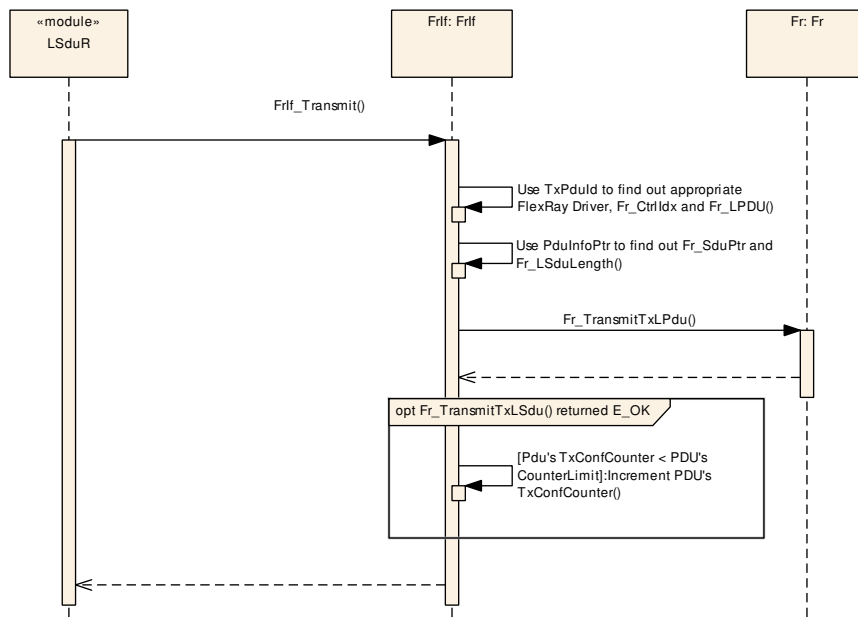


Figure 9.1: Transmit With Immediate Buffer Access

9.1.2 TransmitWithDecoupledBufferAccess

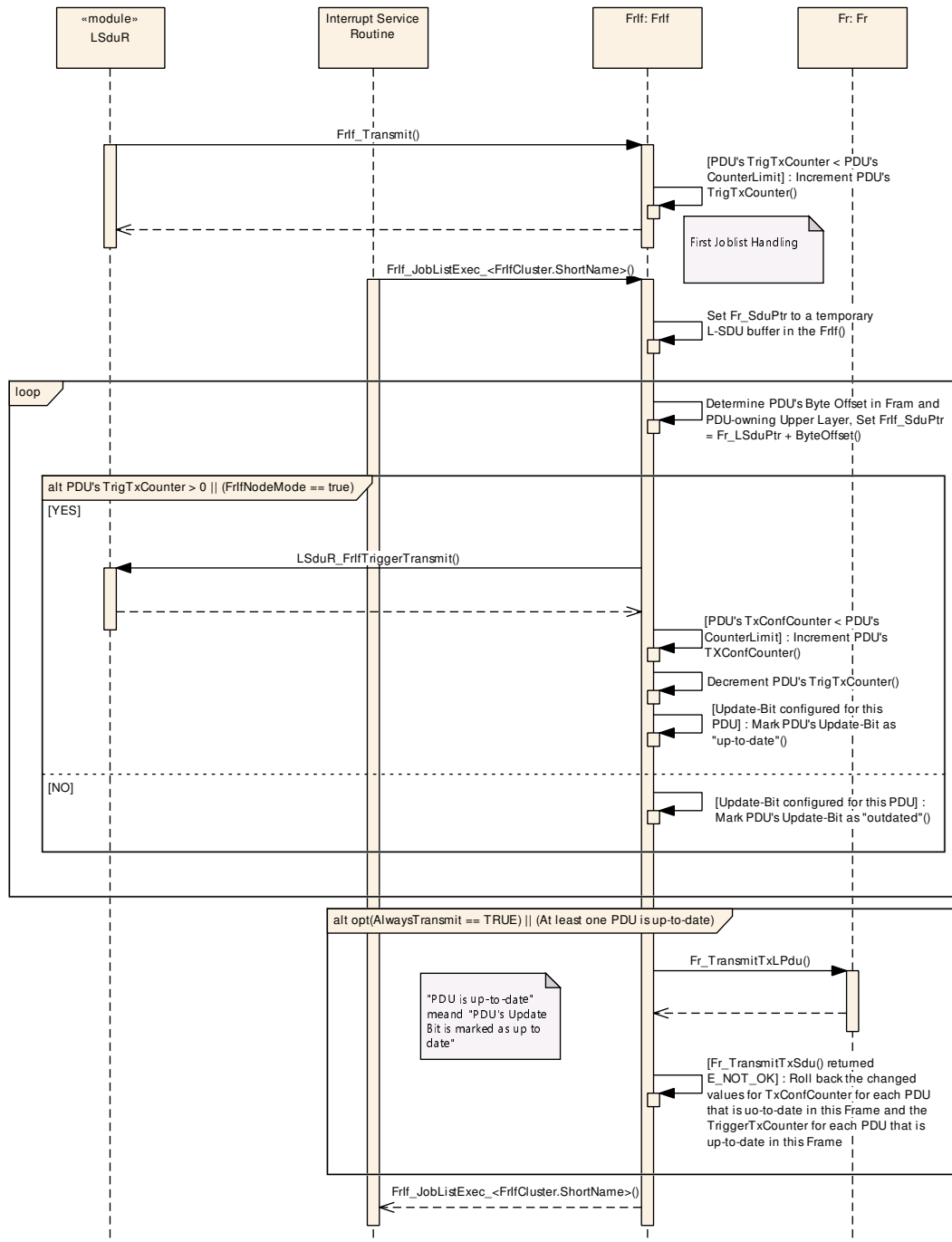


Figure 9.2: Transmit With Decoupled Buffer Access

9.1.3 ProvideTxConfirmation

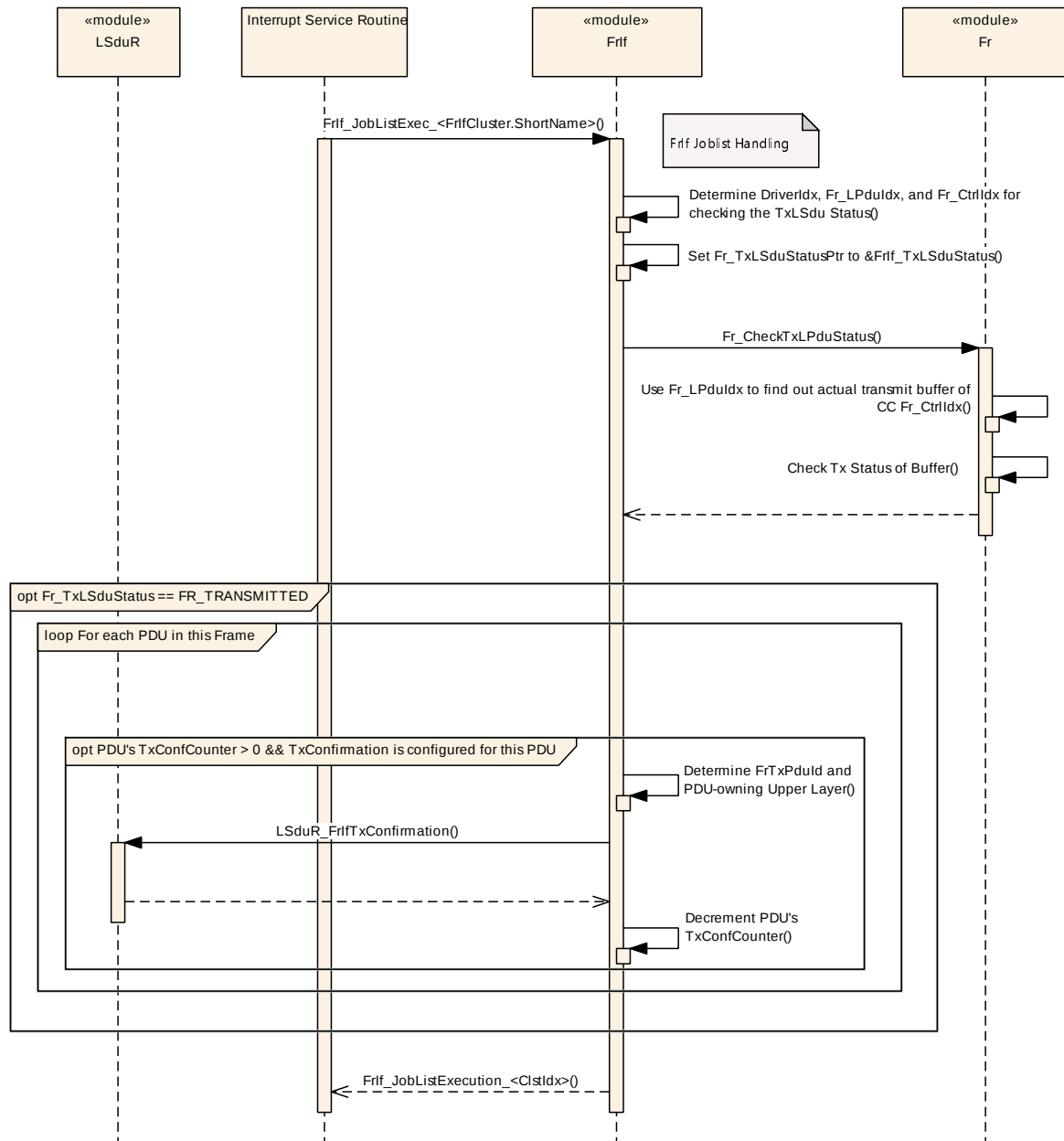


Figure 9.3: Provide TX Confirmation

9.2 Data Reception

9.2.1 ReceiveAndIndicate

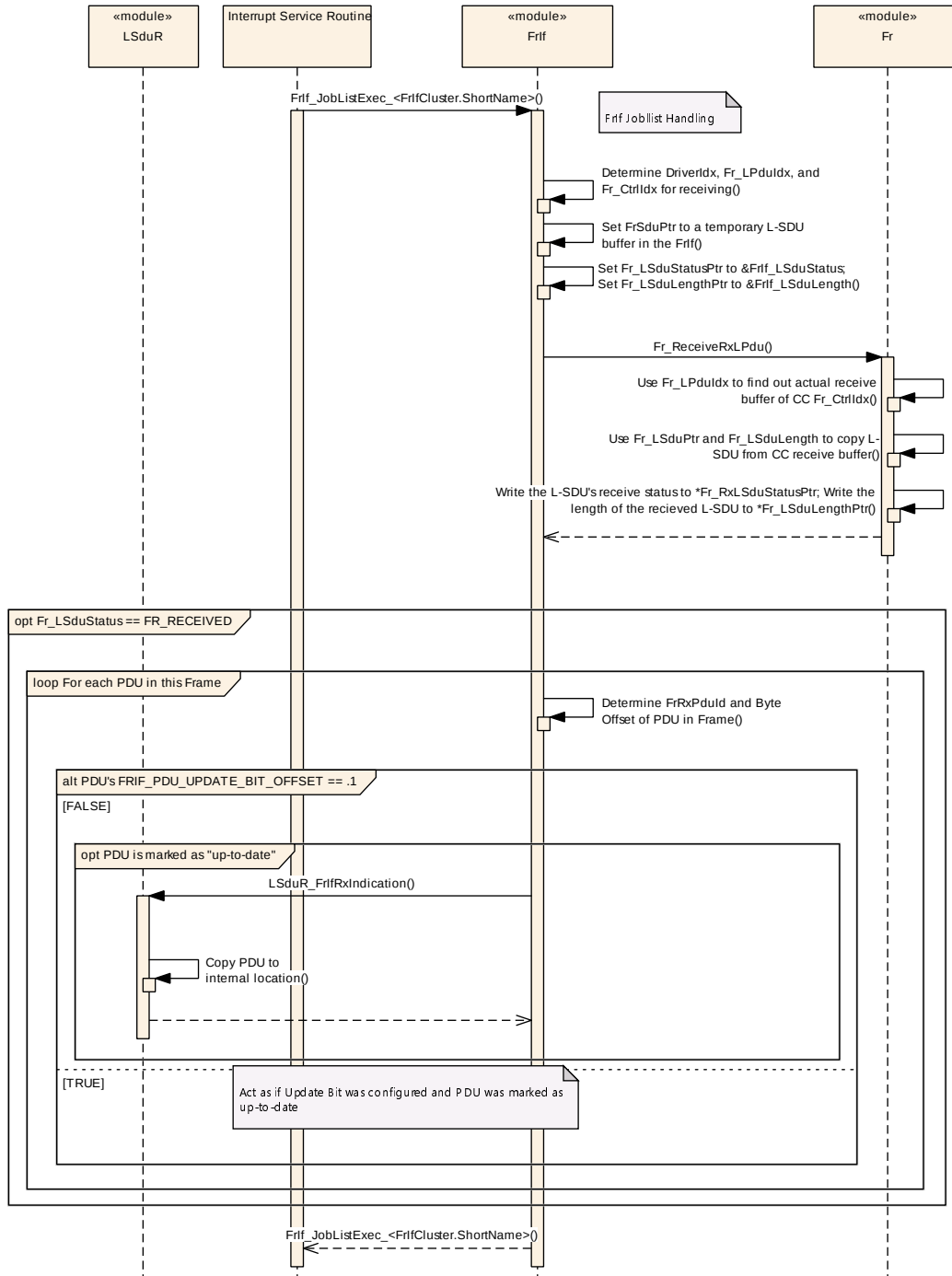


Figure 9.4: Receive and Indicate

9.2.2 ReceiveAndStore

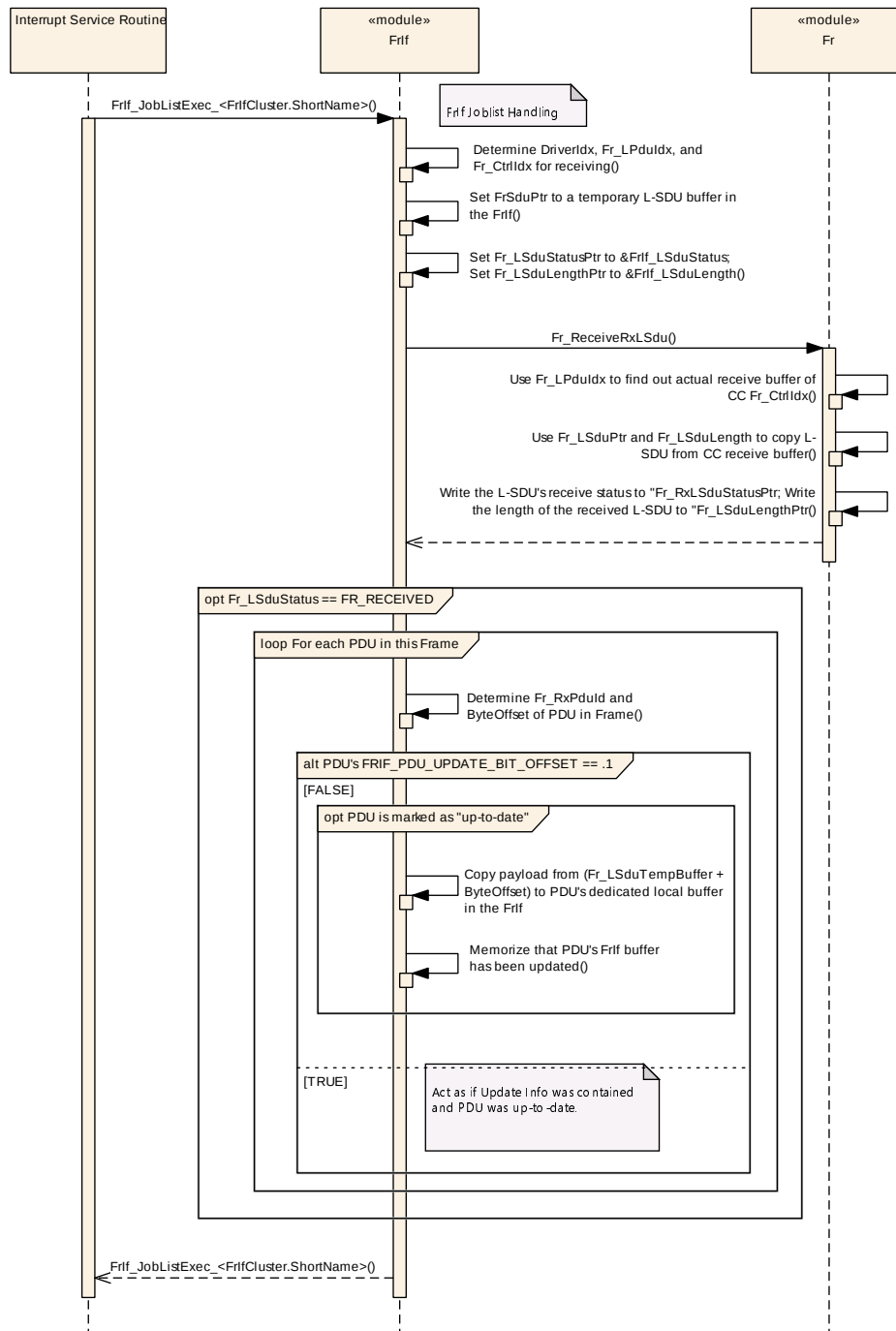


Figure 9.5: Receive and Store

9.2.3 ProvideRxIndication

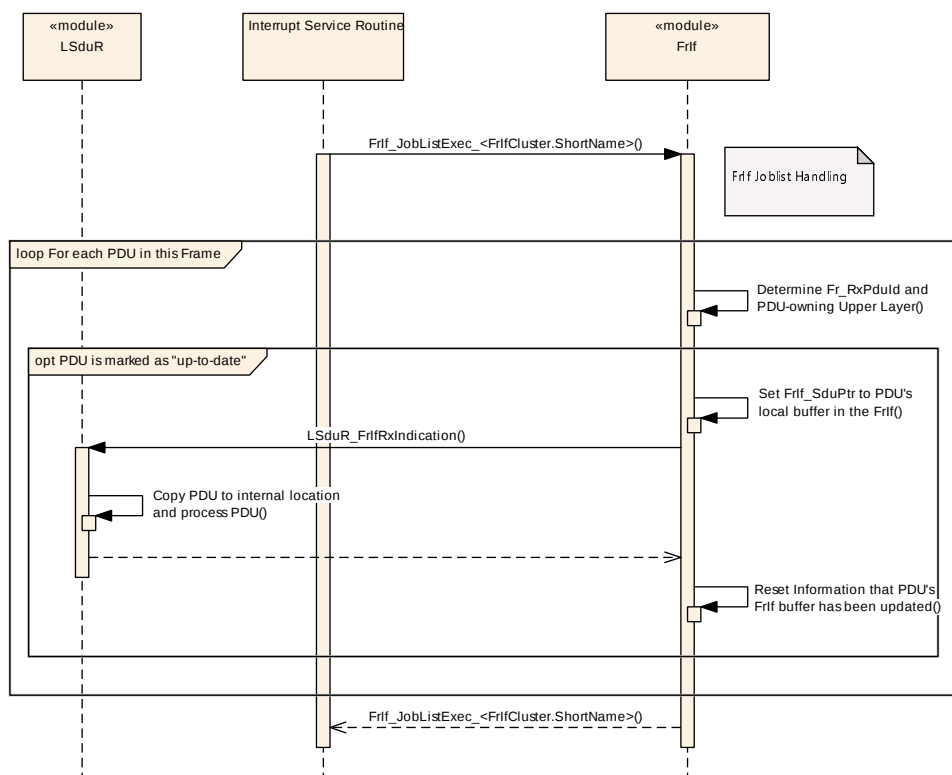


Figure 9.6: Provide Rx Indication

9.2.4 Cancel Transmission

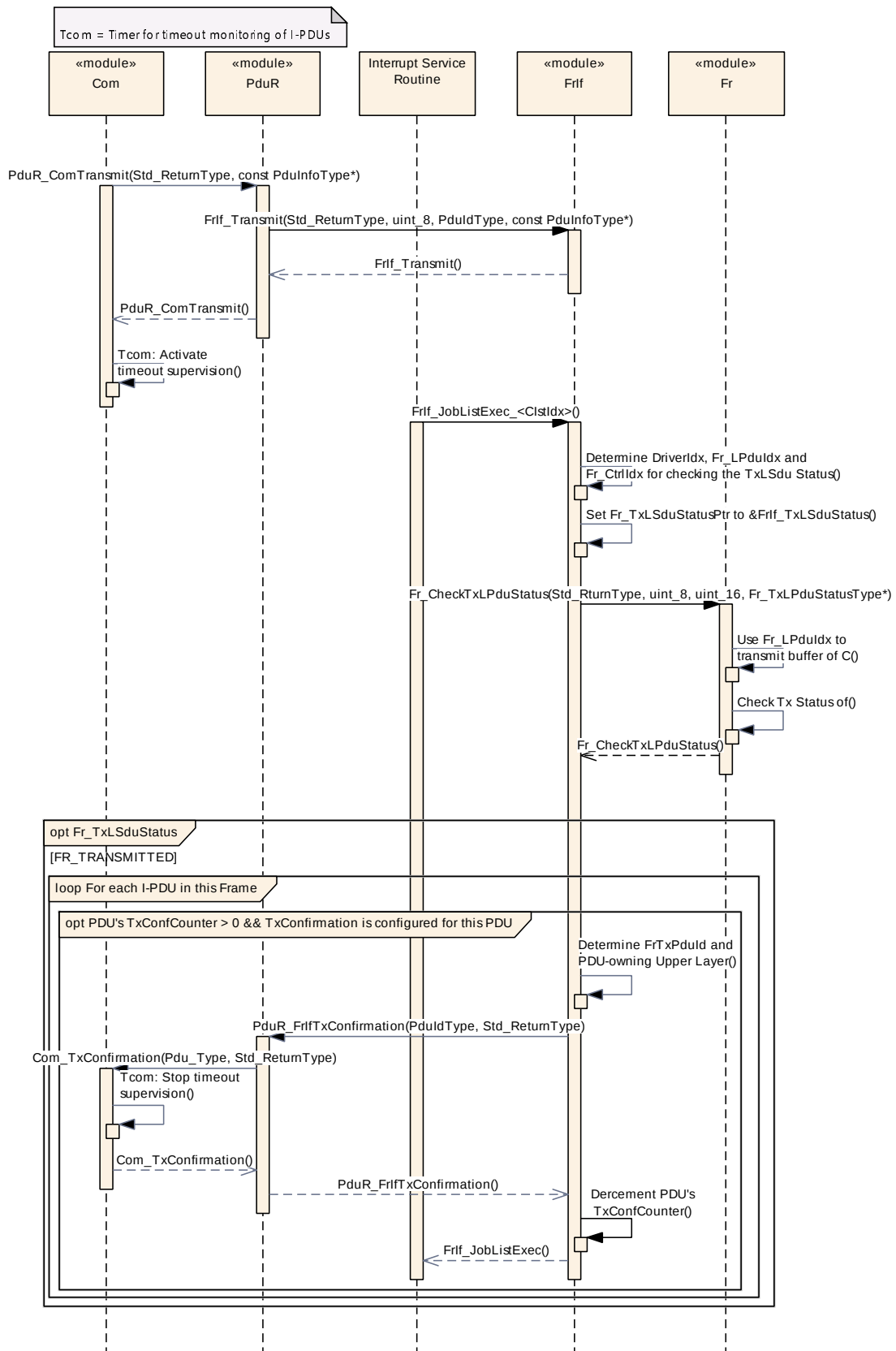


Figure 9.7: Cancel Transmission Part 1

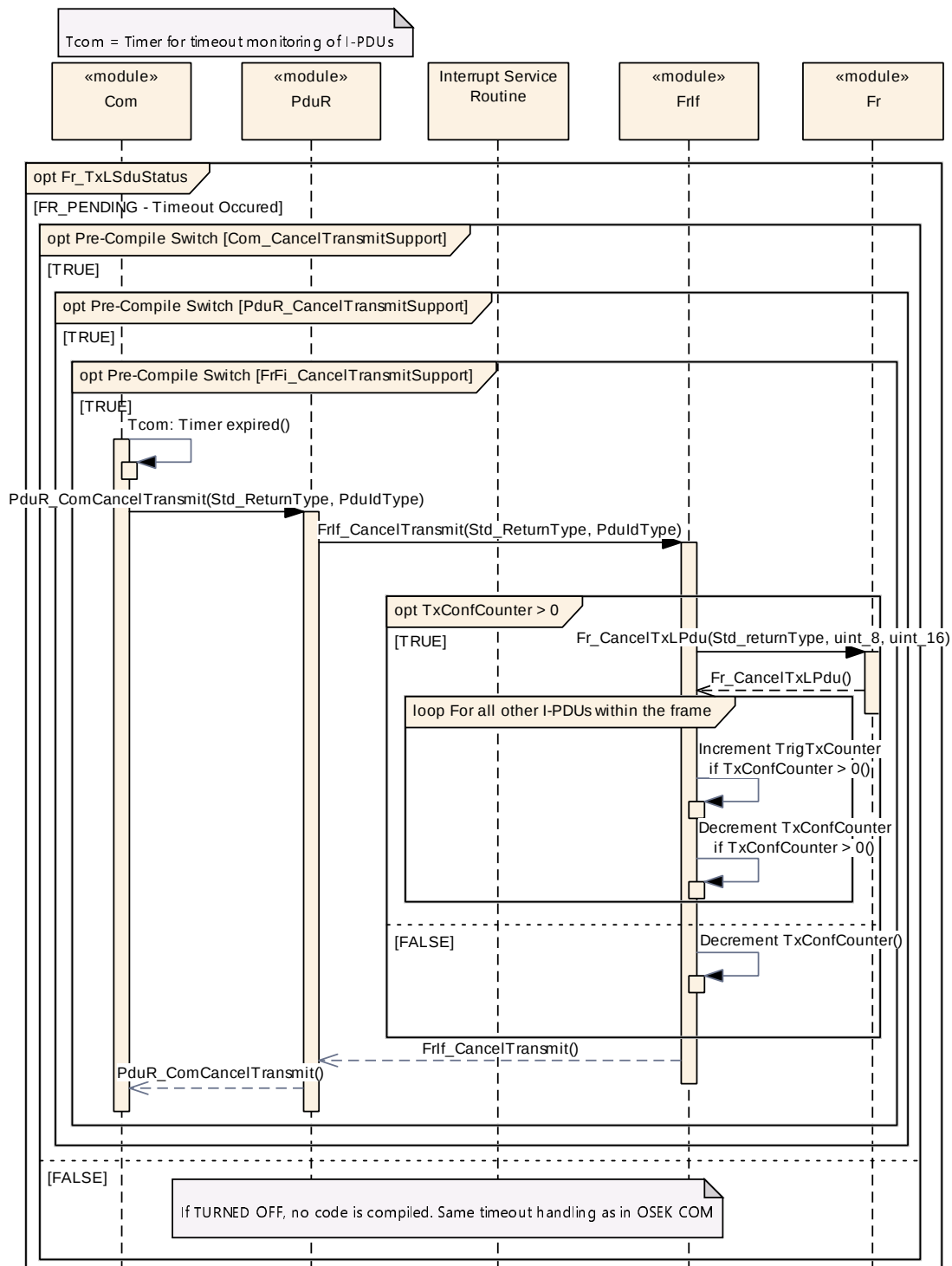


Figure 9.8: Cancel Transmission Part 2

9.3 Prepare LPDU

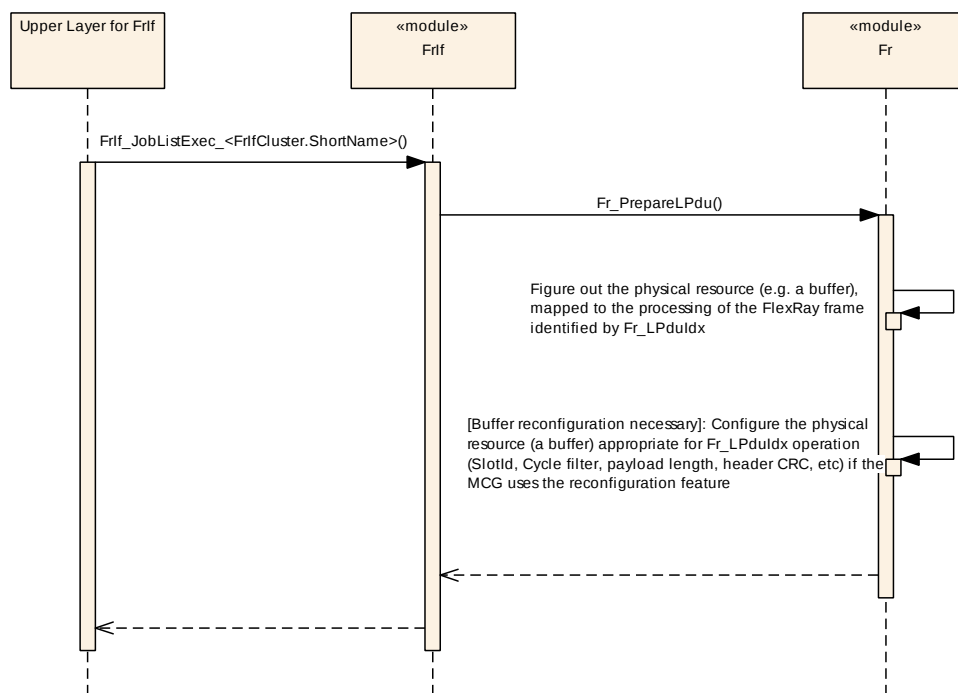


Figure 9.9: Prepare LPdu

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module FrIf.

Chapter 10.3 specifies published information of the module FrIf.

10.1 How to read this chapter

For details refer to [2] Chapter 10.1 *“Introduction to configuration specification”*.

10.2 Containers and configuration parameters

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

The listed configuration items can be derived from a network description database, which is based on the EcuConfigurationTemplate. The configuration tool has to extract all information to configure the FrIf module.

Note:

The configuration tool must check the consistency of the configuration at configuration time.

Note:

These dependencies between FlexRay Interface and FlexRay Driver configuration must be provided at configuration time by the configuration tools.

10.2.1 FrIf

[ECUC_FrIf_06087] Definition of EcucModuleDef FrIf [

Module Name	FrIf
Description	Configuration of the FrIf (FlexRay Interface) module.
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	Multiplicity	Dependency
FrIfConfig	1	This container contains the configuration parameters and sub containers of the AUTOSAR FrIf module.
FrIfGeneral	1	This container contains the general configuration parameters of the FlexRay Interface.

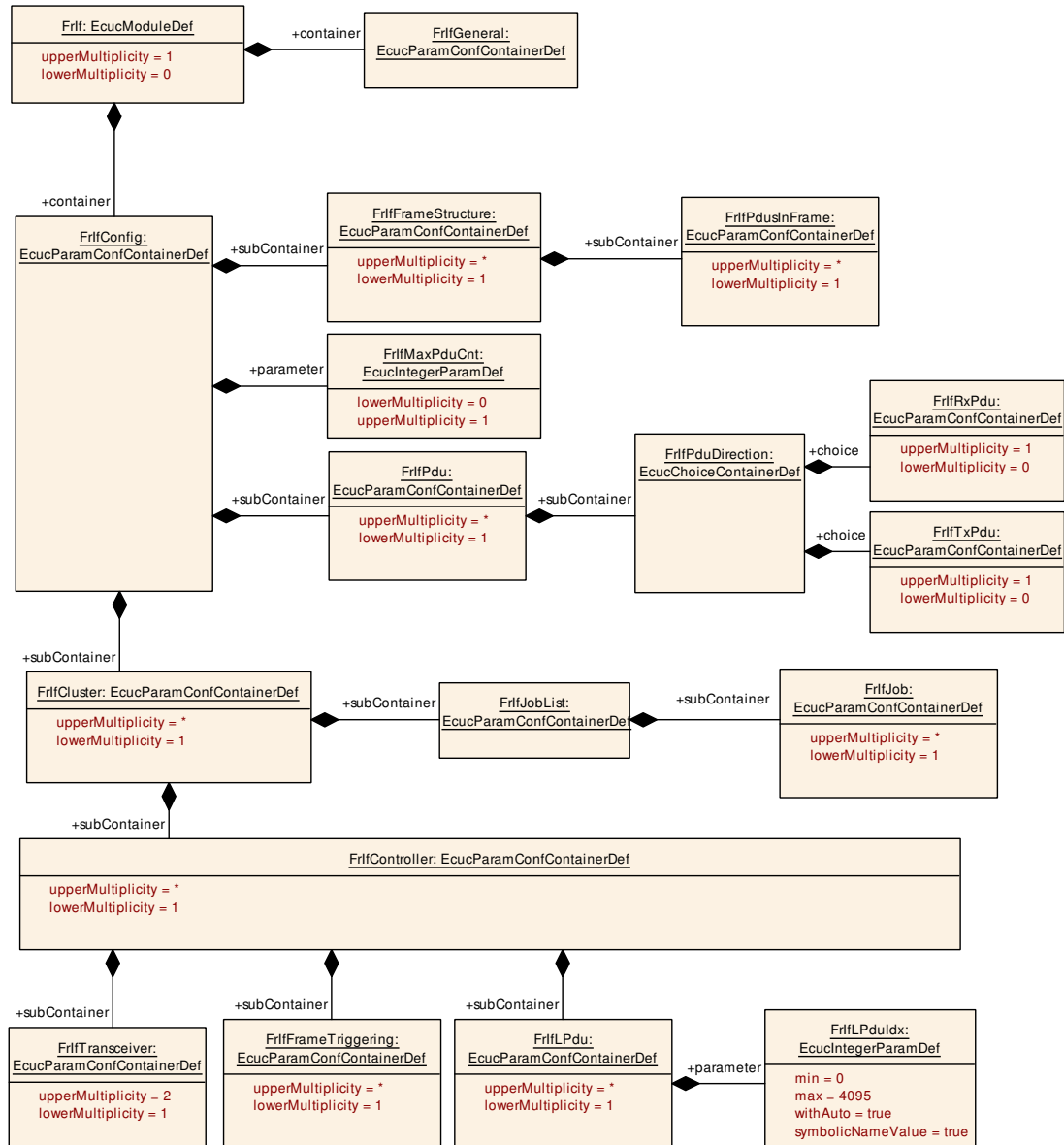


Figure 10.1: FlexRay Interface Module

10.2.2 FrIfGeneral

[ECUC_FrIf_05360] Definition of EcucParamConfContainerDef FrIfGeneral [

Container Name	FrlfGeneral
Parent Container	Frlf
Description	This container contains the general configuration parameters of the FlexRay Interface.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrlfAbsTimerIdx	1	[ECUC_Frlf_06112]
FrlfAllSlotsSupport	1	[ECUC_Frlf_06108]
FrlfBusMirroringSupport	1	[ECUC_Frlf_06124]
FrlfCancelTransmitSupport	1	[ECUC_Frlf_00002]
FrlfDevErrorDetect	1	[ECUC_Frlf_06080]
FrlfDisableLPduSupport	1	[ECUC_Frlf_06110]
FrlfDisableTransceiverBranchSupport	1	[ECUC_Frlf_06102]
FrlfEnableTransceiverBranchSupport	1	[ECUC_Frlf_06103]
FrlfFreeOpAApiName	0..1	[ECUC_Frlf_06118]
FrlfFreeOpBApiName	0..1	[ECUC_Frlf_06119]
FrlfFreeOpsHeader	0..1	[ECUC_Frlf_06120]
FrlfGetClockCorrectionSupport	1	[ECUC_Frlf_06106]
FrlfGetGetChannelStatusSupport	1	[ECUC_Frlf_06105]
FrlfGetNmVectorSupport	1	[ECUC_Frlf_06114]
FrlfGetNumOfStartupFramesSupport	1	[ECUC_Frlf_06104]
FrlfGetSyncFrameListSupport	1	[ECUC_Frlf_06107]
FrlfGetTransceiverErrorSupport	1	[ECUC_Frlf_06101]
FrlfGetWakeupRxStatusSupport	1	[ECUC_Frlf_06111]
FrlfNumClstSupported	1	[ECUC_Frlf_06081]
FrlfNumCtrlSupported	1	[ECUC_Frlf_06082]
FrlfPublicCddHeaderFile	0..*	[ECUC_Frlf_06116]
FrlfReadCCConfigApi	1	[ECUC_Frlf_06117]
FrlfReconfigLPduSupport	1	[ECUC_Frlf_06109]
FrlfTxConflictNotificationHeaderName	0..1	[ECUC_Frlf_06123]
FrlfTxConflictNotificationName	0..1	[ECUC_Frlf_06122]
FrlfUnusedBitValue	0..1	[ECUC_Frlf_00001]
FrlfVersionInfoApi	1	[ECUC_Frlf_06083]

No Included Containers

]

[ECUC_Frlf_06112] Definition of EcucIntegerParamDef FrlfAbsTimerIdx [

Parameter Name	FrlfAbsTimerIdx
Parent Container	FrlfGeneral
Description	Maximum number of supported absolute timers.
Multiplicity	1



△

Type	EcucIntegerParamDef		
Range	1 .. 15		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06108] Definition of EcucBooleanParamDef FrlfAllSlotsSupport [

Parameter Name	FrlfAllSlotsSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable/disable of switching from key-slot / single-slot mode to all slot mode.		
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	–	
Dependency			

[ECUC_Frlf_06124] Definition of EcucBooleanParamDef FrlfBusMirroringSupport [

Parameter Name	FrlfBusMirroringSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable/disable reporting received/transmitted frames to the Bus Mirroring module.		
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	–	
Dependency			

[ECUC_Frlf_00002] Definition of EcucBooleanParamDef FrlfCancelTransmitSupport

Parameter Name	FrlfCancelTransmitSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to request the cancellation of the I-PDU transmission to FrDrv.		
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	–	
Dependency			

[ECUC_Frlf_06080] Definition of EcucBooleanParamDef FrlfDevErrorDetect

Parameter Name	FrlfDevErrorDetect		
Parent Container	FrlfGeneral		
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	–	
Dependency			

[ECUC_Frlf_06110] Definition of EcucBooleanParamDef FrlfDisableLPduSupport

Parameter Name	FrlfDisableLPduSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to disables the hardware resource of a LPdu for transmission/reception.		
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	–	
Dependency			

[ECUC_Frlf_06102] Definition of EcucBooleanParamDef FrlfDisableTransceiverBranchSupport [

Parameter Name	FrlfDisableTransceiverBranchSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to disable branches of an active star.		
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	–	
Dependency			

]

[ECUC_Frlf_06103] Definition of EcucBooleanParamDef FrlfEnableTransceiverBranchSupport [

Parameter Name	FrlfEnableTransceiverBranchSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable branches of an active star.		
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	–	
Dependency			

]

[ECUC_Frlf_06118] Definition of EcucStringParamDef FrlfFreeOpAApiName [

Parameter Name	FrlfFreeOpAApiName		
Parent Container	FrlfGeneral		
Description	API name that is called when FREE_OP_A is selected as communication operation. See also chapter 8.8.3 Configurable Interfaces.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	





Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06119] Definition of EcucStringParamDef FrlfFreeOpBApiName [

Parameter Name	FrlfFreeOpBApiName		
Parent Container	FrlfGeneral		
Description	API name that is called when FREE_OP_B is selected as communication operation. See also chapter 8.8.3 Configurable Interfaces.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06120] Definition of EcucStringParamDef FrlfFreeOpsHeader [

Parameter Name	FrlfFreeOpsHeader		
Parent Container	FrlfGeneral		
Description	Defines header file for configurable FREE_OP_A / FREE_OP_B functions.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06106] Definition of EcucBooleanParamDef FrlfGetClockCorrectionSupport

Parameter Name	FrlfGetClockCorrectionSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable/disable of polling the FlexRay Driver to getting CC clock correction values.		
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	–	
Dependency			

[ECUC_Frlf_06105] Definition of EcucBooleanParamDef FrlfGetGetChannelStatusSupport

Parameter Name	FrlfGetGetChannelStatusSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable/disable of polling the FlexRay Driver to getting error information about the FlexRay communications bus.		
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	–	
Dependency			

[ECUC_Frlf_06114] Definition of EcucBooleanParamDef FrlfGetNmVectorSupport

Parameter Name	FrlfGetNmVectorSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to request the FlexRay hardware NMVector.		
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	–	





Dependency	
------------	--

└

[ECUC_Frlf_06104] Definition of EcucBooleanParamDef FrlfGetNumOfStartupFramesSupport

Parameter Name	FrlfGetNumOfStartupFramesSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable/disable of polling the FlexRay Driver for the actual number of received startup frames on the bus.		
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	–	
Dependency			

└

[ECUC_Frlf_06107] Definition of EcucBooleanParamDef FrlfGetSyncFrameListSupport

Parameter Name	FrlfGetSyncFrameListSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to enable/disable of polling the FlexRay Driver to getting a list of actual received sync frames.		
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	–	
Dependency			

└

[ECUC_Frlf_06101] Definition of EcucBooleanParamDef FrlfGetTransceiverErrorSupport

Parameter Name	FrlfGetTransceiverErrorSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to get the FlexRay Transceiver errors by calling the FlexRay Transceiver module.		
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	–	
Dependency			

[ECUC_Frlf_06111] Definition of EcucBooleanParamDef FrlfGetWakeupRxStatus Support

Parameter Name	FrlfGetWakeupRxStatusSupport		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable Frlf support to get the wakeup received information from the FlexRay controller.		
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	–	
Dependency			

[ECUC_Frlf_06081] Definition of EcucIntegerParamDef FrlfNumClstSupported

Parameter Name	FrlfNumClstSupported		
Parent Container	FrlfGeneral		
Description	Maximum number of FlexRay Clusters that the FlexRay Interface supports.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 15		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06082] Definition of EcucIntegerParamDef FrlfNumCtrlSupported

Parameter Name	FrlfNumCtrlSupported		
Parent Container	FrlfGeneral		
Description	Maximum number of FlexRay CCs that the FlexRay Interface supports		





Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 15		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06116] Definition of EcucStringParamDef FrlfPublicCddHeaderFile [

Parameter Name	FrlfPublicCddHeaderFile		
Parent Container	FrlfGeneral		
Description	Defines header files for callback functions which shall be included in case of CDDs. Range of characters is 1.. 32.		
Multiplicity	0..*		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_Frlf_06117] Definition of EcucBooleanParamDef FrlfReadCCConfigApi [

Parameter Name	FrlfReadCCConfigApi		
Parent Container	FrlfGeneral		
Description	Configuration parameter to enable/disable the optional Frlf_ReadCCConfig API.		
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	–	
Dependency			

[ECUC_FrIf_06109] Definition of EcucBooleanParamDef FrIfReconfigLPduSupport

Parameter Name	FrIfReconfigLPduSupport		
Parent Container	FrIfGeneral		
Description	Configuration parameter to enable/disable FrIf support to enable/disable the reconfiguration of a given LPdu according to the parameters (FrameId, Channel, Cycle Repetition, CycleOffset, PayloadLength, HeaderCRC) at runtime.		
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	–	
Dependency			

[ECUC_FrIf_06123] Definition of EcucStringParamDef FrIfTxConflictNotificationHeaderName

Parameter Name	FrIfTxConflictNotificationHeaderName		
Parent Container	FrIfGeneral		
Description	Configuration of the header file name that defines the UL_TxConflictNotification.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	FrIfTxConflictNotificationName		

[ECUC_FrIf_06122] Definition of EcucStringParamDef FrIfTxConflictNotificationName

Parameter Name	FrIfTxConflictNotificationName		
Parent Container	FrIfGeneral		
Description	Configuration of the API name that is called in case a TxConflict has been detected.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	–		





Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	FrIfTxConflictNotificationHeaderName		

[ECUC_FrIf_00001] Definition of EcucIntegerParamDef FrIfUnusedBitValue [

Parameter Name	FrIfUnusedBitValue		
Parent Container	FrIfGeneral		
Description	Set unused bits of transmitted Pdus to a defined value.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 1		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency			

[ECUC_FrIf_06083] Definition of EcucBooleanParamDef FrIfVersionInfoApi [

Parameter Name	FrIfVersionInfoApi		
Parent Container	FrIfGeneral		
Description	Enables/disables the existence of the FrIf_GetVersionInfo() API service true: FrIf_GetVersionInfo() API service exists false: FrIf_GetVersionInfo() API service does not exist		
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	–	
Dependency			

10.2.3 FrIfCluster

[ECUC_FrIf_05366] Definition of EcucParamConfContainerDef FrIfCluster [

Container Name	FrIfCluster		
Parent Container	FrIfConfig		
Description	This container specifies a FrIf Cluster and all related data which is required to enable communication of the Cluster. A Cluster may consist of more than one Controller.		
Multiplicity	1..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfCstIdx	1	[ECUC_FrIf_06002]
FrIfDetectNITError	1	[ECUC_FrIf_00003]
FrIfGChannels	1	[ECUC_FrIf_06006]
FrIfGColdStartAttempts	1	[ECUC_FrIf_06008]
FrIfGCycleCountMax	1	[ECUC_FrIf_06086]
FrIfGdActionPointOffset	1	[ECUC_FrIf_06020]
FrIfGdBit	1	[ECUC_FrIf_06021]
FrIfGdCasRxLowMax	1	[ECUC_FrIf_06024]
FrIfGdCycle	1	[ECUC_FrIf_06025]
FrIfGdDynamicSlotIdlePhase	1	[ECUC_FrIf_06026]
FrIfGdIgnoreAfterTx	1	[ECUC_FrIf_00012]
FrIfGdMacroTick	1	[ECUC_FrIf_06027]
FrIfGdMinislot	1	[ECUC_FrIf_06033]
FrIfGdMiniSlotActionPointOffset	1	[ECUC_FrIf_06032]
FrIfGdNit	1	[ECUC_FrIf_06034]
FrIfGdSampleClockPeriod	1	[ECUC_FrIf_06035]
FrIfGdStaticSlot	1	[ECUC_FrIf_06036]
FrIfGdSymbolWindow	1	[ECUC_FrIf_06037]
FrIfGdSymbolWindowActionPointOffset	1	[ECUC_FrIf_00011]
FrIfGdTSSTransmitter	1	[ECUC_FrIf_06038]
FrIfGdWakeupRxIdle	1	[ECUC_FrIf_06039]
FrIfGdWakeupRxLow	1	[ECUC_FrIf_06040]
FrIfGdWakeupRxWindow	1	[ECUC_FrIf_06041]
FrIfGdWakeupTxActive	1	[ECUC_FrIf_06043]
FrIfGdWakeupTxIdle	1	[ECUC_FrIf_06042]
FrIfGListenNoise	1	[ECUC_FrIf_06009]
FrIfGMacroPerCycle	1	[ECUC_FrIf_06010]
FrIfGMaxWithoutClockCorrectFatal	1	[ECUC_FrIf_06011]
FrIfGMaxWithoutClockCorrectPassive	1	[ECUC_FrIf_06012]





Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfGNetworkManagementVectorLength	1	[ECUC_FrIf_06013]
FrIfGNumberOfMinislots	1	[ECUC_FrIf_06014]
FrIfGNumberOfStaticSlots	1	[ECUC_FrIf_06015]
FrIfGPayloadLengthStatic	1	[ECUC_FrIf_06018]
FrIfGSyncFrameIDCountMax	1	[ECUC_FrIf_06019]
FrIfMainFunctionPeriod	1	[ECUC_FrIf_06003]
FrIfSafetyMargin	1	[ECUC_FrIf_00004]

Included Containers		
Container Name	Multiplicity	Dependency
FrIfClusterDemEventParameter Refs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
FrIfController	1..*	This container contains the configuration of FlexRay CC.
FrIfJobList	1	This container specifies a list of all FlexRay Jobs of the Cluster to be performed by FrIf_JobListExec_<FrIfCluster.ShortName>().

[ECUC_FrIf_06002] Definition of EcucIntegerParamDef FrIfClstIdx [

Parameter Name	FrIfClstIdx		
Parent Container	FrIfCluster		
Description	This parameter provides a zero-based consecutive index of the FlexRay Clusters. Upper layer BSW modules and the FrIf itself use this index to identify a FlexRay Cluster.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

[ECUC_FrIf_00003] Definition of EcucBooleanParamDef FrIfDetectNITError [

Parameter Name	FrIfDetectNITError		
Parent Container	FrIfCluster		
Description	Indicates whether NIT error status of each cluster shall be detected 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	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06006] Definition of EcucEnumerationParamDef FrlfGChannels [

Parameter Name	FrlfGChannels		
Parent Container	FrlfCluster		
Description	The channels that are used by the cluster. Implementation Type: Fr_ChannelType		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FR_CHANNEL_A	Cluster uses channel A	
	FR_CHANNEL_AB	Cluster uses channel A and B Implementation Type: Fr_ChannelType	
	FR_CHANNEL_B	Cluster uses channel B	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06008] Definition of EcucIntegerParamDef FrlfGColdStartAttempts [

Parameter Name	FrlfGColdStartAttempts		
Parent Container	FrlfCluster		
Description	Maximum number of times a node in the cluster is permitted to attempt to start the cluster by initiating schedule synchronization		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 31		
Default value	—		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06086] Definition of EcucIntegerParamDef FrlfGCycleCountMax [

Parameter Name	FrlfGCycleCountMax		
Parent Container	FrlfCluster		
Description	Maximum cycle counter value in a given cluster. Remark: Set to 63 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	7 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06020] Definition of EcucIntegerParamDef FrlfGdActionPointOffset [

Parameter Name	FrlfGdActionPointOffset		
Parent Container	FrlfCluster		
Description	Number of macroticks the action point is offset from the beginning of a static slot.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06021] Definition of EcucEnumerationParamDef FrlfGdBit [

Parameter Name	FrlfGdBit		
Parent Container	FrlfCluster		
Description	Nominal bit time in seconds		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	T100NS	–	
	T200NS	–	
	T400NS	–	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD





Dependency	
------------	--

]

[ECUC_Frlf_06024] Definition of EcucIntegerParamDef FrlfGdCasRxLowMax [

Parameter Name	FrlfGdCasRxLowMax		
Parent Container	FrlfCluster		
Description	Upper limit of the CAS acceptance windows [gdBit] Remark: Range 67 to 99 for FlexRay Protocol 2.1 Rev. A compliance		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	28 .. 254		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06025] Definition of EcucFloatParamDef FrlfGdCycle [

Parameter Name	FrlfGdCycle		
Parent Container	FrlfCluster		
Description	Length of the cycle, expressed in [s] Remark: Lower limit 0.000024 for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[2.4E-5 .. 0.016]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06026] Definition of EcucIntegerParamDef FrlfGdDynamicSlotIdle Phase [

Parameter Name	FrlfGdDynamicSlotIdlePhase		
Parent Container	FrlfCluster		
Description	Duration of the idle phase within a dynamic slot [Minislots].		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 2		





Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_00012] Definition of EcucIntegerParamDef FrlfGdIgnoreAfterTx [

Parameter Name	FrlfGdIgnoreAfterTx		
Parent Container	FrlfCluster		
Description	Duration for which the bitstrobing is paused after transmission [gdBit]. Remark: Set to 0 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 15		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06027] Definition of EcucFloatParamDef FrlfGdMacrotick [

Parameter Name	FrlfGdMacrotick		
Parent Container	FrlfCluster		
Description	Duration of the cluster wide nominal macrotick, expressed in s		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[1E-6 .. 6E-6]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06033] Definition of EcucIntegerParamDef FrlfGdMinislot [

Parameter Name	FrlfGdMinislot		
Parent Container	FrlfCluster		
Description	Duration of a minislot [Macroticks]		





Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06032] Definition of EcucIntegerParamDef FrlfGdMiniSlotActionPointOffset

Parameter Name	FrlfGdMiniSlotActionPointOffset		
Parent Container	FrlfCluster		
Description	Number of Macroticks the Minislot action point is offset from the beginning of a Minislot [Macroticks].		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 31		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06034] Definition of EcucIntegerParamDef FrlfGdNit

Parameter Name	FrlfGdNit		
Parent Container	FrlfCluster		
Description	Duration of the Network Idle Time [Macroticks] Remark: Upper limit 805 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 15978		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06035] Definition of EcucEnumerationParamDef FrlfGdSampleClockPeriod [

Parameter Name	FrlfGdSampleClockPeriod		
Parent Container	FrlfCluster		
Description	Sample clock period		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	T12_5NS	—	
	T25NS	—	
	T50NS	—	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06036] Definition of EcucIntegerParamDef FrlfGdStaticSlot [

Parameter Name	FrlfGdStaticSlot		
Parent Container	FrlfCluster		
Description	Duration of a static slot [Macroticks]. Remark: Range 4-661 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	3 .. 664		
Default value	—		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06037] Definition of EcucIntegerParamDef FrlfGdSymbolWindow [

Parameter Name	FrlfGdSymbolWindow		
Parent Container	FrlfCluster		
Description	Duration of the symbol window [Macroticks]. Remark: Range 0-142 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 162		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD





Dependency	
------------	--

[ECUC_Frlf_00011] Definition of EcucIntegerParamDef FrlfGdSymbolWindowActionPointOffset

Parameter Name	FrlfGdSymbolWindowActionPointOffset		
Parent Container	FrlfCluster		
Description	Number of macroticks the action point offset is from the beginning of the symbol window [Macroticks]. Remark: Set to GdActionPointOffset for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06038] Definition of EcucIntegerParamDef FrlfGdTSSTransmitter

Parameter Name	FrlfGdTSSTransmitter		
Parent Container	FrlfCluster		
Description	Number of bits in the Transmission Start Sequence [gdBits]. Remark: Lower limit 3 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 15		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06039] Definition of EcucIntegerParamDef FrlfGdWakeupRxIdle

Parameter Name	FrlfGdWakeupRxIdle		
Parent Container	FrlfCluster		
Description	Number of bits used by the node to test the duration of the 'idle' or HIGH phase of a received wakeup [gdBit]. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gdWakeupSymbolRxIdle. Lower limit 14 for FlexRay Protocol 2.1 Rev. A compliance.		





Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 59		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06040] Definition of EcucIntegerParamDef FrlfGdWakeupRxLow

Parameter Name	FrlfGdWakeupRxLow		
Parent Container	FrlfCluster		
Description	Number of bits used by the node to test the duration of the LOW phase of a received wakeup [gdBit]. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gdWakeupSymbolRxLow. Lower limit 11 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 59		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06041] Definition of EcucIntegerParamDef FrlfGdWakeupRxWindow

Parameter Name	FrlfGdWakeupRxWindow		
Parent Container	FrlfCluster		
Description	The size of the window used to detect wakeups [gdBit]. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gdWakeupSymbolRxWindow. Upper limit 301 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	76 .. 485		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06043] Definition of EcucIntegerParamDef FrlfGdWakeupTxActive [

Parameter Name	FrlfGdWakeupTxActive		
Parent Container	FrlfCluster		
Description	Number of bits used by the node to transmit the LOW phase of awakeup symbol and the HIGH and LOW phases of a WUDOP [gdBit]. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gdWakeupSymbolTxLow.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	15 .. 60		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06042] Definition of EcucIntegerParamDef FrlfGdWakeupTxIdle [

Parameter Name	FrlfGdWakeupTxIdle		
Parent Container	FrlfCluster		
Description	Number of bits used by the node to transmit the 'idle' part of a wakeup symbol [gdBit]. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gdWakeupSymbolTxIdle.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	45 .. 180		
Default value	—		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06009] Definition of EcucIntegerParamDef FrlfGListenNoise [

Parameter Name	FrlfGListenNoise		
Parent Container	FrlfCluster		
Description	Upper limit for the start up listen timeout and wake up listen timeout in the presence of noise. It is used as a multiplier of the node parameter pdListenTimeout.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 16		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME





	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06010] Definition of EcucIntegerParamDef FrlfGMacroPerCycle [

Parameter Name	FrlfGMacroPerCycle		
Parent Container	FrlfCluster		
Description	Number of macroticks in a communication cycle. Note: Lower limit 10 for FlexRay Protocol 2.1 Rev. A compliance		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	8 .. 16000		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06011] Definition of EcucIntegerParamDef FrlfGMaxWithoutClockCorrectFatal [

Parameter Name	FrlfGMaxWithoutClockCorrectFatal		
Parent Container	FrlfCluster		
Description	Threshold used for testing the vClockCorrectionFailed counter. Defines the number of consecutive even/odd Cycle pairs with missing clock correction terms that will cause the protocol to transition from the POC:normal active or POC:normal passive state into the POC:halt state. [Even/odd cycle pairs].		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 15		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06012] Definition of EcucIntegerParamDef FrlfGMaxWithoutClockCorrectPassive [

Parameter Name	FrlfGMaxWithoutClockCorrectPassive		
Parent Container	FrlfCluster		
Description	Threshold used for testing the vClockCorrectionFailed counter. Defines the number of consecutive even/odd Cycle pairs with missing clock correction terms that will cause the protocol to transition from the POC:normal active state to the POC:normal passive state. [Even/Odd cycle pairs]		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 15		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06013] Definition of EcucIntegerParamDef FrlfGNetworkManagementVectorLength [

Parameter Name	FrlfGNetworkManagementVectorLength		
Parent Container	FrlfCluster		
Description	Length of the Network Management vector in a cluster [bytes]		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 12		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06014] Definition of EcucIntegerParamDef FrlfGNumberOfMinislots [

Parameter Name	FrlfGNumberOfMinislots		
Parent Container	FrlfCluster		
Description	Number of minislots in the dynamic segment Remark: Upper limit 7986 for FlexRay Protocol 2.1 Rev. A compliance		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 7988		
Default value	–		
Post-Build Variant Value	true		



△

Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

┌

[ECUC_Frlf_06015] Definition of EcucIntegerParamDef FrlfGNumberOfStatic Slots

Parameter Name	FrlfGNumberOfStaticSlots		
Parent Container	FrlfCluster		
Description	Number of static slots in the static segment		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 1023		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

┌

[ECUC_Frlf_06018] Definition of EcucIntegerParamDef FrlfGPayloadLengthStatic

Parameter Name	FrlfGPayloadLengthStatic		
Parent Container	FrlfCluster		
Description	Payload length of a static frame [16 bit words]		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 127		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

┌

[ECUC_Frlf_06019] Definition of EcucIntegerParamDef FrlfGSyncFrameIDCountMax

Parameter Name	FrlfGSyncFrameIDCountMax		
Parent Container	FrlfCluster		
Description	Maximum number of distinct syncframe identifiers present in a given cluster. This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gSyncNodeMax.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	2 .. 15		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06003] Definition of EcucFloatParamDef FrlfMainFunctionPeriod

Parameter Name	FrlfMainFunctionPeriod		
Parent Container	FrlfCluster		
Description	The execution cycle of the Frlf_MainFunction_<FrlfCluster.ShortName>() in seconds. The Frlf does not require this information but the BSW scheduler, which invokes the cluster main functions, needs it in order to plan its tasks.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Dependency			

[ECUC_Frlf_00004] Definition of EcucIntegerParamDef FrlfSafetyMargin

Parameter Name	FrlfSafetyMargin		
Parent Container	FrlfCluster		
Description	Additional timespan in macroticks which takes jitter into account to be able to set the JobListPointer to the next possible job which can be executed in case the FlexRay Job List Execution Function has be resynchronized.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 1024000		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE





	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.4 FrIfController

[ECUC_FrIf_05363] Definition of EcucParamConfContainerDef FrIfController [

Container Name	FrIfController		
Parent Container	FrIfCluster		
Description	This container contains the configuration of FlexRay CC.		
Multiplicity	1..*		
Post-Build Variant Multiplicity	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE, VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Link time	–	
	Post-build time	–	
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfCtrlIdx	1	[ECUC_FrIf_06045]
FrIfFrCtrlRef	1	[ECUC_FrIf_06044]

Included Containers		
Container Name	Multiplicity	Dependency
FrIfFrameTriggering	1..*	A Frame triggering contains the communication parameters of the FlexRay Frame as well as a reference to the Frame Construction Plan.
FrIfLPdu	1..*	Reference to a L-PDU index
FrIfTransceiver	1..2	Up to two FlexRay Transceivers may connect a Controller to a Cluster. This container realizes a Controller-Transceiver assignment.

]

[ECUC_FrIf_06045] Definition of EcucIntegerParamDef FrIfCtrlIdx [

Parameter Name	FrIfCtrlIdx
Parent Container	FrIfController
Description	This parameter provides a zero-based consecutive index of the FlexRay Communication Controllers. Upper layer BSW modules and the FrIf itself use this index to identify a FlexRay CC.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)



△

Range	0 .. 31		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

]

[ECUC_FrIf_06044] Definition of EcucReferenceDef FrIfFrCtrlRef [

Parameter Name	FrIfFrCtrlRef		
Parent Container	FrIfController		
Description	Reference to a Controller, which is handled by a specific Driver. This reference is unique for the ECU.		
Multiplicity	1		
Type	Symbolic name reference to FrController		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

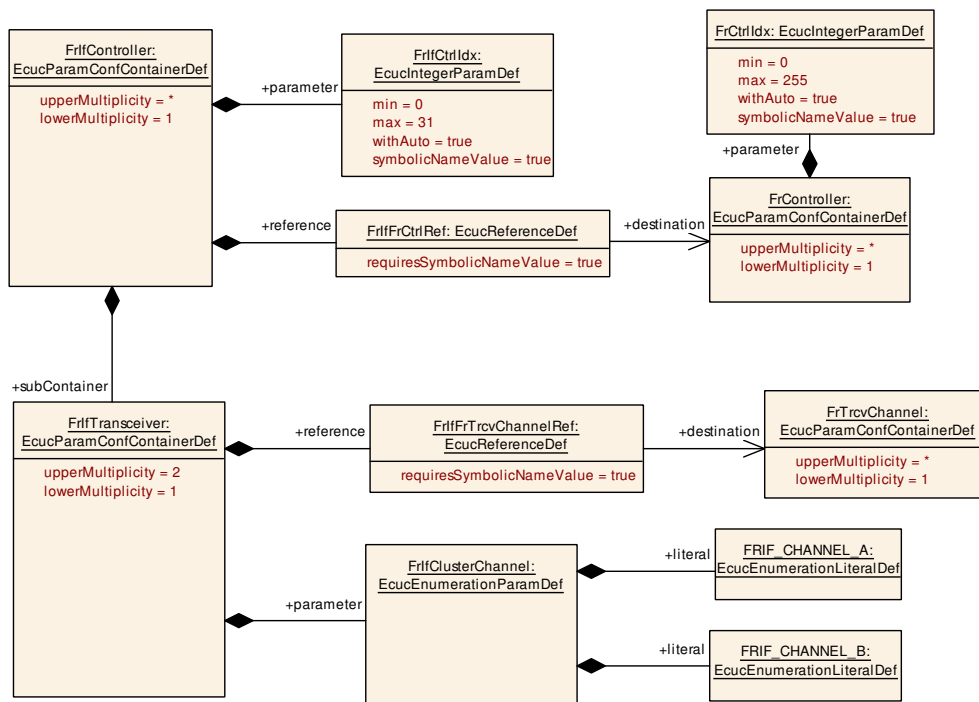


Figure 10.2: FlexRay Interface Controller (hardware reference)

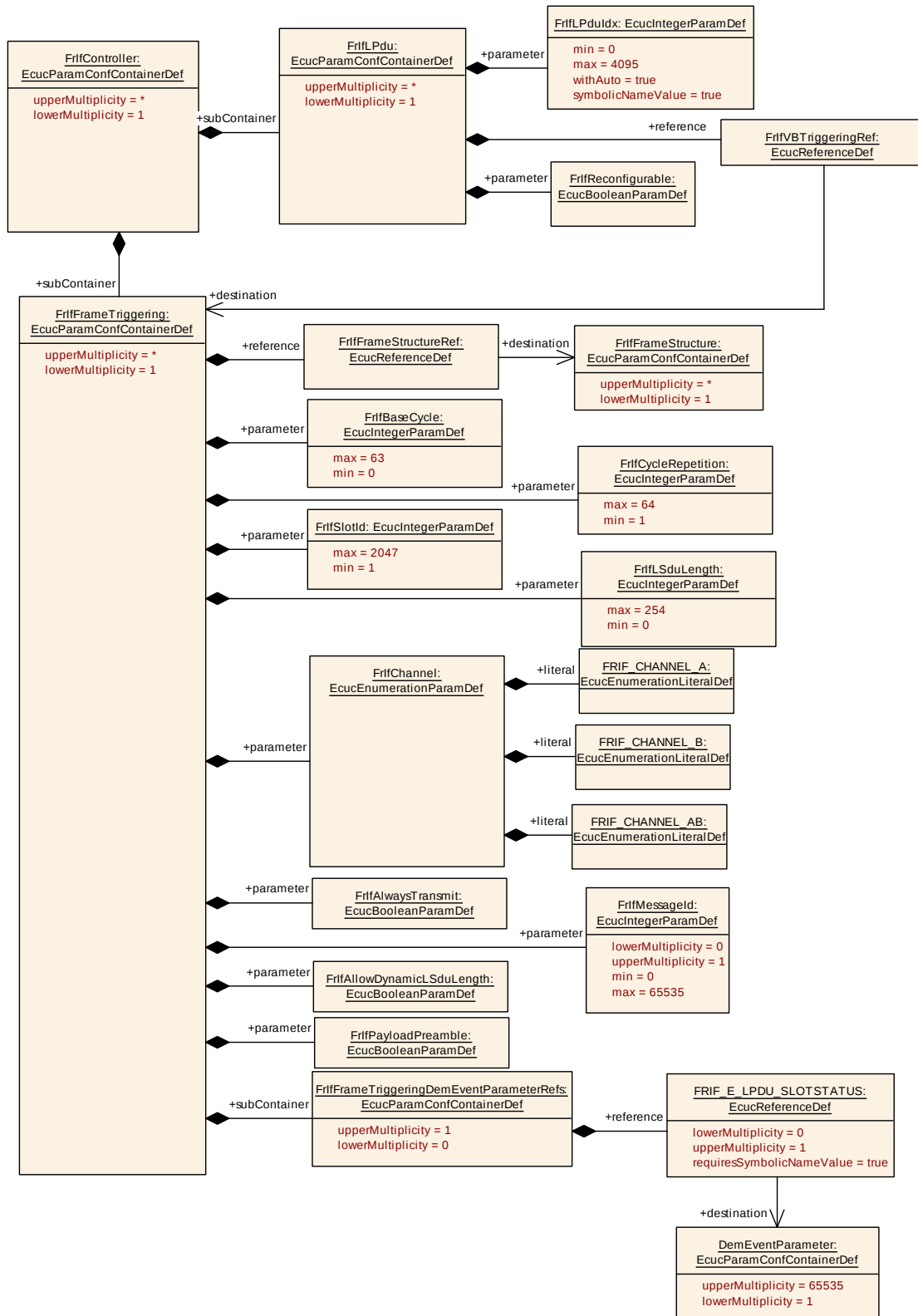


Figure 10.3: FlexRay Interface Controller (data reference)

10.2.5 FrIfTransceiver

[ECUC_FrIf_05391] Definition of EcucParamConfContainerDef FrIfTransceiver [

Container Name	FrIfTransceiver
Parent Container	FrIfController
Description	Up to two FlexRay Transceivers may connect a Controller to a Cluster. This container realizes a Controller-Transceiver assignment.
Multiplicity	1..2
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfClusterChannel	1	[ECUC_FrIf_06062]
FrIfFrTrcvChannelRef	1	[ECUC_FrIf_06061]

No Included Containers

]

[ECUC_FrIf_06062] Definition of EcucEnumerationParamDef FrIfClusterChannel [

Parameter Name	FrIfClusterChannel		
Parent Container	FrIfTransceiver		
Description	This parameter identifies to which one of the two Channels (A, B, A and B) of the Cluster the Transceiver is connected. FrIfClusterChannel shall map to Fr_ChannelType: FRIF_CHANNEL_A == FR_CHANNEL_A FRIF_CHANNEL_B == FR_CHANNEL_B FR_CHANNEL_AB shall not be used.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRIF_CHANNEL_A	Channel A	
	FRIF_CHANNEL_B	Channel B	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_FrIf_06061] Definition of EcucReferenceDef FrIfFrTrcvChannelRef [

Parameter Name	FrIfFrTrcvChannelRef		
Parent Container	FrIfTransceiver		
Description	Reference to a Transceiver Driver Channel. This reference is unique for the ECU.		
Multiplicity	1		
Type	Symbolic name reference to FrTrcvChannel		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE





	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.6 FrIfLPdu

[ECUC_FrIf_05364] Definition of EcucParamConfContainerDef FrIfLPdu [

Container Name	FrIfLPdu		
Parent Container	FrIfController		
Description	Reference to a L-PDU index		
Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfLPduldx	1	[ECUC_FrIf_06058]
FrIfReconfigurable	1	[ECUC_FrIf_00008]
FrIfVBTtriggeringRef	1	[ECUC_FrIf_06057]

No Included Containers

]

[ECUC_FrIf_06058] Definition of EcucIntegerParamDef FrIfLPduldx [

Parameter Name	FrIfLPduldx		
Parent Container	FrIfLPdu		
Description	This parameter identifies the L-PDU in the interaction between FlexRay Interface and FlexRay Driver.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 4095		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	All Variants
	Link time	–	
	Post-build time	–	
Dependency	withAuto = true		

]

[ECUC_Frlf_00008] Definition of EcucBooleanParamDef FrlfReconfigurable [

Parameter Name	FrlfReconfigurable		
Parent Container	FrlfLPdu		
Description	This parameter specifies that this LPdu is reconfigurable using Frlf_ReconfigLPdu. This means that this LPdu can be assigned to a different FrameTriggering at runtime. However, this reconfiguration is limited by hardware constraints. The direction of the LPdu cannot be reconfigured.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06057] Definition of EcucReferenceDef FrlfVBTriggeringRef [

Parameter Name	FrlfVBTriggeringRef		
Parent Container	FrlfLPdu		
Description	Reference to the assigned Frame triggering.		
Multiplicity	1		
Type	Reference to FrlfFrameTriggering		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.7 FrlfFrameTriggering

[ECUC_Frlf_06090] Definition of EcucParamConfContainerDef FrlfFrameTriggering [

Container Name	FrlfFrameTriggering		
Parent Container	FrlfController		
Description	A Frame triggering contains the communication parameters of the FlexRay Frame as well as a reference to the Frame Construction Plan.		
Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD





Configuration Parameters

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfAllowDynamicLSduLength	1	[ECUC_FrIf_06049]
FrIfAlwaysTransmit	1	[ECUC_FrIf_00013]
FrIfBaseCycle	1	[ECUC_FrIf_06051]
FrIfChannel	1	[ECUC_FrIf_06052]
FrIfCycleRepetition	1	[ECUC_FrIf_06053]
FrIfLSduLength	1	[ECUC_FrIf_06054]
FrIfMessageId	0..1	[ECUC_FrIf_00010]
FrIfPayloadPreamble	1	[ECUC_FrIf_06055]
FrIfSlotId	1	[ECUC_FrIf_06056]
FrIfFrameStructureRef	1	[ECUC_FrIf_06048]

Included Containers		
Container Name	Multiplicity	Dependency
FrIfFrameTriggeringDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.

[ECUC_FrIf_06049] Definition of EcucBooleanParamDef FrIfAllowDynamicLSduLength

Parameter Name	FrIfAllowDynamicLSduLength		
Parent Container	FrIfFrameTriggering		
Description	Allows L-PDU length reduction ('FrIfLSduLength' defines max. length) and indicates that the related CC buffer has to be reconfigured for the actual length and Header-CRC before transmission of the L-PDU.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_00013] Definition of EcucBooleanParamDef FrlfAlwaysTransmit [

Parameter Name	FrlfAlwaysTransmit		
Parent Container	FrlfFrameTriggering		
Description	Defines whether the driver's API function Fr_TransmitTxLPdu() shall always be called for this L-PDU.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06051] Definition of EcucIntegerParamDef FrlfBaseCycle [

Parameter Name	FrlfBaseCycle		
Parent Container	FrlfFrameTriggering		
Description	This parameter contains the FlexRay Base Cycle used to transmit this FlexRay Frame.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06052] Definition of EcucEnumerationParamDef FrlfChannel [

Parameter Name	FrlfChannel		
Parent Container	FrlfFrameTriggering		
Description	This parameter contains the FlexRay Channel used to transmit this FlexRay Frame.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRIF_CHANNEL_A	Channel A	
	FRIF_CHANNEL_AB	Channel A and B	
	FRIF_CHANNEL_B	Channel B	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

[ECUC_Frlf_06053] Definition of EcucIntegerParamDef FrlfCycleRepetition [

Parameter Name	FrlfCycleRepetition		
Parent Container	FrlfFrameTriggering		
Description	This parameter contains the FlexRay Cycle Repetition used to transmit this FlexRay Frame. Possible values for FlexRay Protocol version 2.1: 1,2,4,8,16,32,64 Possible values for FlexRay Protocol version 3.0: 1,2,4,5,8,10,16,20,32,40,50,64		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 64		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06054] Definition of EcucIntegerParamDef FrlfLSduLength [

Parameter Name	FrlfLSduLength		
Parent Container	FrlfFrameTriggering		
Description	The payload length of the Frame is given here. This parameter is required for validation if configured PDUs and update information fits into the Frame at configuration time [bytes].		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 254		
Default value	—		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	The parameter depends on the low level parameters of the FlexRay CC.		

[ECUC_Frlf_00010] Definition of EcucIntegerParamDef FrlfMessageld [

Parameter Name	FrlfMessageld	
Parent Container	FrlfFrameTriggering	
Description	The first two bytes of the payload segment of the FlexRay frame format for frames transmitted in the dynamic segment can be used as receiver filterable data called the message ID.	
Multiplicity	0..1	
Type	EcucIntegerParamDef	
Range	0 .. 65535	
Default value	—	
Post-Build Variant Multiplicity	true	





Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06055] Definition of EcucBooleanParamDef FrlfPayloadPreamble [

Parameter Name	FrlfPayloadPreamble		
Parent Container	FrlfFrameTriggering		
Description	Switching the Payload Preamble bit.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06056] Definition of EcucIntegerParamDef FrlfSlotId [

Parameter Name	FrlfSlotId		
Parent Container	FrlfFrameTriggering		
Description	This parameter contains the FlexRay Slot ID used to transmit this FlexRay Frame.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	1 .. 2047		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06048] Definition of EcucReferenceDef FrlfFrameStructureRef [

Parameter Name	FrlfFrameStructureRef		
Parent Container	FrlfFrameTriggering		
Description	Reference to the Construction Plan of the FlexRay Frame.		





Multiplicity	1		
Type	Reference to FrIfFrameStructure		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.8 FrIfJobList

[ECUC_Frlf_05367] Definition of EcucParamConfContainerDef FrIfJobList [

Container Name	FrIfJobList		
Parent Container	FrIfCluster		
Description	This container specifies a list of all FlexRay Jobs of the Cluster to be performed by FrIf_JobListExec_<FrIfCluster.ShortName>().		
Multiplicity	1		
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfAbsTimerRef	1	[ECUC_Frlf_06063]

Included Containers		
Container Name	Multiplicity	Dependency
FrIfJob	1..*	A job may contain more than one operation that are executed at a specific point in time.

]

[ECUC_Frlf_06063] Definition of EcucReferenceDef FrIfAbsTimerRef [

Parameter Name	FrIfAbsTimerRef		
Parent Container	FrIfJobList		
Description	Reference to the absolute timer to be used to trigger the interrupt whose ISR contains the FrIf_JobListExec_<FrIfCluster.ShortName>() function.		
Multiplicity	1		
Type	Symbolic name reference to FrAbsoluteTimer		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

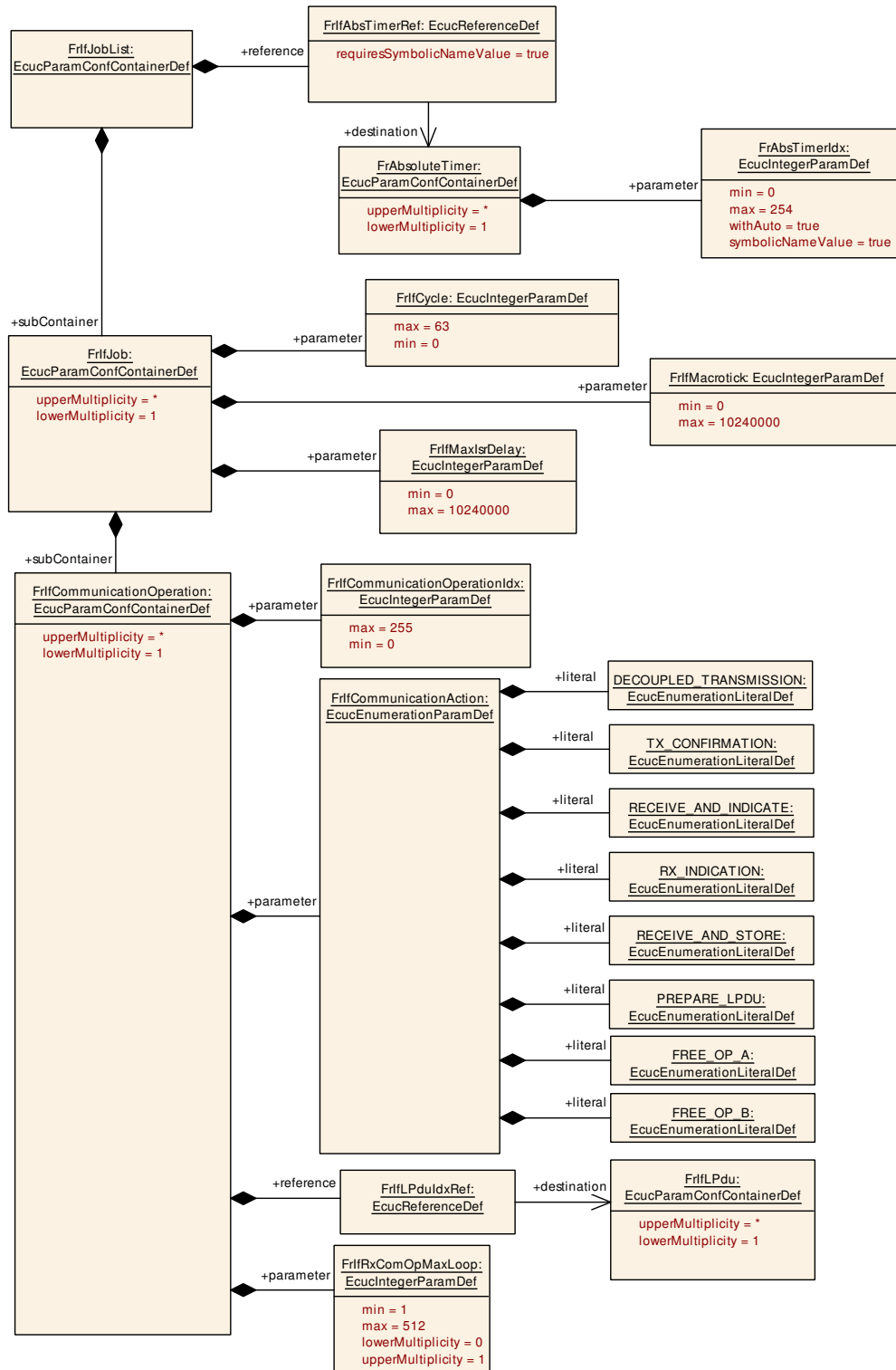


Figure 10.4: FlexRay Interface Joblist

10.2.9 FrIfJob

[ECUC_Frlf_05368] Definition of EcucParamConfContainerDef FrIfJob [

Container Name	FrIfJob		
Parent Container	FrIfJobList		
Description	A job may contain more than one operation that are executed at a specific point in time.		
Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfCycle	1	[ECUC_FrIf_06064]
FrIfMacrotick	1	[ECUC_FrIf_06065]
FrIfMaxIsrDelay	1	[ECUC_FrIf_06004]

Included Containers		
Container Name	Multiplicity	Dependency
FrIfCommunicationOperation	1..*	A separate operation which is part of a FlexRay Job and defines what type of action is executed.

[ECUC_FrIf_06064] Definition of EcucIntegerParamDef FrIfCycle [

Parameter Name	FrIfCycle		
Parent Container	FrIfJob		
Description	The FlexRay Cycle in which the communication operation will execute this job		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 63		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_FrIf_06065] Definition of EcucIntegerParamDef FrIfMacrotick [

Parameter Name	FrIfMacrotick		
Parent Container	FrIfJob		
Description	Macrotick offset in the Cycle [Macrotick]		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 10240000		
Default value	–		





Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06004] Definition of EcucIntegerParamDef FrlfMaxIsrDelay [

Parameter Name	FrlfMaxIsrDelay		
Parent Container	FrlfJob		
Description	The maximum delay in macroticks the Frlf_JobListExec_<FrlfCluster.ShortName>() function is processed after the absolute timer interrupt was triggered.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 10240000		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

10.2.10 FrlfCommunicationOperation

[ECUC_Frlf_05369] Definition of EcucParamConfContainerDef FrlfCommunicationOperation [

Container Name	FrlfCommunicationOperation		
Parent Container	FrlfJob		
Description	A separate operation which is part of a FlexRay Job and defines what type of action is executed.		
Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrlfCommunicationAction	1	[ECUC_Frlf_06067]
FrlfCommunicationOperationIdx	1	[ECUC_Frlf_06068]
FrlfRxComOpMaxLoop	0..1	[ECUC_Frlf_00007]
FrlfLPdulIdxRef	1	[ECUC_Frlf_06066]

No Included Containers

[ECUC_FrIf_06067] Definition of EcucEnumerationParamDef FrIfCommunicationAction

Parameter Name	FrIfCommunicationAction		
Parent Container	FrIfCommunicationOperation		
Description	The action to be performed in the FlexRay Operation		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	DECOUPLED_TRANSMISSION	Decoupled transmission	
	FREE_OP_A	User defined communication operation.	
	FREE_OP_B	User defined communication operation.	
	PREPARE_LPDU	Prepare message buffer of CC	
	RECEIVE_AND_INDICATE	Immediate reception	
	RECEIVE_AND_STORE	Decoupled reception	
	RX_INDICATION	Reception indication	
	TX_CONFIRMATION	Transmission confirmation with optional Tx Conflict check	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	FrIfCommunicationAction can be configured as PREPARE_LPDU only if FrPrepare LPduSupport (ECUC_Fr_00453) is configured as TRUE.		

[ECUC_FrIf_06068] Definition of EcucIntegerParamDef FrIfCommunicationOperationIdx

Parameter Name	FrIfCommunicationOperationIdx		
Parent Container	FrIfCommunicationOperation		
Description	For each FlexRay Communication Job, this index spans a range of zero-based consecutive values and thus defines the order of the FlexRay Communication Operation in the respective FlexRay Communication Job.		
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	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_00007] Definition of EcucIntegerParamDef FrlfRxComOpMaxLoop [

Parameter Name	FrlfRxComOpMaxLoop		
Parent Container	FrlfCommunicationOperation		
Description	Defines the maximum number of loops for the receive RECEIVE_AND_INDICATE (Use case: emptying a FIFO). Please note that the parameter is mandatory if FrlfCommunicationAction parameter is set to RECEIVE_AND_INDICATE. For all other operations this parameter can be ignored.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	1 .. 512		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06066] Definition of EcucReferenceDef FrlfLPduldxRef [

Parameter Name	FrlfLPduldxRef		
Parent Container	FrlfCommunicationOperation		
Description	Reference to a L-PDu index		
Multiplicity	1		
Type	Reference to FrlfLPdu		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

10.2.11 FrlfFrameStructure

[ECUC_Frlf_05370] Definition of EcucParamConfContainerDef FrlfFrameStructure [

Container Name	FrlfFrameStructure
Parent Container	FrlfConfig
Description	The Frame structure specifies a Construction Plan how a Frame is assembled with PDUs and their respective Update-Bits.





Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrlfByteOrder	1	[ECUC_Frlf_06113]

Included Containers		
Container Name	Multiplicity	Dependency
FrlfPduInFrame	1..*	This container holds all the information about a PDU in a Flex Ray Frame.

]

[[ECUC_Frlf_06113](#)] Definition of EcucEnumerationParamDef FrlfByteOrder [

Parameter Name	FrlfByteOrder		
Parent Container	FrlfFrameStructure		
Description	This parameter defines the ByteOrder of all Pdus that are mapped into the Frame. The absolute position of a Pdu in the Frame is determined by the definition of the Byte Order parameter: If BIG_ENDIAN is specified, the FrlfPduOffset indicates the position of the most significant bit in the Frame. If LITTLE_ENDIAN is specified, the FrlfPdu Offset indicates the position of the least significant bit in the Frame.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	BIG_ENDIAN	–	
	LITTLE_ENDIAN	–	
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.12 FrlfPduInFrame

[[ECUC_Frlf_05371](#)] Definition of EcucParamConfContainerDef FrlfPduInFrame [

Container Name	FrlfPduInFrame
Parent Container	FrlfFrameStructure
Description	This container holds all the information about a PDU in a FlexRay Frame.





Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrlfPduOffset	1	[ECUC_Frlf_06070]
FrlfPduUpdateBitOffset	0..1	[ECUC_Frlf_06071]
FrlfPduRef	1	[ECUC_Frlf_06069]

No Included Containers

[ECUC_Frlf_06070] Definition of EcucIntegerParamDef FrlfPduOffset [

Parameter Name	FrlfPduOffset		
Parent Container	FrlfPduInFrame		
Description	The value specifies the offset of the PDU within the Frame [bytes].		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 253		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter depends on the number of PDUs contained in the Frame, PDU length, and Update-Bits of other PDUs in the Frame. In addition, if the Frame will be sent in static segment, this parameter depends on GPayloadLengthStatic.		

[ECUC_Frlf_06071] Definition of EcucIntegerParamDef FrlfPduUpdateBitOffset [

Parameter Name	FrlfPduUpdateBitOffset		
Parent Container	FrlfPduInFrame		
Description	This value specifies where the PDU's Update-Bit is stored in the Frame (bit location of PDU's Update-Bit in the FlexRay Frame).		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 2031		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE





Value Configuration Class	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	This parameter depends on the number of PDUs contained in the Frame, PDU length, and Update-Bits of other PDUs in the Frame. In addition, if the Frame will be sent in static segment, this parameter depends on GPayloadLengthStatic.		

[ECUC_Frlf_06069] Definition of EcucReferenceDef FrlfPduRef [

Parameter Name	FrlfPduRef		
Parent Container	FrlfPduInFrame		
Description	This is the reference to the local definition of a PDU.		
Multiplicity	1		
Type	Reference to FrlfPdu		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

10.2.13 FrlfPdu

[ECUC_Frlf_05372] Definition of EcucParamConfContainerDef FrlfPdu [

Container Name	FrlfPdu		
Parent Container	FrlfConfig		
Description	Contains PDU information. A PDU may be either a transmission PDU or a reception PDU.		
Multiplicity	1..*		
Post-Build Variant Multiplicity	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Configuration Parameters			

No Included Parameters

Included Containers

Container Name	Multiplicity	Dependency
FrlfPduDirection	1	A PDU is either transmit or receive

10.2.14 FrlfTxPdu

[ECUC_Frlf_05374] Definition of EcucParamConfContainerDef FrlfTxPdu [

Container Name	FrlfTxPdu
Parent Container	FrlfPduDirection
Description	This container specifies transmission PDUs.
Multiplicity	0..1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrlfConfirm	1	[ECUC_Frlf_06075]
FrlfCounterLimit	0..1	[ECUC_Frlf_06076]
FrlfImmediate	1	[ECUC_Frlf_06077]
FrlfNoneMode	0..1	[ECUC_Frlf_06050]
FrlfTxPduId	1	[ECUC_Frlf_06078]
FrlfTxPduRef	1	[ECUC_Frlf_06074]

No Included Containers

]

[ECUC_Frlf_06075] Definition of EcucBooleanParamDef FrlfConfirm [

Parameter Name	FrlfConfirm		
Parent Container	FrlfTxPdu		
Description	Defines whether the transmission of a PDU should be checked and confirmed to the PDU owning BSW module. If "FrlfUserTxUL" is configured as FR_TSYN then this parameter has to be set to FALSE for this PDU.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	FrlfUserTxUL		

]

[ECUC_Frlf_06076] Definition of EcucIntegerParamDef FrlfCounterLimit [

Parameter Name	FrlfCounterLimit
Parent Container	FrlfTxPdu
Description	This value states the maximum number of indication of ready PDU data to the Frlf (i.e. maximum number of invocations of Frlf_Transmit) without an intermediate transmission of the PDU.
Multiplicity	0..1
Type	EcucIntegerParamDef





Range	1 .. 255	
Default value	–	
Post-Build Variant Value	true	
Value Configuration Class	Pre-compile time	X VARIANT-PRE-COMPILE
	Link time	X VARIANT-LINK-TIME
	Post-build time	X VARIANT-POST-BUILD
Dependency		

[ECUC_FrIf_06077] Definition of EcucBooleanParamDef FrIfImmediate [

Parameter Name	FrIfImmediate		
Parent Container	FrIfTxPdu		
Description	Defines whether the PDU is transmitted immediate or decoupled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_FrIf_06050] Definition of EcucBooleanParamDef FrIfNoneMode [

Parameter Name	FrIfNoneMode		
Parent Container	FrIfTxPdu		
Description	Using the "None-Mode" which means that there is no API FrIf_Transmit call of the upper layer for this PDU.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency	FrIfImmediate		

[ECUC_Frlf_06078] Definition of EcucIntegerParamDef FrlfTxPduld [

Parameter Name	FrlfTxPduld		
Parent Container	FrlfTxPdu		
Description	The global PDU identifier, which has to be used by the upper layer BSW module. The identifier has to be zero based and consecutive.		
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	–	
Dependency	withAuto = true		

]

[ECUC_Frlf_06074] Definition of EcucReferenceDef FrlfTxPduRef [

Parameter Name	FrlfTxPduRef		
Parent Container	FrlfTxPdu		
Description	Reference to the external PDU definition.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

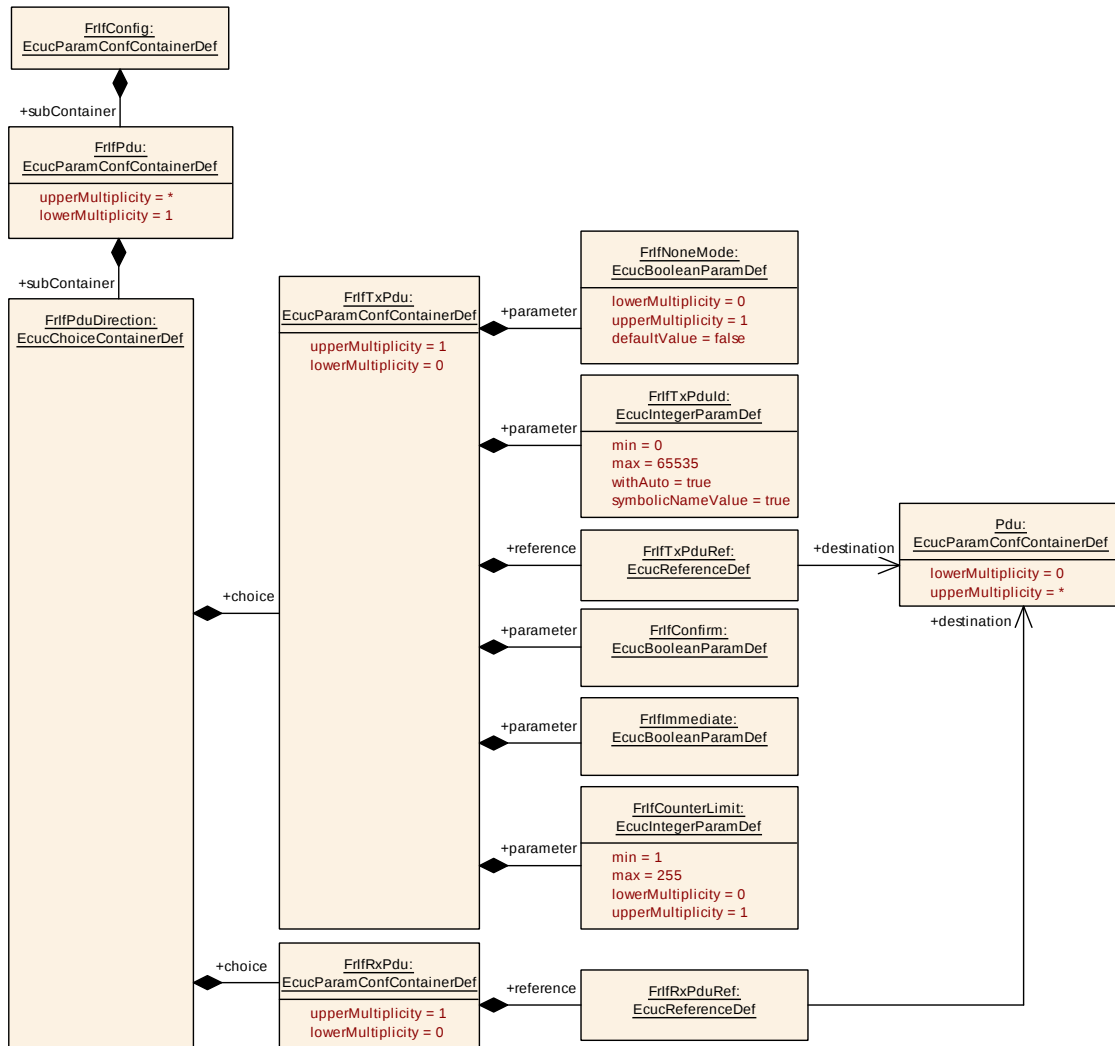


Figure 10.5: FlexRay Interface Pdu

10.2.15 FrIfRxPdu

[ECUC_Frlf_05373] Definition of EcucParamConfContainerDef FrIfRxPdu [

Container Name	FrIfRxPdu
Parent Container	FrIfPduDirection
Description	Receive PDU
Multiplicity	0..1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrIfRxPduRef	1	[ECUC_Frlf_06073]

No Included Containers

[ECUC_Frlf_06073] Definition of EcucReferenceDef FrlfRxPduRef [

Parameter Name	FrlfRxPduRef		
Parent Container	FrlfRxPdu		
Description	Reference to the external PDU definition.		
Multiplicity	1		
Type	Reference to Pdu		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

10.2.16 FrlfPduDirection

[ECUC_Frlf_06072] Definition of EcucChoiceContainerDef FrlfPduDirection [

Choice Container Name	FrlfPduDirection
Parent Container	FrlfPdu
Description	A PDU is either transmit or receive
Multiplicity	1

No Included Parameters

Container Choices		
Container Name	Multiplicity	Dependency
FrlfRxPdu	0..1	Receive PDU
FrlfTxPdu	0..1	This container specifies transmission PDUs.

10.2.17 FrlfConfig

[ECUC_Frlf_06001] Definition of EcucParamConfContainerDef FrlfConfig [

Container Name	FrlfConfig
Parent Container	Frlf
Description	This container contains the configuration parameters and sub containers of the AUTOSAR Frlf module.
Multiplicity	1
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrlfMaxPduCnt	0..1	[ECUC_Frlf_06121]

Included Containers		
Container Name	Multiplicity	Dependency
FrlfCluster	1..*	This container specifies a Frlf Cluster and all related data which is required to enable communication of the Cluster. A Cluster may consist of more than one Controller.
FrlfFrameStructure	1..*	The Frame structure specifies a Construction Plan how a Frame is assembled with PDUs and their respective Update-Bits.
FrlfPdu	1..*	Contains PDU information. A PDU may be either a transmission PDU or a reception PDU.

[[ECUC_Frlf_06121](#)] Definition of EcucIntegerParamDef [FrlfMaxPduCnt](#) [

Parameter Name	FrlfMaxPduCnt		
Parent Container	FrlfConfig		
Description	Maximum number of Pdus. This parameter is needed only in case of post-build loadable implementation using static memory allocation.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 18446744073709551615		
Default value	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Dependency			

10.2.18 FrlfClusterDemEventParameterRefs

[[ECUC_Frlf_06091](#)] Definition of EcucParamConfContainerDef [FrlfClusterDemEventParameterRefs](#) [

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

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FRIF_E_ACS_CH_A	0..1	[ECUC_Frlf_06097]
FRIF_E_ACS_CH_B	0..1	[ECUC_Frlf_06098]
FRIF_E_NIT_CH_A	0..1	[ECUC_Frlf_06093]
FRIF_E_NIT_CH_B	0..1	[ECUC_Frlf_06094]
FRIF_E_SW_CH_A	0..1	[ECUC_Frlf_06095]
FRIF_E_SW_CH_B	0..1	[ECUC_Frlf_06096]

No Included Containers

[ECUC_Frlf_06097] Definition of EcucReferenceDef FRIF_E_ACS_CH_A [

Parameter Name	FRIF_E_ACS_CH_A		
Parent Container	FrIfClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when an error in ACS on channel A was detected. If the reference is not configured the error shall not be reported.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06098] Definition of EcucReferenceDef FRIF_E_ACS_CH_B [

Parameter Name	FRIF_E_ACS_CH_B		
Parent Container	FrlfClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when an error in ACS on channel B was detected. If the reference is not configured the error shall not be reported.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06093] Definition of EcucReferenceDef FRIF_E_NIT_CH_A [

Parameter Name	FRIF_E_NIT_CH_A		
Parent Container	FrlfClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when an error in NIT on channel A was detected. If the reference is not configured the error shall not be reported.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06094] Definition of EcucReferenceDef FRIF_E_NIT_CH_B [

Parameter Name	FRIF_E_NIT_CH_B		
Parent Container	FrlfClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when an error in NIT on channel B was detected. If the reference is not configured the error shall not be reported.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		





Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06095] Definition of EcucReferenceDef FRIF_E_SW_CH_A [

Parameter Name	FRIF_E_SW_CH_A		
Parent Container	FrlfClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when an error in SW on channel A was detected. If the reference is not configured the error shall not be reported.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

[ECUC_Frlf_06096] Definition of EcucReferenceDef FRIF_E_SW_CH_B [

Parameter Name	FRIF_E_SW_CH_B		
Parent Container	FrlfClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when an error in SW on channel B was detected. If the reference is not configured the error shall not be reported.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME





	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.2.19 FrlfFrameTriggeringDemEventParameterRefs

[ECUC_Frlf_06099] Definition of EcucParamConfContainerDef FrlfFrameTriggeringDemEventParameterRefs [

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

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FRIF_E_LPDU_SLOTSTATUS	0..1	[ECUC_Frlf_00009]

No Included Containers

]

[ECUC_Frlf_00009] Definition of EcucReferenceDef FRIF_E_LPDU_SLOTSTATUS [

Parameter Name	FRIF_E_LPDU_SLOTSTATUS		
Parent Container	FrlfFrameTriggeringDemEventParameterRefs		
Description	Reference to DEM event Id that is reported when FlexRay driver module detects slot errors. If this parameter is not configured, no event reporting happens.		
Multiplicity	0..1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Dependency			

]

10.3 Published Information

For details refer to [\[2\]](#) Chapter 10.3 “*Published Information*”.

A Not applicable requirements

[SWS_Frlf_NA_06118]

Upstream requirements: SRS_BSW_00159, SRS_BSW_00167, SRS_BSW_00416, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00417, SRS_BSW_00386, SRS_BSW_00306, SRS_BSW_00369, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00403, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491, SRS_Fr_05009

[These requirements are not applicable to this specification.]

B Change history of AUTOSAR traceable items

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

B.1 Traceable item history of this document according to AUTOSAR Release R23-11

B.1.1 Added Specification Items in R23-11

none

B.1.2 Changed Specification Items in R23-11

none

B.1.3 Deleted Specification Items in R23-11

none

B.2 Traceable item history of this document according to AUTOSAR Release R24-11

B.2.1 Added Specification Items in R24-11

[\[SWS_Frlf_05320\]](#) [\[SWS_Frlf_05321\]](#) [\[SWS_Frlf_05322\]](#) [\[SWS_Frlf_05323\]](#) [\[SWS_Frlf_05324\]](#) [\[SWS_Frlf_05325\]](#) [\[SWS_Frlf_05326\]](#)

B.2.2 Changed Specification Items in R24-11

[\[ECUC_Frlf_05373\]](#) [\[ECUC_Frlf_05374\]](#) [\[SWS_Frlf_05001\]](#) [\[SWS_Frlf_05043\]](#) [\[SWS_Frlf_05044\]](#) [\[SWS_Frlf_05093\]](#) [\[SWS_Frlf_05146\]](#) [\[SWS_Frlf_05207\]](#) [\[SWS_Frlf_05287\]](#) [\[SWS_Frlf_05288\]](#) [\[SWS_Frlf_05291\]](#) [\[SWS_Frlf_05293\]](#) [\[SWS_Frlf_05423\]](#) [\[SWS_Frlf_05426\]](#) [\[SWS_Frlf_05427\]](#) [\[SWS_Frlf_05428\]](#) [\[SWS_Frlf_05429\]](#) [\[SWS_Frlf_05430\]](#) [\[SWS_Frlf_05431\]](#) [\[SWS_Frlf_05435\]](#) [\[SWS_Frlf_05501\]](#) [\[SWS_Frlf_05705\]](#) [\[SWS_Frlf_05757\]](#)

B.2.3 Deleted Specification Items in R24-11

[ECUC_Frlf_00014] [ECUC_Frlf_00015] [ECUC_Frlf_00016] [ECUC_Frlf_00017]
[ECUC_Frlf_06084] [SWS_Frlf_05045] [SWS_Frlf_05046] [SWS_Frlf_05047] [SWS_Frlf_05434]
[SWS_Frlf_05729] [SWS_Frlf_05730] [SWS_Frlf_05731] [SWS_Frlf_05732] [SWS_Frlf_05733]
[SWS_Frlf_05734] [SWS_Frlf_05735] [SWS_Frlf_05736] [SWS_Frlf_05737] [SWS_Frlf_05738]
[SWS_Frlf_05739] [SWS_Frlf_05740] [SWS_Frlf_05741] [SWS_Frlf_05742] [SWS_Frlf_05743]
[SWS_Frlf_05759]

B.3 Traceable item history of this document according to AUTOSAR Release R25-11

B.3.1 Added Specification Items in R25-11

none

B.3.2 Changed Specification Items in R25-11

[ECUC_Frlf_00001]	[ECUC_Frlf_00002]	[ECUC_Frlf_00003]	[ECUC_Frlf_00004]
[ECUC_Frlf_00007]	[ECUC_Frlf_00008]	[ECUC_Frlf_00009]	[ECUC_Frlf_00010]
[ECUC_Frlf_00011]	[ECUC_Frlf_00012]	[ECUC_Frlf_00013]	[ECUC_Frlf_05360]
[ECUC_Frlf_05363]	[ECUC_Frlf_05364]	[ECUC_Frlf_05366]	[ECUC_Frlf_05367]
[ECUC_Frlf_05368]	[ECUC_Frlf_05369]	[ECUC_Frlf_05370]	[ECUC_Frlf_05371]
[ECUC_Frlf_05372]	[ECUC_Frlf_05373]	[ECUC_Frlf_05374]	[ECUC_Frlf_05391]
[ECUC_Frlf_06001]	[ECUC_Frlf_06002]	[ECUC_Frlf_06003]	[ECUC_Frlf_06004]
[ECUC_Frlf_06006]	[ECUC_Frlf_06008]	[ECUC_Frlf_06009]	[ECUC_Frlf_06010]
[ECUC_Frlf_06011]	[ECUC_Frlf_06012]	[ECUC_Frlf_06013]	[ECUC_Frlf_06014]
[ECUC_Frlf_06015]	[ECUC_Frlf_06018]	[ECUC_Frlf_06019]	[ECUC_Frlf_06020]
[ECUC_Frlf_06021]	[ECUC_Frlf_06024]	[ECUC_Frlf_06025]	[ECUC_Frlf_06026]
[ECUC_Frlf_06027]	[ECUC_Frlf_06032]	[ECUC_Frlf_06033]	[ECUC_Frlf_06034]
[ECUC_Frlf_06035]	[ECUC_Frlf_06036]	[ECUC_Frlf_06037]	[ECUC_Frlf_06038]
[ECUC_Frlf_06039]	[ECUC_Frlf_06040]	[ECUC_Frlf_06041]	[ECUC_Frlf_06042]
[ECUC_Frlf_06043]	[ECUC_Frlf_06044]	[ECUC_Frlf_06045]	[ECUC_Frlf_06048]
[ECUC_Frlf_06049]	[ECUC_Frlf_06050]	[ECUC_Frlf_06051]	[ECUC_Frlf_06052]
[ECUC_Frlf_06053]	[ECUC_Frlf_06054]	[ECUC_Frlf_06055]	[ECUC_Frlf_06056]
[ECUC_Frlf_06057]	[ECUC_Frlf_06058]	[ECUC_Frlf_06061]	[ECUC_Frlf_06062]
[ECUC_Frlf_06063]	[ECUC_Frlf_06064]	[ECUC_Frlf_06065]	[ECUC_Frlf_06066]
[ECUC_Frlf_06067]	[ECUC_Frlf_06068]	[ECUC_Frlf_06069]	[ECUC_Frlf_06070]
[ECUC_Frlf_06071]	[ECUC_Frlf_06072]	[ECUC_Frlf_06073]	[ECUC_Frlf_06074]
[ECUC_Frlf_06075]	[ECUC_Frlf_06076]	[ECUC_Frlf_06077]	[ECUC_Frlf_06078]
[ECUC_Frlf_06080]	[ECUC_Frlf_06081]	[ECUC_Frlf_06082]	[ECUC_Frlf_06083]
[ECUC_Frlf_06086]	[ECUC_Frlf_06087]	[ECUC_Frlf_06090]	[ECUC_Frlf_06091]
[ECUC_Frlf_06093]	[ECUC_Frlf_06094]	[ECUC_Frlf_06095]	[ECUC_Frlf_06096]
[ECUC_Frlf_06097]	[ECUC_Frlf_06098]	[ECUC_Frlf_06099]	[ECUC_Frlf_06101]

[ECUC_Frlf_06102] [ECUC_Frlf_06103] [ECUC_Frlf_06104] [ECUC_Frlf_06105]
[ECUC_Frlf_06106] [ECUC_Frlf_06107] [ECUC_Frlf_06108] [ECUC_Frlf_06109]
[ECUC_Frlf_06110] [ECUC_Frlf_06111] [ECUC_Frlf_06112] [ECUC_Frlf_06113]
[ECUC_Frlf_06114] [ECUC_Frlf_06116] [ECUC_Frlf_06117] [ECUC_Frlf_06118]
[ECUC_Frlf_06119] [ECUC_Frlf_06120] [ECUC_Frlf_06121] [ECUC_Frlf_06122]
[ECUC_Frlf_06123] [ECUC_Frlf_06124] [SWS_Frlf_05002] [SWS_Frlf_05003]
[SWS_Frlf_05087] [SWS_Frlf_05111] [SWS_Frlf_05301] [SWS_Frlf_05303] [SWS_
Frlf_05321] [SWS_Frlf_05322] [SWS_Frlf_05323] [SWS_Frlf_05324] [SWS_Frlf_
05325] [SWS_Frlf_05326] [SWS_Frlf_05755]

B.3.3 Deleted Specification Items in R25-11

none