

Document Title	Specification of FlexRay Interface
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	027
Document Classification	Standard

Document Version	3.4.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History			
Date	Version	Changed by	Change Description
28.02.2014	3.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> • FrlfGAssumedPrecision set to deprecated/obsolete • Added Frlf_GetWakeupRxStatus() • Several minor corrections • Editorial changes • Removed chapter(s) on change documentation
29.05.2012	3.3.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Extension of FrlfMaxIsrDelay (Configurable per Job) • Extension for Vendor Specific APIs (FREE_OP_A & FREE_OP_B)
27.04.2011	3.2.0	AUTOSAR Administration	<ul style="list-style-type: none"> • UnusedBidValue functionality added • API Frlf_GetClockCorrection added • API Frlf_GetChannelStatus added • API Frlf_ReadCCConfig added • User defined ComOps added
20.09.2010	3.1.0	AUTOSAR Administration	<ul style="list-style-type: none"> • Update sequence chart 9-6 • Extension of Frlf05063 (behavior in case of return value E_NOT_OK for API TriggerTransmit())
23.06.2008	3.0.2	AUTOSAR Administration	Legal disclaimer revised
22.01.2008	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> • Correction of Figure 5.1

18.12.2007	3.0.0	AUTOSAR Administration	<ul style="list-style-type: none"> • 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
06.02.2007	2.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> • "Advice for users" revised • Legal disclaimer added • "Revision Information" added • Release Notes added
30.06.2006	2.0.0	AUTOSAR Administration	Second Release
19.09.2005	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

1	Introduction and Functional Overview	8
2	Information about this Document.....	10
2.1	General Hints	10
2.2	Acronyms and Abbreviations.....	11
3	Related Documentation	12
3.1	Input Documents	12
3.2	Related Standards and Norms	13
4	Constraints and Assumptions.....	14
4.1	Limitations	14
4.2	Applicability to Car Domains	14
5	Dependencies to Other Modules	15
5.1	AUTOSAR Operating System	15
5.2	AUTOSAR BSW Scheduler.....	15
5.3	All Upper Layer AUTOSAR BSW Modules.....	15
5.4	AUTOSAR PDU-Router	15
5.5	AUTOSAR FlexRay Network Management.....	16
5.6	AUTOSAR FlexRay Transport Protocol	16
5.7	AUTOSAR FlexRay Driver	16
5.8	AUTOSAR FlexRay Transceiver Driver.....	16
5.9	AUTOSAR Development Error Tracer.....	16
5.10	AUTOSAR Diagnostic Event Manager	16
5.11	File Structure.....	17
5.11.1	Code File Structure	17
5.11.2	Header File Structure	17
6	Requirements Traceability	19
6.1	General Requirements on Basic Software Modules	19
6.2	Requirements on FlexRay.....	21
6.3	Specification Items	22
7	Functional Specification.....	24
7.1	FlexRay BSW Stack.....	24
7.2	Indexing Scheme.....	24
7.2.1	Principle	24
7.2.2	Supported Indexed Resources.....	27
7.3	FlexRay Interface State Machine	27
7.3.1	FlexRay Interface Main Function.....	29
7.4	Implementation Requirements	31
7.5	Configuration description.....	32
7.6	Data Communication via FlexRay	33
7.6.1	PDU Packing, PDU Update-Bits, and Frame Construction Plans	33
7.6.1.1	AlwaysTransmit	35
7.6.2	Realization of the Time-Driven FlexRay Schedule	36
7.6.2.1	FlexRay Job List.....	36

7.6.2.2	FlexRay Job List Execution Function.....	37
7.6.3	Communication Operations.....	39
7.6.3.1	TransmitWithDecoupledBufferAccess	39
7.6.3.2	ProvideTxConfirmation	40
7.6.3.3	ReceiveAndStore.....	40
7.6.3.4	ProvideRxIndication.....	41
7.6.3.5	ReceiveAndIndicate.....	41
7.6.3.6	PREPARE_LPDU.....	42
7.6.3.7	FREE_OP_A	42
7.6.3.8	FREE_OP_B	42
7.6.4	Transmission with Immediate Buffer Access.....	42
7.7	Error Classification	43
7.8	Error Detection	44
7.9	Error Notification.....	44
8	API Service Specification	45
8.1	Imported types.....	45
8.2	Type Definitions.....	45
8.2.1	Frlf_ConfigType	45
8.2.2	Frlf_StateType	46
8.2.3	Frlf_StateTransitionType.....	46
8.3	Function Definitions.....	46
8.3.1	Frlf_GetVersionInfo	47
8.3.2	Frlf_Init.....	48
8.3.3	Frlf_ControllerInit	49
8.3.4	Frlf_StartCommunication	49
8.3.5	Frlf_HaltCommunication	50
8.3.6	Frlf_AbortCommunication	51
8.3.7	Frlf_GetState.....	51
8.3.8	Frlf_SetState	52
8.3.9	Frlf_SetWakeupChannel.....	53
8.3.10	Frlf_SendWUP	54
8.3.11	Frlf_GetSyncState.....	55
8.3.12	Frlf_SetExtSync	56
8.3.13	Frlf_GetPOCStatus	57
8.3.14	Frlf_GetGlobalTime.....	57
8.3.15	Frlf_AllowColdstart.....	58
8.3.16	Frlf_GetMacroticksDuration	59
8.3.17	Frlf_Transmit.....	60
8.3.18	Frlf_SetTransceiverMode.....	61
8.3.19	Frlf_GetTransceiverMode	62
8.3.20	Frlf_GetTransceiverWUReason	63
8.3.21	Frlf_EnableTransceiverWakeup.....	64
8.3.22	Frlf_DisableTransceiverWakeup.....	65
8.3.23	Frlf_ClearTransceiverWakeup	66
8.3.24	Frlf_GetCycleLength	67
8.4	Optional Function Definitions	68
8.4.1	Frlf_SetAbsoluteTimer	68
8.4.2	Frlf_SetRelativeTimer	69
8.4.3	Frlf_CancelAbsoluteTimer.....	69

8.4.4	Frlf_CancelRelativeTimer	70
8.4.5	Frlf_EnableAbsoluteTimerIRQ	71
8.4.6	Frlf_EnableRelativeTimerIRQ	72
8.4.7	Frlf_GetAbsoluteTimerIRQStatus	73
8.4.8	Frlf_GetRelativeTimerIRQStatus	74
8.4.9	Frlf_AckAbsoluteTimerIRQ	75
8.4.10	Frlf_AckRelativeTimerIRQ	76
8.4.11	Frlf_DisableAbsoluteTimerIRQ	77
8.4.12	Frlf_DisableRelativeTimerIRQ	77
8.4.13	Frlf_GetNmVector	78
8.4.14	Frlf_GetClockCorrection	79
8.4.15	Frlf_GetChannelStatus	80
8.4.16	Frlf_ReadCCConfig	80
8.4.17	Frlf_GetWakeupRxStatus	81
8.5	Interrupt Service Routines	83
8.5.1	Frlf_JobListExec_<ClstIdx>	83
8.6	Call-back Notifications	84
8.6.1	Frlf_Cbk_WakeupByTransceiver	84
8.7	Scheduled Functions	85
8.7.1	Frlf_MainFunction_<ClstIdx>	85
8.8	Expected Interfaces	86
8.8.1	Mandatory Interfaces	86
8.8.2	Optional Interfaces	87
8.8.3	Configurable Interfaces	87
8.8.3.1	<UL_RxIndication>	87
8.8.3.2	<UL_TxConfirmation>	88
8.8.3.3	<UL_TriggerTransmit>	88
8.8.3.4	<Free_Op_A>	90
8.8.3.5	<Free_Op_B>	90
9	Sequence Diagrams	91
9.1	Data Transmission	91
9.1.1	TransmitWithImmediateBufferAccess	91
9.1.2	TransmitWithDecoupledBufferAccess	92
9.1.3	ProvideTxConfirmation	93
9.2	Data Reception	94
9.2.1	ReceiveAndIndicate	94
9.2.2	ReceiveAndStore	95
9.2.3	ProvideRxIndication	96
10	Configuration Specification	97
10.1	How to Read this Chapter	97
10.1.1	Configuration and Configuration Parameters	97
10.1.2	Variants	97
10.1.3	Containers	98
10.1.4	Specification Template for Configuration Parameters	98
10.2	Containers and Configuration Parameters	99
10.2.1	Variants	99
10.2.2	Frlf	99
10.2.3	FrlfGeneral	100
10.2.4	FrlfCluster	105

10.2.5	FrlfController	117
10.2.6	FrlfAbsTimer	117
10.2.7	FrlfRelTimer	118
10.2.8	FrlfFrameTriggering	119
10.2.9	FrlfJobList	124
10.2.10	FrlfFrameStructure	125
10.2.11	FrlfPduInFrame.....	126
10.2.12	FrlfLPdu.....	127
10.2.13	FrlfPdu.....	127
10.2.14	FrlfTxPdu.....	128
10.2.15	FrlfRxPdu	130
10.2.16	FrlfPduDirection.....	131
10.2.17	FrlfTransceiver	132
10.2.18	FrlfConfig.....	133
10.3	Published Information.....	134

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 [2], 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).

Frlf05095:

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)

Frlf05096:

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.

Frlf05097:

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

2 Information about this Document

2.1 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, unless stated otherwise, 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

Frlf05098:

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

One example for such a non-AUTOSAR software module is a proprietary XCPonFlexRay module that users may add to their AUTOSAR BSW stack.

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.
- "**by a flashing process**" = data (not code!) manipulation carried out in a *flashing process* of a flashable memory (in general a Flash-EEPROM) e.g. in a garage, but **not** while the car is being driven. Usually used to **replace** a static configuration *already stored* in the ECU, or a part thereof. Therefore, the respective data are of configuration class "link time" or "post build time".
- "**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.

2.2 Acronyms and Abbreviations

The following acronyms and abbreviations are used throughout this document:

Acronym:	Description:
FrIf	FlexRay Interface (AUTOSAR BSW module)
CC	(FlexRay) Communication Controller
WUP	Wake-Up Pattern
WUS	Wake-Up Symbol
CAS	Collision Avoidance Symbol
MTS	Media Access Test Symbol
POC	Protocol Operation Control
CHI	Controller Host Interface of a FlexRay CC
BSW	(AUTOSAR) Basic Software
ISR	Interrupt Service Routine
COM	Communication (AUTOSAR BSW module)
PduR	PDU Router (AUTOSAR BSW module)
FrTp	FlexRay Transport Layer (AUTOSAR BSW module)
FrNm	FlexRay Network Management (AUTOSAR BSW module)
DEM	Diagnostic Event Manager (AUTOSAR BSW module)
DET	Development Error Tracer (AUTOSAR BSW module)
CDD	Complex Device Driver
ComM	Communication Manager (AUTOSAR BSW module)
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 <i>which operations</i> this executable code performs on <i>which data</i> and at <i>which points in time</i> .

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

- [1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf
- [2] AUTOSAR Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf
- [3] AUTOSAR General Requirements on Basic Software Modules
AUTOSAR_SRS_General.pdf
- [4] Specification of AUTOSAR COM
AUTOSAR_SWS_COM.pdf
- [5] AUTOSAR Requirements on FlexRay
AUTOSAR_SRS_FlexRay.pdf
- [6] Specification of FlexRay Driver
AUTOSAR_SWS_FlexRay_Driver.pdf
- [7] FlexRay State Manager
AUTOSAR_SWS_FlexRayStateManager.pdf
- [8] Specification of FlexRay Transceiver Driver
AUTOSAR_SWS_FlexRayTransceiver.pdf
- [9] Specification of FlexRay Transport Layer
AUTOSAR_SWS_FlexRay_TP.pdf
- [10] Specification of FlexRay Network Management
AUTOSAR_SWS_FlexRay_NM.pdf
- [11] Specification of PDU Router
AUTOSAR_SWS_PDU_Router.pdf
- [12] Specification of BSW Scheduler
AUTOSAR_SWS_Scheduler.pdf
- [13] ECU Configuration Specification
AUTOSAR_SWS_ECU_StateManager.pdf
- [14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf
- [15] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related Standards and Norms

- [16] FlexRay Communications System Protocol Specification Version 2.1
Revision A
- [17] FlexRay Communications System Electrical Physical Layer Specification
Version 2.1 Revision A

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.

Frlf05008:

In order for the AUTOSAR FlexRay [BSW](#) ([Frlf](#) 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)

$$\text{Cycle Number} = (\mathbf{B} + \mathbf{n} * 2^{\mathbf{R}})_{\text{mod}64}$$

with **exactly one tuple** of values for **B** and $2^{\mathbf{R}}$, where:

- Base Cycle **B** $\in [0 \dots 63]$
- Cycle Repetition $2^{\mathbf{R}}$; $\mathbf{R} \in [0 \dots 6]$
- Variable **n** = 0 ... 63
- **B** < $2^{\mathbf{R}}$

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

Frlf05074:

5.1 AUTOSAR Operating System

Frlf05099:

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

Frlf05100:

The FlexRay Interface executes the Flexray Job List Execution Function.

Note:

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

5.2 AUTOSAR BSW Scheduler

Frlf05101:

There is one dedicated FlexRay Interface Main Function for each FlexRay Cluster

Frlf05102:

Each of these FlexRay Interface Main Functions must be called cyclically from a task body provided by the [BSW](#) Scheduler. The calling period must be configurable.

5.3 All Upper Layer AUTOSAR BSW Modules

Frlf05050:

The calling of the FlexRay Job List Execution Function 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.

Frlf05049:

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. Therefore, this [BSW](#) module shall allow access to its PDU buffers at all times and it also has to ensure data consistency in its buffers.

5.4 AUTOSAR PDU-Router

The [Frlf](#) declares and calls some callback functions of the PDU-Router in order to confirm transmission and notify reception of PDUs.

5.5 AUTOSAR FlexRay Network Management

The [Frlf](#) declares and calls some callback functions of the FlexRay Network Management in order to confirm transmission and notify reception of PDUs.

5.6 AUTOSAR FlexRay Transport Protocol

The [Frlf](#) declares and calls some callback functions of the FlexRay Transport Protocol in order to confirm transmission and notify reception of PDUs.

5.7 AUTOSAR FlexRay Driver

The [Frlf](#) has a tight relation to the FlexRay Driver since many of the FlexRay-related services offered by the [Frlf](#) to upper layer [BSW](#) modules are actually carried out by the FlexRay Driver [BSW](#) module. For those services, the [Frlf](#) 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.

Frlf05051:

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

5.8 AUTOSAR FlexRay Transceiver Driver

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

5.9 AUTOSAR Development Error Tracer

Frlf05065:

In order to be able to report development errors, the [Frlf](#) has to have access to the error hook of the Development Error Tracer.

5.10 AUTOSAR Diagnostic Event Manager

Frlf05066:

In order to be able to report production errors, the [Frlf](#) has to have access to the Diagnostic Event Manager.

5.11 File Structure

5.11.1 Code File Structure

The code file structure shall not completely be defined within this specification.

Frlf05075:

However, the code-file structure shall include the following files:

Frlf.c	general source code file of the FlexRay Interface
Frlf_Cfg.c	contains pre-compile time configurable parameters
Frlf_Lcfg.c	contains link time configurable parameters
Frlf_PBcfg.c	contains post build time configurable parameters

5.11.2 Header File Structure

Frlf05076:

The header file structure shall contain the following header files:

Frlf.h	general header file of the FlexRay Interface
Frlf_Cfg.h	contains the Frlf pre-compile-time configurable parameters (pre processor constants)
Frlf_Cbk.h	contains the declarations of the callback functions provided by the Frlf to other BSW modules
Fr.h	contains the declarations of the API services of the FlexRay Driver used by the FlexRay Interface
FrTrcv.h	contains the declarations of the API services of the FlexRay Transceiver Driver used by the FlexRay Interface.
Fr_Types.h	contains declarations shared by all AUTOSAR FlexRay BSW modules
CDD_Cbk.h	Contains the callback declaration of CDDs, configurable via configuration parameter with FlexRay Interface
ComStack_Types.h	Contains the communication module abstracted datatypes shared by AUTOSAR communication BSW.
PduR_Frlf.h	contains the declarations of API services the PDU router offers to the FlexRay Interface
FrNm_Cbk.h	contains the declarations of API services the FrNm offers to the FlexRay Interface
FrTp_Cbk.h	contains the declarations of API services the FrTp offers to the FlexRay Interface
Dem.h	contains the declarations of the API services of the Dem used by the FlexRay Interface
Det.h	contains the declarations of the API services of the Det optionally used by the FlexRay Interface
SchM_Frlf.h	Contains the declaration of the API services the SchM offers to the FlexRay Interface
MemMap.h	

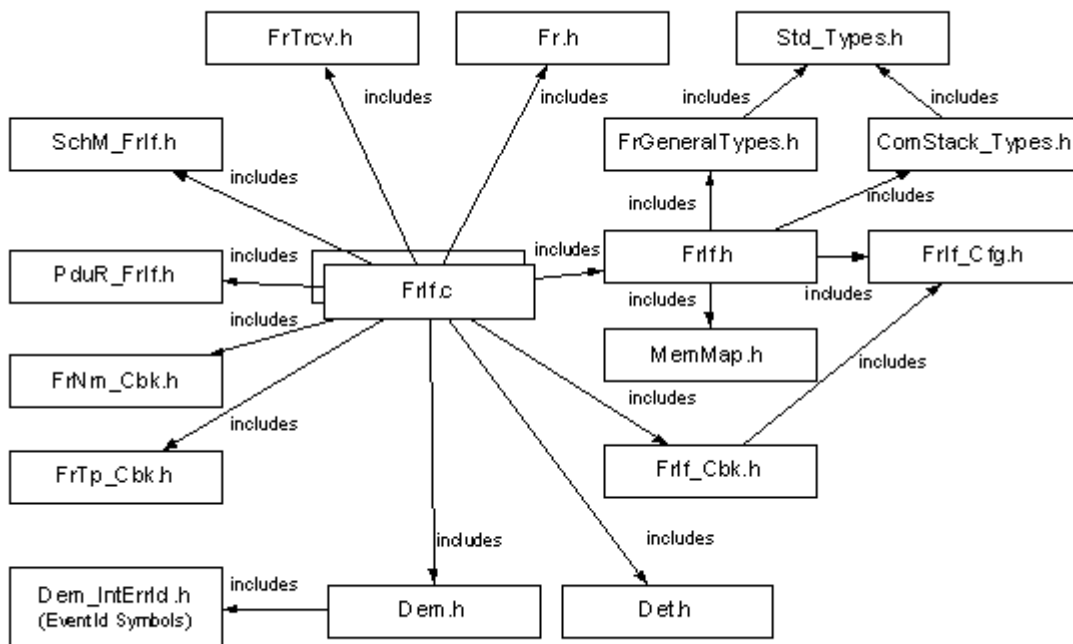


Figure 5-1: FlexRay Interface Header File Structure

Besides the FlexRay-specific header files, the FlexRay Interface shall include the file `Dem.h`. By this inclusion, the API services used to report errors as well as the required Event-Id symbols are included. The specification in hand defines the name of the Event-Id symbols, which are provided in XML format to the [DEM](#) configuration tool. The [DEM](#) configuration tool assigns ECU-dependent values to the Event-Id symbols and publishes the symbols in `Dem_IntErrId.h`.

FrIf05081:

All files related to the `FrIf` module shall follow the naming convention `FrIf[<description>].<extension>`

FrIf05091:

The `FrIf` module shall include the `Dem.h` file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in `Dem_IntErrId.h`.

FrIf05140:

The implementation of the `FrIf` module shall provide the header file `FrIf.h`, which is the main module interface file. It shall contain all types and function prototypes required by the `FrIf` module's environment.

FrIf05141:

The implementation of the `FrIf` module shall provide the header file `FrIf_Cfg.h` that shall contain the pre-compile-time configuration parameters.

6 Requirements Traceability

6.1 General Requirements on Basic Software Modules

Requirement	Satisfied by
BSW00344 Reference to link-time configuration	FrIf05068
BSW00404 Reference to post build time configuration	FrIf05069
BSW00405 Reference to multiple configuration sets	FrIf05003
BSW00345 Pre-compile-time configuration	FrIf05069
BSW159 Tool-based configuration	FrIf05070
BSW167 Static configuration checking	FrIf05071, FrIf05072, FrIf05073
BSW171 Configurability of optional functionality	FrIf05089
BSW170 Data for reconfiguration of AUTOSAR SW-Components	FrIf05089
BSW00380 Separate C-Files for configuration parameters	FrIf05075
BSW00419 Separate C-Files for pre-compile time configuration parameters	FrIf05075
BSW00381 Separate configuration header file for pre-compile time parameters	FrIf05075
BSW00412 Separate H-File for configuration parameters	FrIf05076
BSW00383 List dependencies of configuration files	FrIf05075
BSW00384 List dependencies to other modules	FrIf05074
BSW00387 Specify the configuration class of callback function	n/a
BSW00388 Introduce containers	FrIf05077
BSW00389 Containers shall have names	Chapter 10.2
BSW00390 Parameter content shall be unique within the module	Chapter 10.2
BSW00391 Parameter shall have unique names	Chapter 10.2
BSW00392 Parameters shall have a type	Chapter 10.2
BSW00393 Parameters shall have a range	Chapter 10.2
BSW00394 Specify the scope of the parameters	Chapter 10.2
BSW00395 List the required parameters (per parameter)	Chapter 10.2
BSW00396 Configuration classes	Chapter 10.2
BSW00397 Pre-compile-time parameters	Chapter 10.2
BSW00398 Link-time parameters	Chapter 10.2
BSW00399 Loadable Post-build time parameters	Chapter 10.2
BSW00400 Selectable Post-build time parameters	Chapter 10.2
BSW00402 Published information	FrIf05086
BSW00375 Notification of wake-up reason	
BSW101 Initialization interface	FrIf05003
BSW00416 Sequence of Initialization	n/a
BSW00406 Check module initialization	FrIf05002
BSW168 Diagnostic Interface of SW components	n/a
BSW00407 Function to read out published parameters	FrIf05002
BSW00423 Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	
BSW00424 BSW main processing function task allocation	n/a
BSW00425 Trigger conditions for schedulable objects	
BSW00426 Exclusive areas in BSW modules	n/a
BSW00427 ISR description for BSW modules	n/a
BSW00428 Execution order dependencies of main processing functions	n/a
BSW00429 Restricted BSW OS functionality access	n/a
BSW00431 The BSW Scheduler module implements task bodies	n/a
BSW00432 Modules should have separate main processing functions for read/receive and write/transmit data path	n/a
BSW00433 Calling of main processing functions	
BSW00434 The Schedule Module shall provide an API for exclusive areas	n/a

BSW00336	Shutdown interface	FrIf05006
BSW00337	Classification of errors	FrIf05139
BSW00338	Detection and Reporting of development errors	Chapter 8.3
BSW00369	Do not return development error codes via API	Chapter 8.3
BSW00339	Reporting of production relevant error status	Chapter 8.3
BSW00417	Reporting of Error Events by Non-Basic Software	n/a
BSW00323	API parameter checking	Chapter 8.3
BSW004	Version check	FrIf05090
BSW00409	Header files for production code error IDs	FrIf05091
BSW00385	List possible error notifications	Chapter 8.3
BSW00386	Configuration for detecting an error	n/a
BSW161	Microcontroller abstraction	n/a
BSW162	ECU layout abstraction	n/a
BSW005	No hard coded horizontal interfaces within MCAL	n/a
BSW00415	User dependent include files	n/a
BSW164	Implementation of interrupt service routines	n/a
BSW00325	Runtime of interrupt service routines	n/a
BSW00326	Transition from ISRs to OS tasks	n/a
BSW00342	Usage of source code and object code	FrIf05078
BSW00343	Specification and configuration of time	FrIf05141
BSW160	Human-readable configuration data	FrIf05079
BSW007	HIS MISRA C	FrIf05080
BSW00300	Module naming convention	FrIf05081
BSW00413	Accessing instances of BSW modules	n/a
BSW00347	Naming separation of different instances of BSW drivers	n/a
BSW00305	Self-defined data types naming convention	FrIf05082
BSW00307	Global variables naming convention	FrIf05083
BSW00310	API naming convention	FrIf05083
BSW00373	Main processing function naming convention	n/a
BSW00327	Error values naming convention	FrIf05142
BSW00335	Status values naming convention	n/a
BSW00350	Development error detection keyword	FrIf05084
BSW00408	Configuration parameter naming convention	
BSW00410	Compiler switches shall have defined values	n/a
BSW00411	Get version info keyword	FrIf05002
BSW00346	Basic set of module files	FrIf05075
BSW158	Separation of configuration from implementation	FrIf05075
BSW00314	Separation of interrupt frames and service routines	n/a
BSW00370	Separation of callback interface from API	n/a
BSW00348	Standard type header	FrIf05001
BSW00353	Platform specific type header	FrIf05001
BSW00361	Compiler specific language extension header	FrIf05001
BSW00301	Limit imported information	Figure 5-1
BSW00302	Limit exported information	FrIf05141
BSW00328	Avoid duplication of code	n/a
BSW00312	Shared code shall be reentrant	n/a
BSW006	Platform independency	n/a
BSW00357	Standard API return type	Chapter 8.3
BSW00377	Module specific API return types	n/a
BSW00304	AUTOSAR integer data types	FrIf05001
BSW00355	Do not redefine AUTOSAR integer data types	FrIf05001
BSW00378	AUTOSAR boolean type	FrIf05001
BSW00306	Avoid direct use of compiler and platform specific keywords	n/a
BSW00308	Definition of global data	FrIf05143
BSW00309	Global data with read-only constraint	FrIf05068
BSW00371	Do not pass function pointers via API	n/a
BSW00358	Return type of init() functions	FrIf05003
BSW00414	Parameter of init function	FrIf05003

BSW00376	Return type and parameters of main processing functions	n/a
BSW00359	Return type of callback functions	Chapter 8.7.3
BSW00360	Parameters of callback functions	Chapter 8.7.3
BSW00329	Avoidance of generic interfaces	n/a
BSW00330	Usage of macros / inline functions instead of functions	n/a
BSW00331	Separation of error and status values	n/a
BSW009	Module User Documentation	n/a
BSW00401	Documentation of multiple instances of configuration parameters	
BSW172	Compatibility and documentation of scheduling strategy	n/a
BSW010	Memory resource documentation	n/a
BSW00333	Documentation of callback function context	n/a
BSW00374	Module vendor identification	FrIf 05086
BSW00379	Module identification	FrIf 05086
BSW003	Version identification	FrIf 05086
BSW00318	Format of module version numbers	FrIf 05086
BSW00321	Enumeration of module version numbers	FrIf 05086
BSW00341	Microcontroller compatibility documentation	n/a
BSW00334	Provision of XML file	FrIf05089
BSW00435	Header File Structure for the Basic Software Scheduler	FrIf05087
BSW00436	Module Header File Structure for the Memory Mapping	FrIf05088

6.2 Requirements on FlexRay

Requirement	Satisfied by
BSW05000 Support of Synchronous SW Modules	FrIf05050
BSW05001 Support of Asynchronous SW Modules	FrIf05049
BSW05002 FlexRay Interface and FlexRay Driver as Only Necessarily Synchronous SW Modules	FrIf05051
BSW05003 Support of Slot/Cycle Multiplexing	FrIf05008
BSW05169 Avoid Timer Interrupts during Start-up	8.3.2
BSW05055 Avoid Timer Interrupts during Shutdown	8.3.8, 8.3.9
BSW05004 PDU-Based Data API	FrIf05055
BSW05010 Unique PDU-ID	FrIf05052
BSW05126 PDU Update/Valid Information	FrIf05056
BSW05097 Number of FlexRay Drivers per FlexRay Interface	FrIf05057
BSW05007 Number of FlexRay CCs per Interface	FrIf05053
BSW05130 Transmit Request Queuing	FrIf05058
BSW05060 Scheduling of Copy Operation into/from FlexRay CC	FrIf05059
BSW05096 Assignment of Drivers to Controllers	FrIf05060
BSW05013 Initialize Local Memory Space	FrIf05003
BSW05031] Initialization of a FlexRay CC	FrIf05004
BSW05078 Initialization of a FlexRay Cluster	N/A (moved to FrSm)
BSW05034 Configuration Modifiable by a Flashing Process	
BSW05042 Switch Configuration in Normal Active Mode	FrIf05061
BSW05015 Start-up of a FlexRay CC	FrIf05005
BSW05101 Start-up of a FlexRay Cluster	N/A (moved to FrSm)
BSW05018 Sending of a Wake-Up Pattern	FrIf05011
BSW05158 Get FlexRay Transceiver Wake-up Reason	FrIf05036
BSW05159 Enable FlexRay Transceiver Wake-up Indication	FrIf05037
BSW05163 Cluster-wide Enable FlexRay Transceiver Wake-Up Indication	N/A (moved to FrSm)
BSW05160 Disable FlexRay Transceiver Wake-up	FrIf05038
BSW05164 Cluster-wide Disable FlexRay Transceiver Wake-up Indication	N/A (moved to FrSm)
BSW05161 Clear FlexRay Transceiver Wake-up Events	FrIf05039
BSW05165 Cluster-wide Clear FlexRay Transceiver Wake-up Events	N/A (moved to FrSm)
BSW05067 Set FlexRay Cluster Offline Mode	N/A
BSW05068 Set FlexRay Cluster Online Mode	N/A (moved to FrSm)
BSW05155 Set FlexRay Cluster Offline Mode	FrIf05005

BSW05069	Get FlexRay Cluster Mode	N/A (moved to FrSm)
BSW05153	Get FlexRay Controller Mode	N/A (moved to FrSm)
BSW05022	Get FlexRay CC POC Status	Frlf05014
BSW05023	Get FlexRay CC Sync State	Frlf05012
BSW05035	MTS Sending	N/A (no use without BG)
BSW05038	Get MTS Reception Status	N/A (no use without BG)
BSW05039	Set FlexRay Transceiver Operation Mode	Frlf05034
BSW05162	Set Cluster-wide FlexRay Transceiver Operation Mode	N/A (moved to FrSm)
BSW05157	Get FlexRay Transceiver Operation Mode	Frlf05035
BSW05174	Interrupt Handling	
BSW05170	Receive PDU	Frlf05062
BSW05027	Transmit PDU	Frlf05063
BSW05016	Abortion of a FlexRay CC Communication	Frlf05007
BSW05113	Abortion of a FlexRay Cluster Communication	N/A (moved to FrSm)
BSW05063	Halt of a FlexRay CC Communication	Frlf05006
BSW05102	Halt of a FlexRay Cluster Communication	N/A (moved to FrSm)
BSW05102	Provide Error Information	Frlf05065, Frlf05066
BSW05006	Abstraction of FlexRay-Specific Features	
BSW05009	Local Memory Space Usage	
BSW05056	Configuration of the FlexRay Interface at System Configuration Time	Frlf05067, Frlf05054

6.3 Specification Items

The following Items shall be seen as implementation hints only!

Functional Specification

Usage of abstract index	Frlf05103
Passing of translated index	Frlf05104
Abstraction of FlexRay Transceivers	Frlf05105
Usage of Controller and Channel Index	Frlf05106
Usage of zero-based index	Frlf05107
Usage of FR Cluster Index	Frlf05108
Configuration Data	Frlf05109
Usage of PDU index	Frlf05110
Support one of both or both FlexRay Channels	Frlf05111
Support of at least four FlexRay Clusters	Frlf05112
Support of at least one absolute timer per FlexRay CCs	Frlf05113
Support of at least one relative timer per FlexRay CC	Frlf05114

FlexRay Interface State Machine

One State Machine per Cluster	Frlf05115
Frlf_State offline during initialization	Frlf05116
Frlf_State offline during initialization	Frlf05117

FlexRay Interface Main Function

One Main Function for each FlexRay Cluster	Frlf05119
Main Function tasks	Frlf05120

Data Communication via FlexRay

Packaging of multiple PDUs in one FR Frame	Frlf05121
Frame construction plan (layout)	Frlf05122
Frame construction plan (config)	Frlf05123
Transmission rule	Frlf05124
Update Information per PDU	Frlf05125
Location of Update Information	Frlf05126
Configuration of Update Information	Frlf05127
Indication in case of no update information	Frlf05128
Transmission with Immediate Buffer Access	Frlf05129
Ensure synchronous buffer access	Frlf05130
Sortation of Communication Job	Frlf05131
Communication Job properties	Frlf05132
Communication Job execution start time	Frlf05133
Actions specified by Communication Operation	Frlf05134
Communication Operation properties	Frlf05135
Job List Execution Function nameing	Frlf05136
Job List synchronously to global time	Frlf05137
Job List Execution Function actions	Frlf05138

7 Functional Specification

7.1 FlexRay BSW Stack

As part of the AUTOSAR Layered Software Architecture according to [2], the FlexRay [BSW](#) modules also form a layered software stack.

Figure 7-1 depicts the basic structure of this FlexRay [BSW](#) stack. The [Frlf](#) 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.

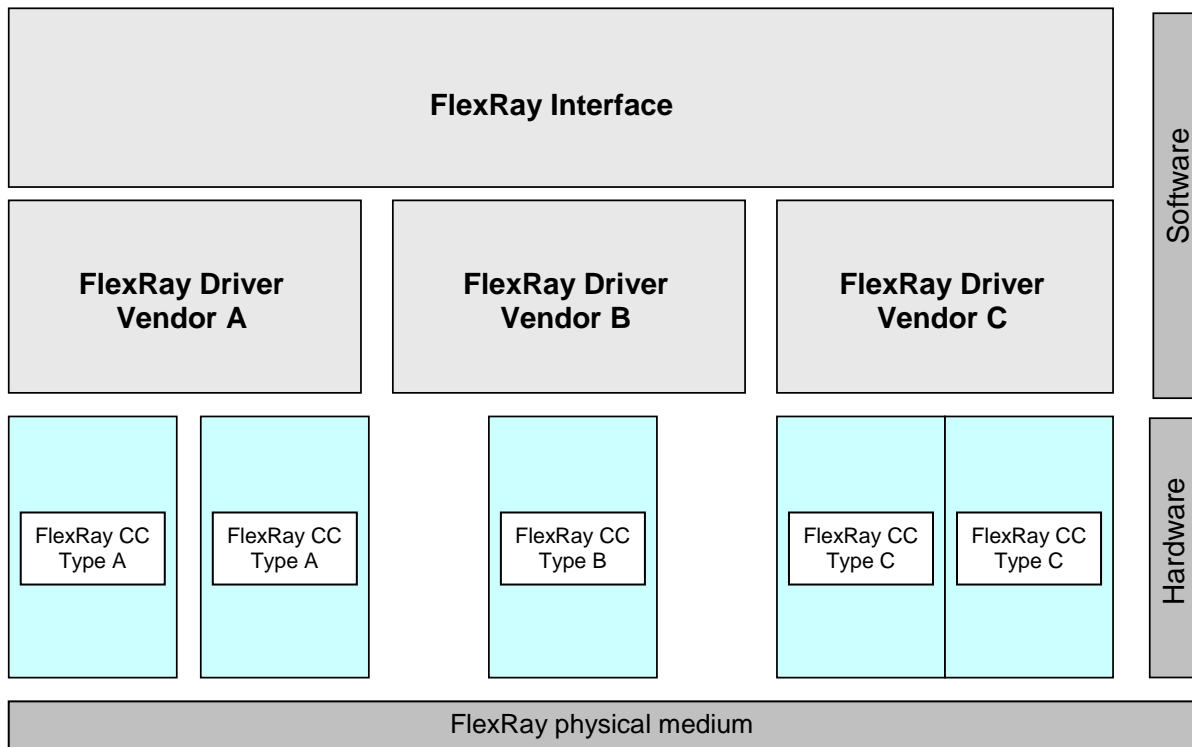


Figure 7-1: Basic Structure of the FlexRay BSW Stack

7.2 Indexing Scheme

7.2.1 Principle

Most of the [Frlf](#)'s API services used for accessing the numerous (hardware and software) resources¹ map to corresponding API services of the underlying FlexRay Driver(s), or FlexRay Transceiver Driver(s), respectively.

¹ E.g. timers, configuration data sets, etc.
24 of 134

In order to select those resources spread over the various entities² accessed via the [Frlf](#), the FlexRay-related AUTOSAR [BSW](#) modules use an indexing scheme that exemplarily described in

Figure 7-2 and Figure 7-3.

Frlf05052:

The [Frlf](#) achieves the abstraction (of the CCs and Drivers) by providing to the upper layer [BSW](#) modules an abstract, unique, zero-based consecutive index³ for each sort of resource, independent of their type, location, and access method.

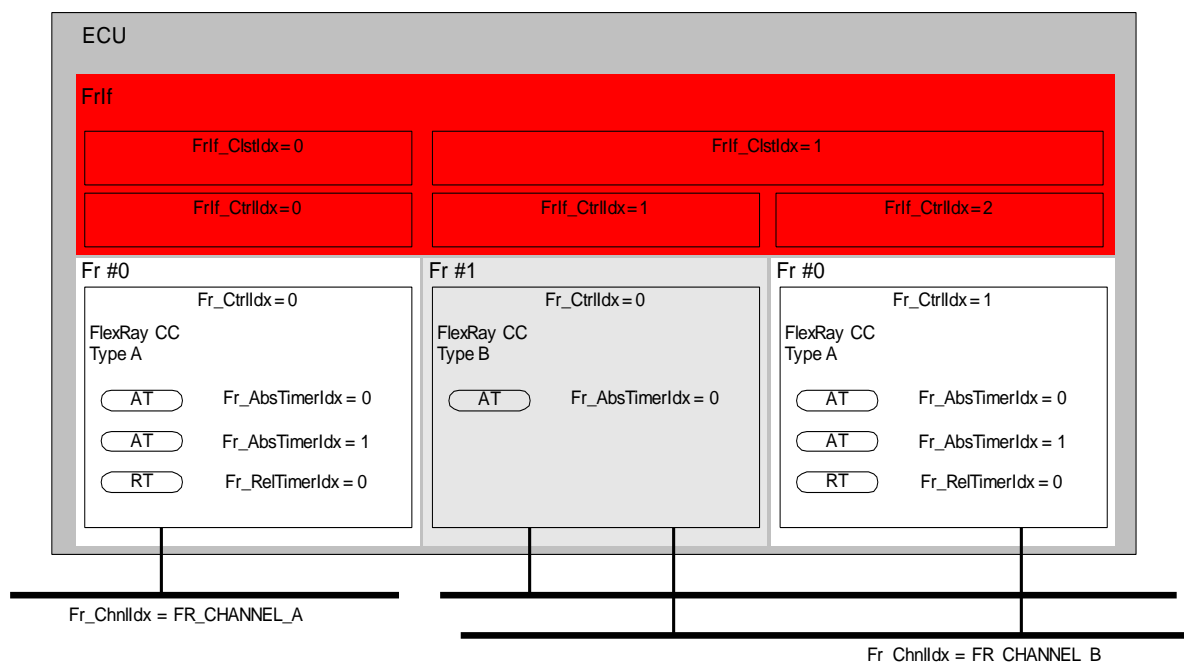
Frlf05103:

In general, the [Frlf](#) API service then uses the abstract index passed to it by the upper layer [BSW](#) module to retrieve:

1. **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 [Frlf](#)⁴, and
2. **the translated index used in the call to the lower layer BSW module's API service** from a static configuration data table.

Frlf05104:

The [Frlf](#) then calls the corresponding lower layer [BSW](#) module's API service via the function pointer and passes the translated index in the API call.



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

³ Like ControllerIndex, ClusterIndex, ChannelIndex, etc.

⁴ Since this table contains function pointers to the lower layer BSW modules' API services, it obviously has to be linked against the linked and located code of the lower layer BSW modules.

Figure 7-2: CC Indexing Scheme of the FlexRay Interface

Frlf05060:

In order to abstract for upper layer [BSW](#) modules the various CCs, which the [Frlf](#) controls via the FlexRay Drivers, the [Frlf](#) generates an abstract, unique, zero-based consecutive index `Frlf_CtrlIdx`, which maps to a tuple of FlexRay Driver API Service function pointer and CC index `Fr_CtrlIdx`.

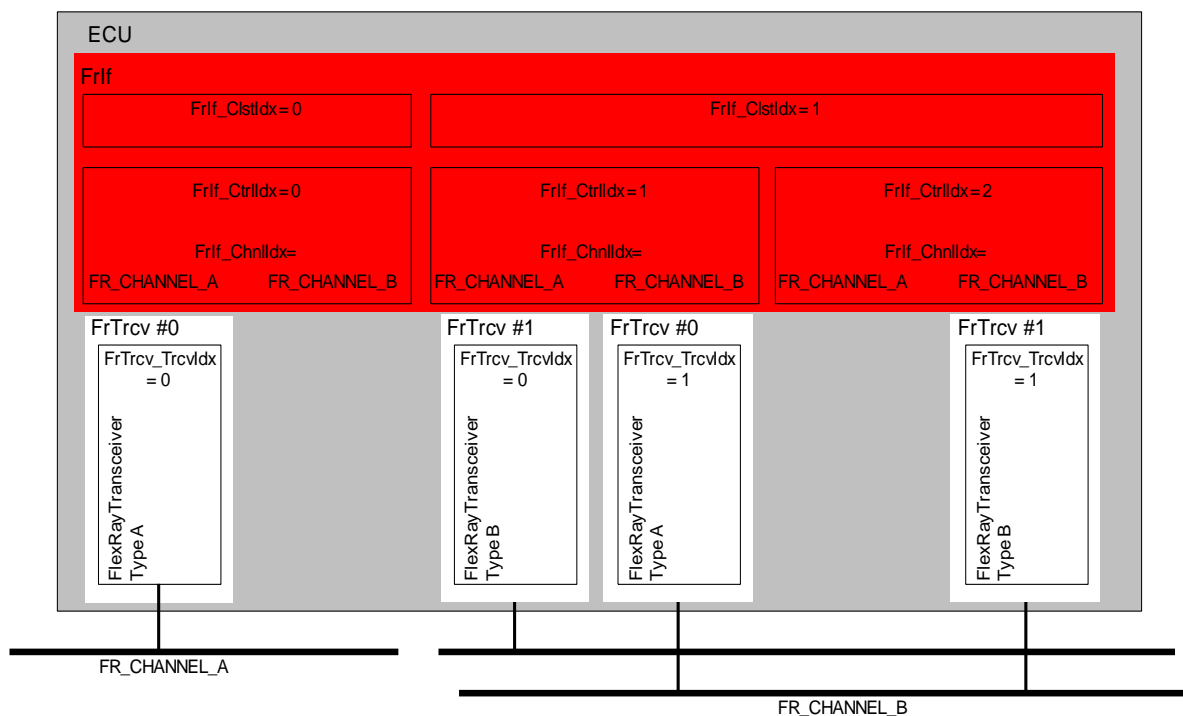


Figure 7-3: Flexray Transceiver Indexing Scheme of the FlexRay Interface

Frlf05105:

In order to abstract for upper layer [BSW](#) modules the various FlexRay Transceivers, which the [Frlf](#) accesses via the FlexRay Transceiver Drivers, the [Frlf](#) takes advantage of the fact that each FlexRay Transceiver is unambiguously assigned to a specific Channel on a specific FlexRay [CC](#).

Frlf05106:

Therefore, the [Frlf](#) abstracts the various FlexRay Transceivers by a **combination** of the two indices `Frlf_CtrlIdx` (Controller Index) and `Frlf_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)

Frlf05107:

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

Frlf05108:

The FlexRay Clusters are numbered with the index `Frlf_ClstIdx` in order to provide Cluster-based API services

Frlf05109:

The static configuration data of the [Frlf](#) contains a data structure that specifies which FlexRay [CC](#) and which FlexRay Transceivers are connected to which Clusters, or in other words, that maps each value of `Frlf_ClstIdx` to (one, or in general) a set of values for `Frlf_CtrlIdx` and tuples of (`Frlf_CtrlIdx`, `Frlf_ChnlIdx`).

Frlf05110:

The [Frlf](#) numbers all PDUs to be transmitted with an abstract, unique, zero-based consecutive index `Frlf_TxPduld`.

Note:

This index is used in the [Frlf](#) API service `Frlf_Transmit()` and allows the [Frlf](#) 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

Frlf05057:

The [Frlf](#) can be configured to support at least four (possibly different) **FlexRay Drivers** to access the FlexRay Communication Controllers.

Frlf05053:

The [Frlf](#) can be configured to support at least four (possibly different) **FlexRay CCs**.

Frlf05111:

The [Frlf](#) can be configured to support one of both or both **FlexRay Channels** as specified in [16].

Frlf05112:

The [Frlf](#) can be configured to support at least four **FlexRay Clusters**.

Frlf05113:

The [Frlf](#) can be configured to support at least one **absolute timer** per FlexRay [CCs](#).

Frlf05114:

The [Frlf](#) can be configured to support at least one **relative timer** per FlexRay [CCs](#).

7.3 FlexRay Interface State Machine

In order to allow to control the communication operations of the FlexRay system,

Frlf05115:

the [Frlf](#) implements a simple state machine (one per FlexRay cluster), called FlexRay Interface State Machine

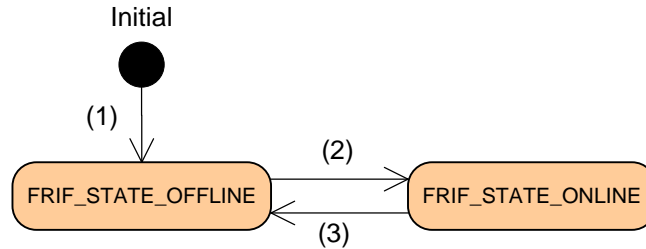


Figure 7-4: FlexRay Interface State Machine

Figure 7-4 shows the states and transitions that are visible to the user of the [Frlf](#). The two different states (Frlf_State) represent the communication capabilities of the Frlf.

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).

Frlf05116:

During initialization of the Frlf by executing Frlf_Init() (transition (1)) all CCs shall be offline (CC is brought to the POC-State 'Ready')

Frlf05117:

and the Frlf_State for each cluster is initialized with state 'FRIF_STATE_OFFLINE'.

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

The following table describes the transition names that can be passed to Frlf_SetState().

Transition Name	Transitions (see Figure 7-4)	Description
FRIF_GOTO_ONLINE	(2)	Transition resulting in Frlf_State FRIF_STATE_ONLINE

<i>Transition Name</i>	<i>Transitions (see Figure 7-4)</i>	<i>Description</i>
FRIF_GOTO_OFFLINE	(3)	Transition resulting in Frlf_State FRIF_STATE_OFFLINE

7.3.1 FlexRay Interface Main Function

Frlf05118:

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

Since the Cycle length of each Cluster is independent, and therefore the desired calling period of the FlexRay Interface Main Function might differ from Cluster to Cluster,

Frlf05119:

there shall be one dedicated FlexRay Interface Main Function for each FlexRay Cluster that is controlled by the FlexRay Interface.

The API names of the FlexRay Interface Main Functions therefore are:

- Frlf_MainFunction_0() for Cluster # 0 (Frlf_ClstIdx = 0)
- Frlf_MainFunction_1() for Cluster # 1 (Frlf_ClstIdx = 1)
- Frlf_MainFunction_2() for Cluster # 2 (Frlf_ClstIdx = 2)
- Frlf_MainFunction_3() for Cluster # 3 (Frlf_ClstIdx = 3)
- ... and so on, if more than 4 FlexRay Clusters are supported.

The Main Function monitors and controls the continuous execution of the FlexRay Job List Execution Function, including the (re)synchronization if a loss of the JobList's synchronization (see [JobListAsyncFlag](#)) or a miss of execution was detected.

This is done by the following steps:

Frlf05120:

The FlexRay Interface Main Function performs the following tasks at each invocation:

- Get the global time (Frlf_GetGlobalTime())

- add some 'safety margin'
- search the FlexRay Job List for the subsequent job, i.e. that job with an invocation time greater than the current global time + safety margin. If the end of the Job List has been reached, wrap around to the beginning of the Job List.
- 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)
- clear the JobListAsyncFlag
- Enable the absolute timer interrupt

7.4 Implementation Requirements

Frlf05068:

Link-time and post-build-time configuration data shall be implemented as read-only data structures. Link-time configuration data shall be immediately referenced by the implementation, the start-address of post-build-time configuration data shall be passed during module initialization

Frlf05069:

The Frlf module shall support pre-compile time, link-time and post-build-time configuration.

Frlf05078:

The Frlf module shall implement the API functions specified by the Fr SWS as real C code functions and shall not implement the API functions as macros.

The rationale is to allow object code module integration.

Frlf05079:

The description of the configuration and initialization data itself is not part of this specification but very implementation specific. The generated configuration data should be "human-readable".

Frlf05080:

The Frlf module's implementation shall conform to the HIS subset of the MISRA C Standard.

Frlf05085:

In case development error detection is enabled for the Frlf module: the Frlf module shall check API parameters for validity and report detected errors to the DET.

Frlf05087:

The Frlf module source code file(s) shall include *SchM_Frlf.h* if data consistency mechanisms of the BSW scheduler are required as described in [13].

Frlf05088:

The Frlf module header file shall include *MemMap.h* and apply the memory mapping abstraction mechanisms as specified by [15].

Frlf05090:

The header file *Frlf.h* shall contain a software and specification version number.

Frlf05092:

The Frlf shall support dynamic payload length for LPdus whose associated parameter *FrlfAllowDynamicLSduLength* is set to true.

Frlf05143:

None of the Frlf module's header files shall define global variables.

7.5 Configuration description

Frlf05089:

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.

7.6 Data Communication via FlexRay

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

Frlf05067:

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.

Frlf05054:

However, the resources (e.g. a buffer in the FlexRay Communication Controller or FlexRay Driver) needed for this data transmission (or reception, respectively) have to be defined [at system configuration time](#) specifically for this 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,

Frlf05055:

the API services that the [Frlf](#) provides to upper layer [BSW](#) modules for data transmission and data reception are PDU-based.

Note:

Since bus-independent AUTOSAR PDUs have a maximal length of 8 bytes, and since according to [16] a FlexRay Frame can contain as many as 254 bytes of payload data,

Frlf05121:

the [Frlf](#) shall be capable of packing multiple PDUs into one FlexRay Frame.

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!

Frlf05122:

The rules defining how to pack PDUs into FlexRay Frames, the so-called **Frame Construction Plans**, are defined [at system configuration time](#)

Frlf05123:

The Frame Construction Plan shall be stored in the static configuration of the [Frlf](#) (configuration parameter [FrlfFrameStructure](#)).

Frlf05124:

In cases where multiple PDUs are packed into a single FlexRay Frame, this FlexRay Frame has to be transmitted if at least one of the contained PDUs has been updated.

Note:

As a result, those 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 'FrlfUnusedBitValue' does not exist, arbitrary values for those PDUs will be transmitted.

Frlf05723:

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

Frlf05056:

In order for the receiving [Frlf](#) 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 `Frlf_Transmit()`) on the transmitter side, additional update information, so called **PDU Update-Bits** within the FlexRay Frame, shall be transmitted to the receiving [Frlf](#).

Frlf05125:

For each PDU, a dedicated PDU Update-Bit in the FlexRay Frame can be configured (see configuration parameter [FrlfPduUpdateBitOffset](#)).

Frlf05126:

This PDU Update-Bit can be located at an arbitrary bit position in the Frame Construction Plan that is not occupied by any PDU.

Frlf05127:

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 [FrlfPduUpdateBitOffset](#)]

Frlf05128:

If no update bit is configured for a specific PDU, this PDU is always assumed to be valid and its reception is always indicated to the upper layer BSW module on the receiver side.

Frlf05724:

On reception side, if the parameter 'FrlfUnusedBitValue' exists, after the FlexRay Driver has copied the L-SDU into the temporary buffer and before disassembling the

L-SDU, the remaining bits in the temporary buffer according to the Frame Construction Plan shall be set to the value given by 'FrlfUnusedBitValue'.

In case the parameter 'FrlfAllowDynamicLSduLength' 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).

Frlf05129:

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.

7.6.1.1 AlwaysTransmit

Note:

According to [16], 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 [Frlf](#) 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 [Frlf](#), called **AlwaysTransmit** (configuration parameter [FrlfAlwaysTransmit](#)). If [AlwaysTransmit](#) is enabled for an L-PDU that is transmitted using the Communication Operation [4], 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 [Frlf](#).

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 [Frlf](#) 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.

7.6.2 Realization of the Time-Driven FlexRay Schedule

According to [16], 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.

Frlf05130:

Therefore, the calling of all functions accessing these buffers (i.e. performing data transmission or reception, respectively) must be synchronous (i.e. synchronized to the FlexRay Global Time) in order to ensure buffer access taking place only at well-defined points in time⁵ and thus avoid concurrent access to the buffers by the hardware and the software.

Note:

In order to provide this necessary synchronicity, the [Frlf](#) defines for each Cluster a FlexRay Job List [Configuration Parameter [FrlfJobList](#)].

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

7.6.2.1 FlexRay Job List

Frlf05131:

A FlexRay Job List is a list of Communication Jobs sorted according to their respective execution start time.

Frlf05132:

Each Communication Job [Configuration Parameter [FrlfJob](#)] contains the following properties:

- Job start time by means of
 - FlexRay Communication Cycle [Configuration Parameter [FrlfCycle](#)]
 - Macrotick Offset within the Communication Cycle [Configuration Parameter [FrlfMacrotick](#)].
- A list of Communication Operations [Configuration Parameter [FrlfCommunicationOperation](#)] sorted according to a configurable Communication operation index [Configuration Parameter [FrlfCommunicationOperationIdx](#)]. The sorting order defines the order of execution of the Communication Operations within a FlexRay Communication Job.

Frlf05133:

The execution start time assigned to each Communication Job defines when the respective Cluster's FlexRay Job List Execution Function shall be called to execute this FlexRay Communication Job.

⁵ In FlexRay Global Time

Frlf05134:

The Communication Operations specify the actions to process within the Communication Job.

Frlf05135:

Each Communication Operation contains the following properties:

- Communication Operation Index [Configuration Parameter [FrlfCommunicationOperationIdx](#)], which determines the execution order of the Communication Operations.
- Communication Action [Configuration Parameter [FrlfCommunicationAction](#)], which specifies the actual action to perform (see 7.6.3):
 - DECOUPLED_TRANSMISSION
 - TX_CONFIRMATION
 - RECEIVE_AND_STORE
 - RX_INDICATION
 - RECEIVE_AND_INDICATE
 - PREPARE_LPDU
 - FREE_OP_A
 - FREE_OP_B
- A reference to a frame triggering (L-PDU) which is associated with the Communication Action to perform [Configuration parameter [FrlfLPduldx](#)]⁶.

Frlf05059:

7.6.2.2 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.

Frlf05136:

The API names of the FlexRay Job List Execution Functions therefore are:

- Frlf_JobListExec_0() for Cluster # 0 (Frlf_ClstIdx = 0)
- Frlf_JobListExec_1() for Cluster # 1 (Frlf_ClstIdx = 1)
- Frlf_JobListExec_2() for Cluster # 2 (Frlf_ClstIdx = 2)
- Frlf_JobListExec_3() for Cluster # 3 (Frlf_ClstIdx = 3)
- ... and so on, if more than 4 FlexRay Clusters are supported.

⁶ The 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.

Frlf05137:

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

Frlf05138:

Upon invocation, the FlexRay Job List Execution Function performs 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 [FrlfMaxlsrDelay](#)], the execution of the FlexRay Job List is considered to be asynchronous to the FlexRay Global Time and thus the following actions are performed:
 - Set a flag (JobListAsyncFlag) indicating that the execution of the FlexRay Job List of this Cluster is asynchronous
 - Call Dem_ReportErrorStatus(FRIF_E_JLE_SYNC, DEM_EVENT_STATUS_FAILED)
 - 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 sorted list of Communication Operations of the current Job pointed to by the current job-list-pointer.
4. Forward the current job-list pointer to the next job-list entry. If the job-list pointer was 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-list 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 in the correct order.

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.

Frlf05093:

In case the parameter 'FrlfAllowDynamicLSduLength' is set to true for the associated frame triggering, the actual used L-PDU length shall be passed to the driver (Fr_TransmitTxLPdu()) taking into account the following for each PDU:

- the position of the PDU within the L-PDU
- the actual PDU length passed via Frlf_Transmit()

- the position of the update-bit information (if configured)

Frlf05094:

In case the parameter 'FrlfAllowDynamicLSduLength' is set to true for the associated frame triggering for reception, PDUs in non received areas (PDU offset > actual L-PDU length) shall not be indicated to upper layer(s).

7.6.3 Communication Operations

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

Frlf05058:

7.6.3.1 TransmitWithDecoupledBufferAccess

Frlf05063:

DECOUPLED_TRANSMISSION is executed if the related CC is in Frlf_State FRIF_STATE_ONLINE. Otherwise, this Communication Operation is ignored.

For Communication Operation DECOUPLED_TRANSMISSION the following steps are performed:

1. Iterate over all PDUs contained in the [FrlfFrameStructure](#) of the associated frame triggering of this Communication Operation and
2. Check whether [TrigTxCounter](#) is > 0 or FrlfNoneMode == true for the PDU. If not, clear the update-bit for this PDU [Configuration Parameter [FrlfPduUpdateBitOffset](#)] and proceed with the next PDU.
3. Call the upper layer's function _TriggerTransmit() with the associated PDUId (Frlf_TxPdulId) and pass a pointer to a temporary buffer within the Frlf that assembles the L-SDU. The pointer shall consider the byte offset [Configuration Parameter [FrlfPduOffset](#)] of the PDU within the frame.
4. Decrement [TrigTxCounter](#) only if TrigTxCounter > 0.
5. Remember that a transmission for this PDU is pending if a transmission confirmation is needed for this PDU [Configuration Parameter [FrlfConfirm](#)] (increment TxConfCounter⁷).
6. Set the update-bit if configured for this PDU [Configuration Parameter [FrlfPduUpdateBitOffset](#)] only in case the API _TriggerTransmit() returned E_OK for the corresponding PDU.
7. If at least one PDU was requested for transmission or for at least one PDU FrlfNoneMode == true and _TriggerTransmit returned E_OK or the frame is configured to be always transmitted [Configuration Parameter FrlfAlwaysTransmit == true] then the FlexRay Driver's API service Fr_TransmitTxLPdu() is called:
 - a. Fr_CtrlIdx is derived according to the indexing scheme described in 7.2
 - b. Fr_LPdulIdx is set to the configured L-PDU index [Configuration Parameter [FrlfLPdulIdx](#)] associated with the Communication Operation

⁷ Limited by static configuration [Configuration Parameter [FrlfCounterLimit](#)]

- c. Fr_LSduPtr is set to the temporary Frlf L-SDU assembling buffer.
 - d. Fr_LSduLength is set to the L-SDU length [Configuration Parameter [FrlfLSduLength](#)]
8. 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.

All previously described actions are depicted in detail in the sequence chart in chapter 9.1.2.

7.6.3.2 ProvideTxConfirmation

Frlf05064:

TX_CONFIRMATION is executed if the related CC is in Frlf_State FRIF_STATE_ONLINE. Otherwise, this Communication Operation is ignored.

For Communication Operation TX_CONFIRMATION the following steps are performed:

1. Call the FlexRay Driver's API function Fr_CheckTxLPduStatus():
 - a. Fr_CtrlIdx is derived according to the indexing scheme described in 7.2
 - b. Fr_LPduldx is set to the configured L-PDU buffer index [Configuration Parameter [FrlfLPduldx](#)] associated with the Communication Operation.
2. If the transmission was performed (Output parameter *Fr_TxLPduStatusPtr is successfully set to FR_TRANSMITTED) then iterate over all PDUs contained in the [FrlfFrameStructure](#) of the associated frame triggering. If [TxConfCounter](#) for a PDU is 0 proceed with the next PDU, otherwise
 - a. Call the upper layer's function _TxConfirmation() with the associated PDUId (Frlf_TxPdulid).
 - b. Decrement [TxConfCounter](#).

7.6.3.3 ReceiveAndStore

RECEIVE_AND_STORE is executed if the related CC is in Frlf_State FRIF_STATE_ONLINE. Otherwise, this Communication Operation is ignored.

For Communication Operation RECEIVE_AND_STORE the following steps are performed:

1. Call the FlexRay Driver's API function Fr_ReceiveRxLPdu():
 - a. Fr_CtrlIdx is derived according to the indexing scheme described in 7.2
 - b. Fr_LPduldx is set to the configured L-PDU index [Configuration Parameter [FrlfLPduldx](#)] associated with the Communication Operation.
 - c. Fr_LSduPtr is set to a temporary buffer.

2. If a L-PDU was received (Output parameter *Fr_LPduStatusPtr is set to FR_RECEIVED) iterate over all PDUs contained in the [FrlfFrameStructure](#) of the associated frame triggering and:
 - a. If an update bit was configured for the PDU [Configuration Parameter [FrlfPduUpdateBitOffset](#)] 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 [FrlfPduOffset](#)] into a Frlf PDU-related static buffer.
 - c. Mark the PDU-related static buffer as up-to-date.

7.6.3.4 ProvideRxIndication

Frlf05062:

RX_INDICATION is executed if the related CC is in Frlf_State FRIF_STATE_ONLINE. Otherwise, this Communication Operation is ignored.

For Communication Operation RX_INDICATION the following steps are performed:

1. Iterate over all PDU-related static buffers of PDUs contained in the [FrlfFrameStructure](#) 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 _RxIndication() with the PDU Id the receiving module expects and a pointer to the PDU-related static buffer as parameters.
 - b. Mark the PDU-related static buffer as outdated.

7.6.3.5 ReceiveAndIndicate

RECEIVE_AND_INDICATE is executed if the related CC is in Frlf_State FRIF_STATE_ONLINE. Otherwise, this Communication Operation is ignored.

For Communication Operation RECEIVE_AND_INDICATE the following steps are performed:

1. Call the FlexRay Driver's API function Fr_ReceiveRxLPdu():
 - a. Fr_CtrlIdx is derived according to the indexing scheme described in 7.2
 - b. Fr_LPduIdx is set to the configured L-PDU index [Configuration Parameter [FrlfLPduIdx](#)] associated with the Communication Operation.
 - c. Fr_LSduPtr is set to a temporary buffer.
2. If an L-PDU was received (Output parameter *Fr_LPduStatusPtr is set to FR_RECEIVED) iterate over all PDUs contained in the [FrlfFrameStructure](#) of the associated frame triggering and:
 - a. If an update bit was configured for the PDU [Configuration Parameter [FrlfPduUpdateBitOffset](#)] and the update bit for the PDU is not set, continue with the next PDU. Otherwise,

- c. Call the upper layer's function _RxIndication() with the PDU Id the receiving module expects and a pointer to the temporary buffer considering the PDU offset within the L-SDU [Configuration Parameter [FrlfPduOffset](#)] as parameters.

7.6.3.6 PREPARE_LPDU

Frlf05061:

PREPARE_LPDU is executed in every Frlf_State.

For Communication Operation PREPARE_LPDU the following steps are performed:

1. Call the FlexRay Driver's API function Fr_PrepareLPdu():
 - a. Fr_CtrlIdx is derived according to the indexing scheme described in 7.2
 - b. Fr_LPduldx is set to the configured L-PDU index [Configuration Parameter [FrlfLPduldx](#)] associated with the Communication Operation.

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 FlexRay Driver does not use the opportunity for reconfiguration.

7.6.3.7 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.

7.6.3.8 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.

7.6.4 Transmission with Immediate Buffer Access

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.

For PDUs transmitted with immediate buffer access, 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.

If an immediate PDU transmission is indicated by calling `Frlf_Transmit()` with `Frlf_TxPduIdx` being configured for an immediate PDU, the following steps are performed within the context of the `Frlf_Transmit()` API service:

1. Call the FlexRay Driver's API function `Fr_TransmitTxLPdu()`:
 - a. `Fr_CtrlIdx` is derived according to the indexing scheme described in 7.2
 - b. `Fr_LPduIdx` is set to the configured L-PDU index [Configuration Parameter [FrlfLPduIdx](#)] associated with the `Frlf_TxPduIdx`.
 - c. `Fr_LSduPtr` is set to the Pdu Payload pointer contained in the `PduInfoPtr` passed as parameter to `Frlf_Transmit`.
 - d. `Fr_LSduLength` is set to the L-SDU length [Configuration Parameter [FrlfLSduLength](#)]
2. In case the Driver's API `Fr_TransmitTxLPdu()` returned `E_OK` (indicating that the transmission request succeeded) the [TxConfCounter](#) is incremented⁸ for the respective PDU.

7.7 Error Classification

Frlf05139:

Values for production code Event Ids are assigned externally by the configuration of the [DEM](#). They are published in the file `Dem_IntErrId.h` and included via `Dem.h`.

Frlf05142:

The error values and EventIds are named in capital letters according to the scheme `FRIF_E_<NAME>`, where `NAME` describes the error/EventId and may consist of several words separated by underscores.

Development error values are of type `uint8`.

⁸ Limited by static configuration [Configuration Parameter [FrlfCounterLimit](#)]

Type or error	Relevance	Related error code	Value [hex]
Invalid pointer	Development	FRIF_E_INV_POINTER	0x01
Invalid Controller index	Development	FRIF_E_INV_CTRL_IDX	0x02
Invalid Cluster index	Development	FRIF_E_INV_CLST_IDX	0x03
Invalid Channel index	Development	FRIF_E_INV_CHNL_IDX	0x04
Invalid timer index	Development	FRIF_E_INV_TIMER_IDX	0x05
Invalid Frlf_TxPdu Index	Development	FRIF_E_INV_TXPDUID	0x06
Frlf not initialized	Development	FRIF_E_NOT_INITIALIZED	0x08
Invalid Controller State	Development	FRIF_E_INV_FRIF_CC_STATE	0x09
Job List Execution lost synchronization to the FlexRay Global Time	Production	FRIF_E_JLE_SYNC	Assigned by DEM

Table 7-1: Definition of Error Codes

7.8 Error Detection

Frlf05084:

The detection of development errors is configurable (ON / OFF) at pre-compile time. The switch [FRIF_DEV_ERROR_DETECT](#) shall activate or deactivate the detection of all development errors.

If the [FRIF_DEV_ERROR_DETECT](#) switch is set to ON, API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.7 and chapter 8.

If the [FRIF_DEV_ERROR_DETECT](#) switch is set to ON, all [Frlf](#) API services other than `Frlf_Init()` and `Frlf_GetVersionInfo()` shall:

- not execute their normal operation,
- report to the DET module (using `FRIF_E_NOT_INITIALIZED`),
- and return `E_NOT_OK`,

unless the [Frlf](#) has been initialized with a preceding call of `Frlf_Init()`.

7.9 Error Notification

Detected development errors shall be reported to the `Det_ReportError()` API service of the [DET](#) if the pre-processor switch [FRIF_DEV_ERROR_DETECT](#) is set to ON.

Production errors shall be reported to the [DEM](#).

8 API Service Specification

Frlf05083:

All AP functions or global variables, whether they are specified or not shall follow the naming scheme `FrIf_<name>`, where the first letter of each word in `<name>` is written uppercase and the remainder of the word lowercase.

8.1 Imported types

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

Frlf05001:

<i>Module</i>	<i>Imported Type</i>
ComStack_Types	BusTrcvErrorType
	PdulType
	PdulInfoType
Dem	Dem_EventIdType
Fr	Fr_ChannelType
	Fr_MTSStatusType
	Fr_OffsetCorrectionType
	Fr_POCStatusType
	Fr_RateCorrectionType
	Fr_RxLPduStatusType
	Fr_SyncStateType
	Fr_TxLPduStatusType
FrTrcv	FrTrcv_TrvcModeType
	FrTrcv_TrvcWUReasonType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type Definitions

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

Frlf05082:

All types whether they are specified or implementation dependant shall follow the naming scheme `FrIf_<name>Type`, where the first letter of each word in `<name>` is written uppercase and the remainder of the word is written lowercase.

8.2.1 Frlf_ConfigType

Name:	FrIf_ConfigType	
Type:	Structure	
Range:	Implementation specific	--

Description:	This type contains the implementation-specific post build time configuration structure. Only pointers of this type are allowed.
---------------------	---

8.2.2 Frlf_StateType

Name:	FrIf_StateType	
Type:	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.	

8.2.3 Frlf_StateTransitionType

Name:	FrIf_StateTransitionType	
Type:	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.	

8.3 Function Definitions

This is a list of API services (functions) the [Frlf](#) provides to upper layer [BSW](#) modules.

8.3.1 Frlf_GetVersionInfo

Frlf05002:

Service name:	Frlf_GetVersionInfo
Syntax:	<pre>void FrIf_GetVersionInfo(Std_VersionInfoType* FrIf_VersionInfoPtr)</pre>
Service ID[hex]:	0x01
Sync/Async:	Synchronous
Reentrancy:	Non 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:	None
Description:	Returns the version information of this module.

Parameter Frlf_VersionInfoPtr: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_POINTER) if Frlf_VersionInfoPtr equals NULL_PTR.

This API service of the FlexRay Interface returns the version information of the FlexRay Interface. The version number consists of three parts:

- Two bytes for the vendor ID
- One byte for the module ID
- Three bytes version number.
 - The numbering shall be vendor specific; it shall consist of:
 - the major version number of the module,
 - the minor version number of the module,
 - and the patch version number of the module
- The AUTOSAR specification version number shall not be included.

This API service shall be pre compile time configurable ON/OFF by the configuration parameter FRIF_VERSION_INFO_API (derived from configuration parameter [FrlfVersionInfoApi](#)).

Hint:

If source code for caller and callee of this API service is available, this function should be realized as a macro. The macro should be defined in the file Frlf_Cfg.h.

Configuration:

If pre-compile-time configuration parameter 'FRIF_VERSION_INFO_API' is 'ON' this API function is included in the compilation process.

If pre-compile-time configuration parameter 'FRIF_VERSION_INFO_API' is 'OFF' this API function is excluded from the compilation process.

8.3.2 Frlf_Init

Frlf05003:

Service name:	Frlf_Init
Syntax:	void FrIf_Init(const FrIf_ConfigType* FrIf_ConfigPtr)
Service ID[hex]:	0x02
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	Frlf_ConfigPtr Base pointer to the configuration structure of the FlexRay Interface.
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Initializes the FlexRay Interface.

Parameter Frlf_ConfigPtr: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_POINTER) if Frlf_ConfigPtr equals NULL_PTR.

This API service of the FlexRay Interface is used to initialize the FlexRay Interface

This initialization is carried out by the following actions:

- Configure the FlexRay Interface: initialize the local memory space used to store the PDU data and the PDU properties and state variables and the FlexRay Interface State Machine.

The AUTOSAR ECU StateManager calls this FlexRay Interface API service with the address of the static configuration structure of the [Frlf](#) in parameter Frlf_ConfigPtr.

Caveats: Since during the execution of this API service, several hardware devices are accessed, the initialization order of [BSW](#) modules needs to ensure that those hardware devices can physically be accessed (e.g. the DIO Driver needs to be available, if applicable).

8.3.3 Frlf_ControllerInit

Frlf05004:

Service name:	Frlf_ControllerInit	
Syntax:	Std_ReturnType Frlf_ControllerInit(uint8 Frlf_CtrlIdx)	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of Frlf_CtrlIdx, reentrant for different values of Frlf_CtrlIdx	
Parameters (in):	Frlf_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.	

Parameter Frlf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if Frlf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Frlf_ControllerInit() by:

Translating (based on static [Frlf](#) configuration) the FlexRay [CC](#) index Frlf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific [CC](#) index Frlf_CtrlIdx).

Setting parameters Fr_LowLevelConfSetIdx and Fr_BufConfSetIdx to 0.

Calling Frlf_ControllerInit() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of Frlf_Init() before this API service may be called, see [\[Frlf05389\]](#).

8.3.4 Frlf_StartCommunication

Frlf05005:

Service name:	Frlf_StartCommunication	
Syntax:	Std_ReturnType Frlf_StartCommunication(uint8 Frlf_CtrlIdx)	
Service ID[hex]:	0x07	
Sync/Async:	Asynchronous	
Reentrancy:	non reentrant for identical values of Frlf_CtrlIdx, reentrant for different values of Frlf_CtrlIdx	
Parameters (in):	Frlf_CtrlIdx	Index of the FlexRay CC to address.
Parameters	None	

(inout):	
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().

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_StartCommunication()` by:

Translating (based on static `FrIf` configuration) the FlexRay CC index `FrIf_CtrlIdx` into a tuple (FlexRay Driver | Driver-specific CC index `Fr_CtrlIdx`).

Calling `Fr_StartCommunication()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[FrIf05389\]](#).

8.3.5 FrIf_HaltCommunication

FrIf05006:

Service name:	<code>FrIf_HaltCommunication</code>
Syntax:	<code>Std_ReturnType FrIf_HaltCommunication(uint8 FrIf_CtrlIdx)</code>
Service ID[hex]:	0x09
Sync/Async:	Asynchronous
Reentrancy:	non reentrant for identical values of <code>FrIf_CtrlIdx</code> , reentrant for different values of <code>FrIf_CtrlIdx</code>
Parameters (in):	<code>FrIf_CtrlIdx</code> 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 <code>Fr_HaltCommunication()</code> .

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function

Fr_HaltCommunication() by:

Translating (based on static FrIf 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 with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[Frlf05389](#)].

8.3.6 Frlf_AbortCommunication

Frlf05007:

Service name:	Frlf_AbortCommunication	
Syntax:	Std_ReturnType FrIf_AbortCommunication(uint8 FrIf_CtrlIdx)	
Service ID[hex]:	0x0b	
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().	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_AbortCommunication() by:

1. Translating (based on static FrIf 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 with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[Frlf05389](#)].

8.3.7 Frlf_GetState

Service name:	FrIf_GetState	
Syntax:	Std_ReturnType FrIf_GetState(uint8 FrIf_ClstIdx, FrIf_StateType* FrIf_StatePtr)	
Service ID[hex]:	0x04	
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 FrIf_State 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.	

Parameter FrIf_ClstIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CLST_IDX) if FrIf_CtrlIdx has an invalid value.

Parameter FrIf_StatePtr: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_POINTER) if FrIf_StatePtr equals NULL_PTR.

This API service of the FlexRay Interface retrieves the FrIf_State of the FlexRay Cluster with index FrIf_ClstIdx.

2. Return the current state of the FrIf State Machine:
 - a. FRIF_STATE_ONLINE
 - b. FRIF_STATE_OFFLINE

The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[FrIf05389\]](#).

8.3.8 FrIf_SetState

Service name:	FrIf_SetState	
Syntax:	Std_ReturnType FrIf_SetState(FrIf_StateTransitionType FrIf_StateTransition, uint8 FrIf_ClstIdx)	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrIf_StateTransition	Requested FrIf state transition.
	FrIf_ClstIdx	Index of the cluster addressed.
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.	

Parameter `FrIf_StateTransition`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_FRIF_CC_STATE`) if `FrIf_StateTransition` has an invalid value.

Parameter `FrIf_ClstIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface requests for FlexRay Cluster with index `FrIf_ClstIdx` the `FrIf_State` reached by the state transition according to `FrIf_StateTransition`.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.3.9 FrIf_SetWakeupChannel

FrIf05010:

Service name:	FrIf_SetWakeupChannel	
Syntax:	Std_ReturnType FrIf_SetWakeupChannel(uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx)	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of <code>FrIf_CtrlIdx</code> , reentrant for different values of <code>FrIf_CtrlIdx</code>	
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):	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 <code>Fr_SetWakeupChannel()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_SetWakeupChannel()` by:

Translating (based on static `FrIf` configuration) the FlexRay CC index `FrIf_CtrlIdx` into a tuple (FlexRay Driver | Driver-specific CC index `Fr_CtrlIdx`).

3. Setting parameters

- `Fr_ChnlIdx` to `FrIf_ChnlIdx`

4. Calling `Fr_SetWakeupChannel()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.3.10 `FrIf_SendWUP`

`FrIf05011`:

Service name:	<code>FrIf_SendWUP</code>	
Syntax:	<code>Std_ReturnType FrIf_SendWUP(</code> <code> uint8 FrIf_CtrlIdx</code> <code>)</code>	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of <code>FrIf_CtrlIdx</code> , reentrant for different values of <code>FrIf_CtrlIdx</code>	
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_SendWUP()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_SendWUP()` by:

Translating (based on static `FrIf` configuration) the FlexRay CC index `FrIf_CtrlIdx` into a tuple (FlexRay Driver | Driver-specific CC index `Fr_CtrlIdx`).

5. Calling `Fr_SendWUP()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.3.11 FrIf_GetSyncState

FrIf05012:

Service name:	FrIf_GetSyncState	
Syntax:	Std_ReturnType FrIf_GetSyncState(uint8 FrIf_CtrlIdx, Fr_SyncStateType* FrIf_SyncStatePtr)	
Service ID[hex]:	0x16	
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_SyncStatePtr	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_GetSyncState()	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

Parameter FrIf_SyncStatePtr: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_POINTER) if FrIf_SyncStatePtr has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_GetSyncState() by:

Translating (based on static FrIf configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

6. Setting parameters

- Fr_SyncStatePtr to FrIf_SyncStatePtr

7. Calling Fr_GetSyncState() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[FrIf05389\]](#).

8.3.12 FrIf_SetExtSync

Frlf05013:

Service name:	Frlf_SetExtSync	
Syntax:	<pre>Std_ReturnType FrIf_SetExtSync(uint8 FrIf_CtrlIdx, Fr_OffsetCorrectionType FrIf_Offset, Fr_RateCorrectionType FrIf_Rate)</pre>	
Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of Frlf_CtrlIdx, reentrant for different values of Frlf_CtrlIdx	
Parameters (in):	Frlf_CtrlIdx	Index of the FlexRay CC to address.
	Frlf_Offset	Offset correction that shall be applied
	Frlf_Rate	Rate correction that shall be applied
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_SetExtSync().	

Parameter Frlf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if Frlf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_SetExtSync() by:

Translating (based on static Frlf configuration) the FlexRay CC index Frlf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

8. Setting parameters

- Fr_Rate to Frlf_Rate
- Fr_Offset to Frlf_Offset
- Calling Fr_SetExtSync() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of Frlf_Init() before this API service may be called, see [[Frlf05389](#)].

8.3.13 FrIf_GetPOCStatus

Frlf05014:

Service name:	Frlf_GetPOCStatus	
Syntax:	Std_ReturnType FrIf_GetPOCStatus (uint8 FrIf_CtrlIdx, Fr_POCStatusType* FrIf_POCStatusPtr)	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of Frlf_CtrlIdx, reentrant for different values of Frlf_CtrlIdx	
Parameters (in):	Frlf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout):	None	
Parameters (out):	Frlf_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().	

Parameter Frlf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if Frlf_CtrlIdx has an invalid value.

Parameter Frlf_POCStatusPtr: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_POINTER) if Frlf_POCStatusPtr has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_GetPOCStatus() by:

Translating (based on static Frlf configuration) the FlexRay CC index Frlf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

9. Setting parameters

- Fr_POCStatusPtr to Frlf_POCStatusPtr

10. Calling Fr_GetPOCStatus() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of Frlf_Init() before this API service may be called, see [[Frlf05389](#)].

8.3.14 Frlf_GetGlobalTime

Frlf05015:

Service name:	Frlf_GetGlobalTime	
Syntax:	Std_ReturnType FrIf_GetGlobalTime (uint8 FrIf_CtrlIdx,)	

	uint8* FrIf_CyclePtr, uint16* FrIf_MacroTickPtr)	
Service ID[hex]:	0x1a	
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().	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_GetGlobalTime() by:

Translating (based on static FrIf configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

11. Setting parameters

- Fr_CylcePtr to FrIf_CyclePtr
- Fr_MacroTickPtr to FrIf_MacroTickPtr

12. Calling Fr_GetGlobalTime() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[FrIf05389](#)].

8.3.15 FrIf_AllowColdstart

FrIf05017:

Service name:	FrIf_AllowColdstart	
Syntax:	Std_ReturnType FrIf_AllowColdstart(uint8 FrIf_CtrlIdx)	
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().	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_AllowColdstart() by:

Translating (based on static FrIf configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

13. Calling Fr__AllowColdstart() of the determined FlexRay Driver

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[FrIf05389\]](#).

8.3.16 FrIf_GetMacroticksDuration

FrIf0501x:

Service name:	FrIf_GetMacroticksDuration	
Syntax:	uint16 FrIf_GetMacroticksDuration(uint8 FrIf_CtrlIdx)	
Service ID[hex]:	0x1b	
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 nanoseconds of one Macrotick
Description:	Retrieves the duration of one Macrotick in nanoseconds	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

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

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[FrIf05389\]](#).

8.3.17 FrIf_Transmit

FrIf05033:

Service name:	FrIf_Transmit	
Syntax:	<pre>Std_ReturnType FrIf_Transmit(PduIdType FrIf_TxPduId, const PduInfoType* FrIf_PduInfoPtr)</pre>	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of FrIf_TxPduId, reentrant for different values of FrIf_TxPduId	
Parameters (in):	FrIf_TxPduId	ID of FlexRay PDU to be transmitted.
	FrIf_PduInfoPtr	Pointer to a structure with FlexRay PDU related data.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error has occurred during the execution of this API service. E_NOT_OK: An error occurred during execution of this API service: <ul style="list-style-type: none"> • FlexRay Driver reported an error in case of immediate transmission • An error has been detected in development mode
Description:	Requests the sending of a PDU.	

Parameter FrIf_TxPduId: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_TXPDUID) if FrIf_TxPduId has an invalid value.

Parameter FrIf_PduInfoPtr: FRIF_E_INV_POINTER shall be reported to DET in case FrIf_PduInfoPtr equals NULL_PTR

If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_POINTER) if SduDataPtr in FrIf_PduInfoPtr equals NULL_PTR.

This API service of the FlexRay Interface allows upper layer [BSW](#) modules to request the sending of a PDU via the FlexRay Communication System.

In case of decoupled transmission the PDU with index `FrIf_TxPduld` is **not yet** passed to the underlying FlexRay Driver for transmission. `FrIf` only remembers the PDU's transmission request (increment `TrigTxCounter`⁹). This decoupling mechanism between the call of `FrIf_Transmit()` and the execution of the [FrlfCommunicationAction](#) has some implications:

- The upper layer BSW module may operate asynchronously to the FlexRay Communication System and thus may call `FrIf_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 `<UL_TriggerTransmit>()` API service at (from the [BSW](#)'s point of view) any arbitrary point in time.

In case of immediate transmission the PDU (single PDU, no Update bit) is passed to the underlying FlexRay Driver immediately for transmission.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[Frlf05389](#)].

8.3.18 Frlf_SetTransceiverMode

Frlf05034:

Service name:	Frlf_SetTransceiverMode	
Syntax:	<pre>Std_ReturnType FrIf_SetTransceiverMode(uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcevModeType FrIf_TrcevMode)</pre>	
Service ID[hex]:	0x28	
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_TrcevMode</code>	Transceiver mode to be set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	<code>Std_ReturnType</code>	<code>E_OK</code> : The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_NO_ERROR</code> . <code>E_NOT_OK</code> : The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_ERROR</code> .
Description:	Wraps the FlexRay Transceiver Driver API function <code>FrTrcv_SetTransceiverMode()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

⁹ Limited by static configuration [Configuration Parameter [FrlfCounterLimit](#)]

Parameter `FrIf_ChnlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CHNL_IDX`) if `FrIf_ChnlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function `FrTrcv_SetTransceiverMode()` by:

1. Translating (based on static [FrIf](#) 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_TrvcMode` to `FrIf_TrvcMode`
3. Calling `FrTrcv_SetTransceiverMode()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[FrIf05389\]](#).

8.3.19 `FrIf_GetTransceiverMode`

FrIf05035:

Service name:	<code>FrIf_GetTransceiverMode</code>	
Syntax:	<pre>Std_ReturnType FrIf_GetTransceiverMode(uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrvcModeType* FrIf_TrvcModePtr)</pre>	
Service ID[hex]:	0x2a	
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_TrvcModePtr</code>	Pointer to a memory location where output value will be stored.
Return value:	<code>Std_ReturnType</code>	E_OK: The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_NO_ERROR</code> . E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_ERROR</code> .
Description:	Wraps the FlexRay Transceiver Driver API function <code>FrTrcv_GetTransceiverMode()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

Parameter `FrIf_ChnlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CHNL_IDX`) if `FrIf_ChnlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function `FrTrcv_GetTransceiverMode()` by:

1. Translating (based on static [FrIf](#) 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 parameters
 - `FrTrcv_TrcvModePtr` to `FrIf_TrcvModePtr`
3. Calling `FrTrcv_GetTransceiverMode()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[FrIf05389\]](#).

8.3.20 `FrIf_GetTransceiverWUReason`

FrIf05036:

Service name:	<code>FrIf_GetTransceiverWUReason</code>	
Syntax:	<pre>Std_ReturnType FrIf_GetTransceiverWUReason (uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcvWUReasonType* FrIf_TrcvWUReasonPtr)</pre>	
Service ID[hex]:	0x2b	
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_TrcvWUReasonPtr</code>	Pointer to a memory location where output value will be stored.
Return value:	<code>Std_ReturnType</code>	E_OK: The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_NO_ERROR</code> . E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_ERROR</code> .
Description:	Wraps the FlexRay Transceiver Driver API function <code>FrTrcv_GetTransceiverWUReason()</code>	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

Parameter `FrIf_ChnlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CHNL_IDX`) if `FrIf_ChnlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function `FrTrcv_GetTransceiverWUReason()` by:

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

Caveats: The FlexRay Interface has to be initialized with a call of `Frlf_Init()` before this API service may be called, see [\[Frlf05389\]](#).

8.3.21 `Frlf_EnableTransceiverWakeup`

Frlf05037:

Service name:	<code>Frlf_EnableTransceiverWakeup</code>	
Syntax:	<pre>Std_ReturnType Frlf_EnableTransceiverWakeup(uint8 Frlf_CtrlIdx, Fr_ChannelType Frlf_ChnlIdx)</pre>	
Service ID[hex]:	0x2c	
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:	<code>Std_ReturnType</code>	E_OK: The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_NO_ERROR</code> . E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned <code>BUSTRCV_E_ERROR</code> .
Description:	Wraps the FlexRay Transceiver Driver API function <code>FrTrcv_EnableTransceiverWakeup()</code> .	

Parameter `Frlf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `Frlf_CtrlIdx` has an invalid value.

Parameter `Frlf_ChnlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CHNL_IDX`) if `Frlf_ChnlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function `FrTrcv_EnableTransceiverWakeup()` by:

Translating (based on static [Frlf](#) configuration) the tuple (FlexRay [CC](#) index `Frlf_CtrlIdx` | FlexRay Channel index `Frlf_ChnlIdx`) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index `FrTrcv_TrcvIdx`).

Calling FrTrcv_EnableTransceiverWakeup() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[Frlf05389](#)].

8.3.22 Frlf_DisableTransceiverWakeup

Frlf05038:

Service name:	Frlf_DisableTransceiverWakeup	
Syntax:	Std_ReturnType Frlf_DisableTransceiverWakeup (uint8 Frlf_CtrlIdx, Fr_ChannelType Frlf_ChnlIdx)	
Service ID[hex]:	0x2e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Frlf_CtrlIdx	Index of the FlexRay CC to address.
	Frlf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller Frlf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_DisableTransceiverWakeup().	

Parameter Frlf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if Frlf_CtrlIdx has an invalid value.

Parameter Frlf_ChnlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CHNL_IDX) if Frlf_ChnlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function FrTrcv_DisableTransceiverWakeup() by:

Translating (based on static [Frlf](#) configuration) the tuple (FlexRay [CC](#) index Frlf_CtrlIdx | FlexRay Channel index Frlf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_TrcvIdx).

Calling FrTrcv_DisableTransceiverWakeup() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[Frlf05389](#)].

8.3.23 FrIf_ClearTransceiverWakeup

FrIf05039:

Service name:	FrIf_ClearTransceiverWakeup	
Syntax:	Std_ReturnType FrIf_ClearTransceiverWakeup(uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx)	
Service ID[hex]:	0x30	
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:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverWakeup().	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

Parameter FrIf_ChnlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CHNL_IDX) if FrIf_ChnlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverWakeup() by:

Translating (based on static [FrIf](#) 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).

Calling FrTrcv_EnableTransceiverWakeup() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[FrIf05389\]](#).

8.3.24 FrIf_GetCycleLength

Service name:	FrIf_GetCycleLength	
Syntax:	uint32 FrIf_GetCycleLength(uint8 FrIf_CtrlIdx)	
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 FrIf_CtrlIdx.	

FrIf05237: If parameter FrIf_CtrlIdx of FrIf_GetCycleLength has an invalid value and if development error detection is enabled (i.e. FRIF_DEV_ERROR_DETECT 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.

FrIf05238: 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 FrIf05002.

8.4 Optional Function Definitions

8.4.1 FrIf_SetAbsoluteTimer

FrIf05021:

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]:	0x1e	
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().	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_SetAbsoluteTimer() by:

Translating (based on static FrIf 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_Cycle to FrIf_Cycle
- Fr_Offset to FrIf_Offset

3. Calling Fr_SetAbsoluteTimer() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[FrIf05389](#)].

8.4.2 Frlf_SetRelativeTimer

Frlf05022:

Service name:	Frlf_SetRelativeTimer	
Syntax:	<pre>Std_ReturnType FrIf_SetRelativeTimer (uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx, uint16 FrIf_Offset)</pre>	
Service ID[hex]:	0x1f	
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_RelTimerIdx	Index of the relative timer to address.
	Frlf_Offset	Number of Macroticks the relative timer shall be set to.
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_SetRelativeTimer().	

Parameter Frlf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if Frlf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_SetRelativeTimer() by:

Translating (based on static Frlf configuration) the FlexRay CC index Frlf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

Setting parameters

- Fr_RelTimerIdx to Frlf_RelTimerIdx
- Fr_Offset to Frlf_Offset
- Calling Fr_SetRelativeTimer() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of Frlf_Init() before this API service may be called, see [\[Frlf05389\]](#).

8.4.3 Frlf_CancelAbsoluteTimer

Frlf05023:

Service name:	Frlf_CancelAbsoluteTimer	
Syntax:	<pre>Std_ReturnType FrIf_CancelAbsoluteTimer (uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx)</pre>	
Service ID[hex]:	0x20	

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() .	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_CancelAbsoluteTimer() by:

Translating (based on static FrIf configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx).

- Setting parameters
- Fr_AbsTimerIdx to FrIf_AbsTimerIdx
- Calling Fr_AbsoluteRelativeTimer() of the determined FlexRay Driver with the parameters determined as described above.

Calling Fr_AbsoluteRelativeTimer() of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[FrIf05389](#)].

8.4.4 FrIf_CancelRelativeTimer

FrIf05024:

Service name:	FrIf_CancelRelativeTimer	
Syntax:	Std_ReturnType FrIf_CancelRelativeTimer(uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx)	
Service ID[hex]:	0x21	
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_RelTimerIdx	Index of the relative 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().	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.
`FrIf_EnableAbsoluteTimerIRQ`

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_CancelRelativeTimer()` by:

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

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[FrIf05389\]](#).

8.4.5 `FrIf_EnableAbsoluteTimerIRQ`

`FrIf05025`:

Service name:	<code>FrIf_EnableAbsoluteTimerIRQ</code>	
Syntax:	<pre>Std_ReturnType FrIf_EnableAbsoluteTimerIRQ(uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx)</pre>	
Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>FrIf_AbsTimerIdx</code>	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 <code>Fr_EnableAbsoluteTimerIRQ()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_EnableAbsoluteTimerIRQ()` by:

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

Caveats: The FlexRay Interface has to be initialized with a call of `Frlf_Init()` before this API service may be called, see [\[Frlf05389\]](#).

8.4.6 `Frlf_EnableRelativeTimerIRQ`

`Frlf05026`:

Service name:	<code>FrIf_EnableRelativeTimerIRQ</code>	
Syntax:	<pre>Std_ReturnType FrIf_EnableRelativeTimerIRQ(uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx)</pre>	
Service ID[hex]:	0x23	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	<code>Frlf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>Frlf_RelTimerIdx</code>	Index of the relative timer 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_EnableRelativeTimerIRQ()</code> .	

Parameter `Frlf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `Frlf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_EnableRelativeTimerIRQ()` by:

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

Caveats: The FlexRay Interface has to be initialized with a call of `Frlf_Init()` before this API service may be called, see [[Frlf05389](#)].

8.4.7 `Frlf_GetAbsoluteTimerIRQStatus`

Frlf05027:

Service name:	<code>Frlf_GetAbsoluteTimerIRQStatus</code>	
Syntax:	<pre>Std_ReturnType Frlf_GetAbsoluteTimerIRQStatus (uint8 Frlf_CtrlIdx, uint8 Frlf_AbsTimerIdx, boolean* Frlf_IRQStatusPtr)</pre>	
Service ID[hex]:	0x39	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	<code>Frlf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>Frlf_AbsTimerIdx</code>	Index of the absolute timer to address.
Parameters (inout):	None	
Parameters (out):	<code>Frlf_IRQStatusPtr</code>	Pointer to a memory location where output value will be stored.
Return value:	<code>Std_ReturnType</code>	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 <code>Fr_GetAbsoluteTimerIRQStatus()</code>	

Parameter `Frlf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `Frlf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_GetAbsoluteTimerIRQStatus()` by:

1. Translating (based on static `FrIf` 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 with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.4.8 `FrIf_GetRelativeTimerIRQStatus`

FrIf05028:

Service name:	<code>FrIf_GetRelativeTimerIRQStatus</code>	
Syntax:	<pre>Std_ReturnType FrIf_GetRelativeTimerIRQStatus (uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx, boolean* FrIf_IRQStatusPtr)</pre>	
Service ID[hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>FrIf_RelTimerIdx</code>	Index of the relative timer to address.
Parameters (inout):	None	
Parameters (out):	<code>FrIf_IRQStatusPtr</code>	Pointer to a memory location where output value will be stored.
Return value:	<code>Std_ReturnType</code>	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 <code>Fr_GetRelativeTimerIRQStatus()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_GetRelativeTimerIRQStatus()` by:

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

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.4.9 `FrIf_AckAbsoluteTimerIRQ`

FrIf05029:

Service name:	<code>FrIf_AckAbsoluteTimerIRQ</code>	
Syntax:	<pre>Std_ReturnType FrIf_AckAbsoluteTimerIRQ(uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx)</pre>	
Service ID[hex]:	0x24	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>FrIf_AbsTimerIdx</code>	Index of the absolute timer to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	<code>Std_ReturnType</code>	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 <code>Fr_AckAbsoluteTimerIRQ()</code>	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_AckAbsoluteTimerIRQ()` by:

1. Translating (based on static `FrIf` 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_AckAbsoluteTimerIRQ()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.4.10 `FrIf_AckRelativeTimerIRQ`

`FrIf05030`:

Service name:	<code>FrIf_AckRelativeTimerIRQ</code>	
Syntax:	<pre>Std_ReturnType FrIf_AckRelativeTimerIRQ(uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx)</pre>	
Service ID[hex]:	0x25	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	<code>FrIf_CtrlIdx</code>	Index of the FlexRay CC to address.
	<code>FrIf_RelTimerIdx</code>	Index of the relative timer 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_AckRelativeTimerIRQ()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_AckRelativeTimerIRQ()` by:

1. Translating (based on static `FrIf` 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_AckRelativeTimerIRQ()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.4.11 FrIf_DisableAbsoluteTimerIRQ

FrIf05031:

Service name:	FrIf_DisableAbsoluteTimerIRQ	
Syntax:	Std_ReturnType FrIf_DisableAbsoluteTimerIRQ(uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx)	
Service ID[hex]:	0x26	
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 <code>Fr_DisableAbsoluteTimerIRQ()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function `Fr_DisableAbsoluteTimerIRQ()` by:

1. Translating (based on static `FrIf` 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_DisableAbsoluteTimerIRQ()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [[FrIf05389](#)].

8.4.12 FrIf_DisableRelativeTimerIRQ

FrIf05032:

Service name:	FrIf_DisableRelativeTimerIRQ	
Syntax:	Std_ReturnType FrIf_DisableRelativeTimerIRQ(uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx)	

Service ID[hex]:	0x27	
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_RelTimerIdx	Index of the relative 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_DisableRelativeTimerIRQ().	

Parameter FrIf_CtrlIdx: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using FRIF_E_INV_CTRL_IDX) if FrIf_CtrlIdx has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_DisableRelativeTimerIRQ() by:

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

Caveats: The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [[FrIf05389](#)].

8.4.13 FrIf_GetNmVector

FrIf05016:

Service name:	FrIf_GetNmVector	
Syntax:	Std_ReturnType FrIf_GetNmVector(uint8 FrIf_CtrlIdx, uint8* FrIf_NmVectorPtr)	
Service ID[hex]:	0x34	
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_NmVectorPtr	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:	Derives the FlexRay NM Vector.
---------------------	--------------------------------

Parameter `Frlf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `Frlf_CtrlIdx` has an invalid value.

This API service of the FlexRay Interface derives the FlexRay NM Vector by:

Translating (based on static `Frlf` configuration) the FlexRay CC index `Frlf_CtrlIdx` into a tuple (FlexRay Driver | Driver-specific CC index `Fr_CtrlIdx`).

14. Setting parameters

- `Fr_NmVectorPtr` to `Frlf_NmVectorPtr`

Calling `Fr_GetNmVector()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `Frlf_Init()` before this API service may be called, see [\[Frlf05389\]](#).

8.4.14 `Frlf_GetClockCorrection`

Frlf05071:

Service name:	<code>Frlf_GetClockCorrection</code>	
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):	<code>Frlf_CtrlIdx</code>	Index of FlexRay CC within the context of the FlexRay Interface.
Parameters (inout):	None	
Parameters (out):	<code>Frlf_RateCorrectionPtr</code>	Address where the current rate correction value shall be stored.
	<code>Frlf_OffsetCorrectionPtr</code>	Address where the current offset correction value 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_GetClockCorrection ()</code> and gets the current clock correction values.	

Frlf05414: The function `Frlf_GetClockCorrection` shall be pre compile time configurable ON/OFF by the configuration parameter `FrlfGetClockCorrectionSupport` (derived from configuration parameter `FrlfGetClockCorrectionSupport`, see [Frlf06106_Conf](#))

Frlf05711: If development error detection for the `Frlf` module is enabled: if the function `Frlf_GetClockCorrection` is called before the `Frlf` was initialized successfully, the function `Frlf_GetClockCorrection` shall raise the development error `FRIF_E_NOT_INITIALIZED` and return `E_NOT_OK`.

Frlf05712: If development error detection for the Frlf module is enabled: the function `Frlf_GetClockCorrection` shall check the parameter `Frlf_CtrlIdx` for being valid. If `Frlf_CtrlIdx` is invalid, the function `Frlf_GetClockCorrection` shall raise the development error `FRIF_E_INV_CTRL_IDX` and return `E_NOT_OK`.

8.4.15 Frlf_GetChannelStatus

Frlf05030:

Service name:	Frlf_GetChannelStatus	
Syntax:	<pre>Std_ReturnType Frlf_GetChannelStatus (uint8 Frlf_CtrlIdx, uint16* Frlf_ChannelAStatusPtr, uint16* Frlf_ChannelBStatusPtr)</pre>	
Service ID[hex]:	0x3c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Frlf_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Interface.
Parameters (inout):	None	
Parameters (out):	Frlf_ChannelAStatusPtr	Address where the bitcoded channel A status information shall be stored.
	Frlf_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 <code>Fr_GetChannelStatus()</code> and gets the channel status information.	

Frlf05413: The function `Frlf_GetChannelStatus` shall be pre compile time configurable ON/OFF by the configuration parameter `FrlfGetChannelStatusSupport` (derived from configuration parameter `FrlfGetChannelStatusSupport`, see `Frlf06105_Conf`)

Frlf05708: If development error detection for the Frlf module is enabled: if the function `Frlf_GetChannelStatus` is called before the Frlf module was initialized successfully, the function `Frlf_GetChannelStatus` shall raise the development error `FRIF_E_NOT_INITIALIZED` and return `E_NOT_OK`.

Frlf05709: If development error detection for the Frlf module is enabled: the function `Frlf_GetChannelStatus` shall check the parameter `Frlf_CtrlIdx` for being valid. If `Frlf_CtrlIdx` is invalid, the function `Frlf_GetChannelStatus` shall raise the development error `FRIF_E_INV_CTRL_IDX` and return `E_NOT_OK`.

8.4.16 Frlf_ReadCCConfig

Frlf05313:

Service name:	FrIf_ReadCCConfig	
Syntax:	<pre>Std_ReturnType FrIf_ReadCCConfig(uint8 FrIf_CtrlIdx, uint8 FrIf_ConfigParamIdx, uint32* FrIf_ConfigParamValuePtr)</pre>	
Service ID[hex]:	0x3d	
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	<p>E_OK: The call of the FlexRay Driver's API service has returned E_OK.</p> <p>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.</p>
Description:	Wraps the FlexRay Driver API function Fr_ReadCCConfig().	

FrIf05314: The function FrIf_ReadCCConfig wraps the FlexRay Driver API Fr_ReadCCConfig function.

FrIf05315: If parameter FrIf_CtrlIdx of FrIf_ReadCCConfig has an invalid value and if development error detection is enabled (i.e. FRIF_DEV_ERROR_DETECT 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.

8.4.17 FrIf_GetWakeupRxStatus

FrIf05702:

Service name:	FrIf_GetWakeupRxStatus	
Syntax:	<pre>Std_ReturnType FrIf_GetWakeupRxStatus(uint8 FrIf_CtrlIdx, uint8* FrIf_WakeupRxStatusPtr)</pre>	
Service ID[hex]:	0x3e	
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	<p>Address where bitcoded wakeup reception status shall be stored.</p> <p>Bit 0: Wakeup received on channel A indicator</p> <p>Bit 1: Wakeup received on channel B indicator</p> <p>Bit 2-7: Unused</p>
Return value:	Std_ReturnType	<p>E_OK: API call finished successfully.</p> <p>E_NOT_OK: API call aborted due to errors.</p>
Description:	Wraps the FlexRay Driver API function Fr_GetWakeupRxStatus and gets the wakeup received information from the FlexRay controller.	

Frlf05417: The function `Frlf_GetWakeupRxStatus` shall be pre compile time configurable ON/OFF by the configuration parameter `FrlfGetWakeupRxStatusSupport` (derived from configuration parameter `FrlfGetWakeupRxStatusSupport`)

Frlf05700: If development error detection for the Frlf module is enabled: if the function `Frlf_GetWakeupRxStatus` is called before the Fr was initialized successfully, the function `Frlf_GetWakeupRxStatus` shall raise the development error `FRIF_E_NOT_INITIALIZED` and return `E_NOT_OK`.

Frlf05701: If development error detection for the Frlf module is enabled: the function `Frlf_GetWakeupRxStatus` shall check the parameter `Frlf_CtrlIdx` for being valid. If `Frlf_CtrlIdx` is invalid, the function `Frlf_GetWakeupRxStatus` shall raise the development error `FRIF_E_INV_CTRL_IDX` and return `E_NOT_OK`.

8.5 Interrupt Service Routines

8.5.1 FrIf_JobListExec_<ClstIdx>

Frlf05040:

Service name:	FrIf_JobListExec_<ClstIdx>
Syntax:	void FrIf_JobListExec_<ClstIdx>()
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 ClstIdx.

This API service of the FlexRay Interface processes the FlexRay Job List of the FlexRay Cluster with index ClstIdx.

For a detailed description of this API service, please refer to chapter 7.6.2.2.

Caveats: This API service of the FlexRay Interface exists once per FlexRay Cluster. The API name contains the index of the respective FlexRay Cluster (ClstIdx).

For each FlexRay Cluster (identified by index ClstIdx), the respective API service FrIf_JobListExec_<ClstIdx> 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 ClstIdx, if the CC does **not guarantee asynchronous buffer access**.

The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[Frlf05389\]](#).

Note:

If the CC guarantees asynchronous buffer access, the execution of FrIf_JobListExec<ClstIdx> can run in a regular OS task.

8.6 Call-back Notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file `FrIf_Cbk.h`

8.6.1 FrIf_Cbk_WakeupByTransceiver

FrIf05041:

Service name:	FrIf_Cbk_WakeupByTransceiver	
Syntax:	<pre>void FrIf_Cbk_WakeupByTransceiver(uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx)</pre>	
Service ID[hex]:	0x3b	
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_Cbk_WakeupByTransceiver()</code> .	

Parameter `FrIf_CtrlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CTRL_IDX`) if `FrIf_CtrlIdx` has an invalid value.

Parameter `FrIf_ChnlIdx`: If development error detection is enabled (i.e. [FRIF_DEV_ERROR_DETECT](#) equals ON), it shall be reported to the DET module (using `FRIF_E_INV_CHNL_IDX`) if `FrIf_ChnlIdx` has an invalid value.

This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API function `FrTrcv_Cbk_WakeupByTransceiver()` by:

Translating (based on static [FrIf](#) 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`).

Calling `FrTrcv_CbkWakeupByTransceiver()` of the determined FlexRay Driver with the parameters determined as described above.

Caveats: The FlexRay Interface has to be initialized with a call of `FrIf_Init()` before this API service may be called, see [\[FrIf05389\]](#).

8.7 Scheduled Functions

8.7.1 FrIf_MainFunction_<ClstIdx>

Frlf05042:

Service name:	FrIf_MainFunction_<ClstIdx>
Syntax:	void FrIf_MainFunction_<ClstIdx> ()
Service ID[hex]:	0x33
Timing:	VARIABLE_CYCLIC
Description:	This function will be called cyclically by a task body provided by the BSW Scheduler.

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

- Initially program the absolute timer interrupt in order to start the execution of FrIf_JobListExec_<ClstIdx>() if the CC does not support asynchronous buffer access.
- Monitoring the proper (in time) execution of the FrIf_JobListExec_<ClstIdx>() and resynchronize the Joblist if necessary.

Please refer to chapter 7.3 for a detailed description.

Pre condition: This API service of the FlexRay Interface is cyclically called from a task body provided by the [BSW](#) Scheduler.

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

Caveats: This API service of the FlexRay Interface exists once per FlexRay Cluster. The API name contains the index of the respective FlexRay Cluster (ClstIdx).

The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [\[Frlf05389\]](#).

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.

Frlf05043:

API function	Description
Dem_ReportErrorStatus	Reports errors to the DEM.
FrTrcv_ClearTransceiverWakeup	This function clears a pending wake up event.
FrTrcv_DisableTransceiverWakeup	This function disables the notification for wake up events on the addressed bus.
FrTrcv_EnableTransceiverWakeup	This function enables the notification for wake up events on the addressed bus.
FrTrcv_GetTransceiverMode	This function returns the actual state of the transceiver.
FrTrcv_GetTransceiverWUReason	This function returns the wakeup reason.
FrTrcv_GetVersionInfo	This service returns the version information of this module.
FrTrcv_MainFunction	--
FrTrcv_SetTransceiverMode	This service returns the transceiver mode.
FrTrcv_TrcvInit	This service initializes the FrTrcv.
Fr_AbortCommunication	Invokes the CC CHI command 'FREEZE'.
Fr_AckAbsoluteTimerIRQ	Resets the interrupt condition of an absolute timer.
Fr_AckRelativeTimerIRQ	Resets the interrupt condition of a relative timer.
Fr_AllowColdstart	Invokes the CC CHI command 'ALLOW_COLDSTART'.
Fr_CancelAbsoluteTimer	Stops an absolute timer.
Fr_CancelRelativeTimer	Stops a relative timer.
Fr_CheckMTS	Checks the MTS.
Fr_CheckTxLPduStatus	Checks the transmit status of the LSdu.
Fr_ControllerInit	Initializes a FlexRay CC.
Fr_DisableAbsoluteTimerIRQ	Disables the interrupt line of an absolute timer.
Fr_DisableRelativeTimerIRQ	Disables the interrupt line of a timer.
Fr_EnableAbsoluteTimerIRQ	Enables the interrupt line of an absolute timer.
Fr_EnableRelativeTimerIRQ	Enables the interrupt line of a relative timer.
Fr_GetAbsoluteTimerIRQStatus	Gets IRQ status of an absolute timer.
Fr_GetGlobalTime	Gets the current global FlexRay time.
Fr_GetNmVector	Gets the network management vector of the last communication cycle.
Fr_GetPOCStatus	Gets the POC status.
Fr_GetRelativeTimerIRQStatus	Gets IRQ status of a relative timer.
Fr_GetSyncState	Gets the sync state.
Fr_GetVersionInfo	Returns the version information of this module.
Fr_HaltCommunication	Invokes the CC CHI command 'HALT'.
Fr_Init	Initializes the Fr.
Fr_PrepareLPdu	Prepares a LPdu.
Fr_ReceiveRxLPdu	Receives data from the FlexRay network.
Fr_SendMTS	Triggers a MTS.
Fr_SendWUP	Invokes the CC CHI command 'WAKEUP'.
Fr_SetAbsoluteTimer	Sets the absolute FlexRay timer.
Fr_SetExtSync	Adjusts the global time of a FlexRay CC to an external clock source.

Fr_SetRelativeTimer	Sets the FlexRay timer.
Fr_SetWakeupChannel	Sets a wakeup channel.
Fr_StartCommunication	Starts communication.
Fr_StopMTS	Stops the periodic transmission of MTS symbols.
Fr_TransmitTxLPdu	Transmits data on the FlexRay network.

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

Frlf05044:

API function	Description
Det_ReportError	Service to report development errors.
Fr_GetChannelStatus	Gets the channel status information.
Fr_GetClockCorrection	Gets the current clock correction values. See variables vInterimRateCorrection and vInterimOffsetCorrection of [12] for details.
Fr_GetWakeupRxStatus	Gets the wakeup received information from the FlexRay controller.
Fr_ReadCCConfig	Reads a FlexRay protocol configuration parameter for a particular FlexRay controller out of the module's configuration.

8.8.3 Configurable Interfaces

This chapter lists all interfaces where the target API service of any upper layer, which require one or more of these mentioned interfaces to be called has to be set up by static configuration of the FlexRay Interface.

These call-out services are specified and implemented in the upper layer BSW modules, which use the FlexRay Interface according to [2]. The specific call-out notification is specified in the corresponding AUTOSAR SWS document (see chapter 3).

In addition to upper layer AUTOSAR BSW modules, the Frlf shall, with the functionality described within the specification in hand, also support other non-AUTOSAR upper layer software modules, provided that these modules interact with the Frlf in the same manner as the upper layer AUTOSAR BSW modules. In particular, those non-AUTOSAR modules need to provide APIs as described in this chapter. One example for such a non-AUTOSAR software module is a proprietary XCPonFlexRay module that users may add to their AUTOSAR BSW stack.

8.8.3.1 <UL_RxIndication>

Frlf05045:

Service name:	<UL_RxIndication>
Syntax:	<pre>void <UL_RxIndication>(PduIdType FrIf_RxPduId, const PduInfoType* PduInfoPtr)</pre>
Sync/Async:	Synchronous

Reentrancy:	Reentrant for different FrIf_RxPduId, Non reentrant for the same FrIf_RxPduId	
Parameters (in):	FrIf_RxPduId	PDU-ID of FlexRay PDU that has been received
	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication module.	

Caveats: This API service is called during the execution of the FlexRay Job List Execution Function.

8.8.3.2 <UL_TxConfirmation>

Frlf05046:

Service name:	<UL_TxConfirmation>	
Syntax:	<pre>void <UL_TxConfirmation>(PduIdType FrIf_TxPduId)</pre>	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrIf_TxPduId, Non reentrant for the same FrIf_TxPduId	
Parameters (in):	FrIf_TxPduId	PDU-ID of FlexRay PDU whose transmission is being confirmed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This API service of an upper layer BSW module (e.g. PduR, FrTp, FrNm) is called by the FlexRay Interface to confirm to this upper layer BSW module that the PDU with index FrIf_TxPduId has been transmitted via the FlexRay Communication System.	

Caveats: This API service is called during the execution of the FlexRay Job List Execution Function.

8.8.3.3 <UL_TriggerTransmit>

Frlf05047:

Service name:	<UL_TriggerTransmit>	
Syntax:	<pre>Std_ReturnType <UL_TriggerTransmit>(PduIdType FrIf_TxPduId, PduInfoType* PduInfoPtr)</pre>	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrIf_TxPduId, Non reentrant for the same FrIf_TxPduId	
Parameters (in):	FrIf_TxPduId	PDU-ID of FlexRay PDU that shall be copied to the FrIf
Parameters (inout):	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

Caveats: This API service is called during the execution of the FlexRay Job List Execution Function.

8.8.3.4 <Free_Op_A>

[SWS_Frlf_05316]

Service name:	<Free_Op_A>	
Syntax:	void <Free_Op_A>(uint8 FrIf_CtrlIdx, uint16 FrIf_LPduIdx)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different Frlf_LPduIndex, non reentrant for same Frlf_LPduIndex	
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.	

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

8.8.3.5 <Free_Op_B>

[SWS_Frlf_05317]

Service name:	<Free_Op_B>	
Syntax:	void <Free_Op_B>(uint8 FrIf_CtrlIdx, uint16 FrIf_LPduIdx)	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different Frlf_LPduIndex, non reentrant for same Frlf_LPduIndex	
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.	

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

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 [Frlf](#) 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

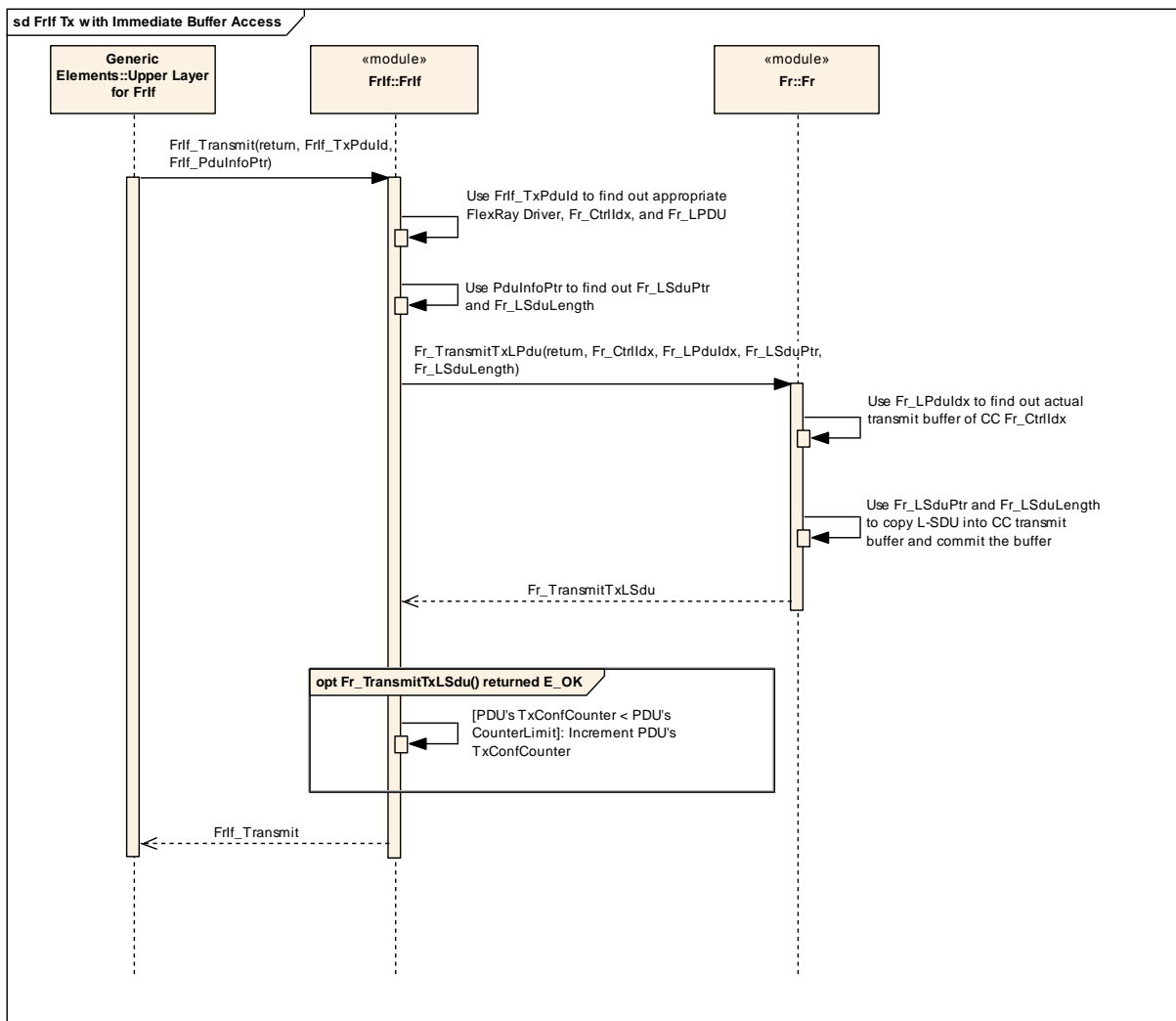


Figure 9-1: TransmitWithImmediateBufferAccess

9.1.2 TransmitWithDecoupledBufferAccess

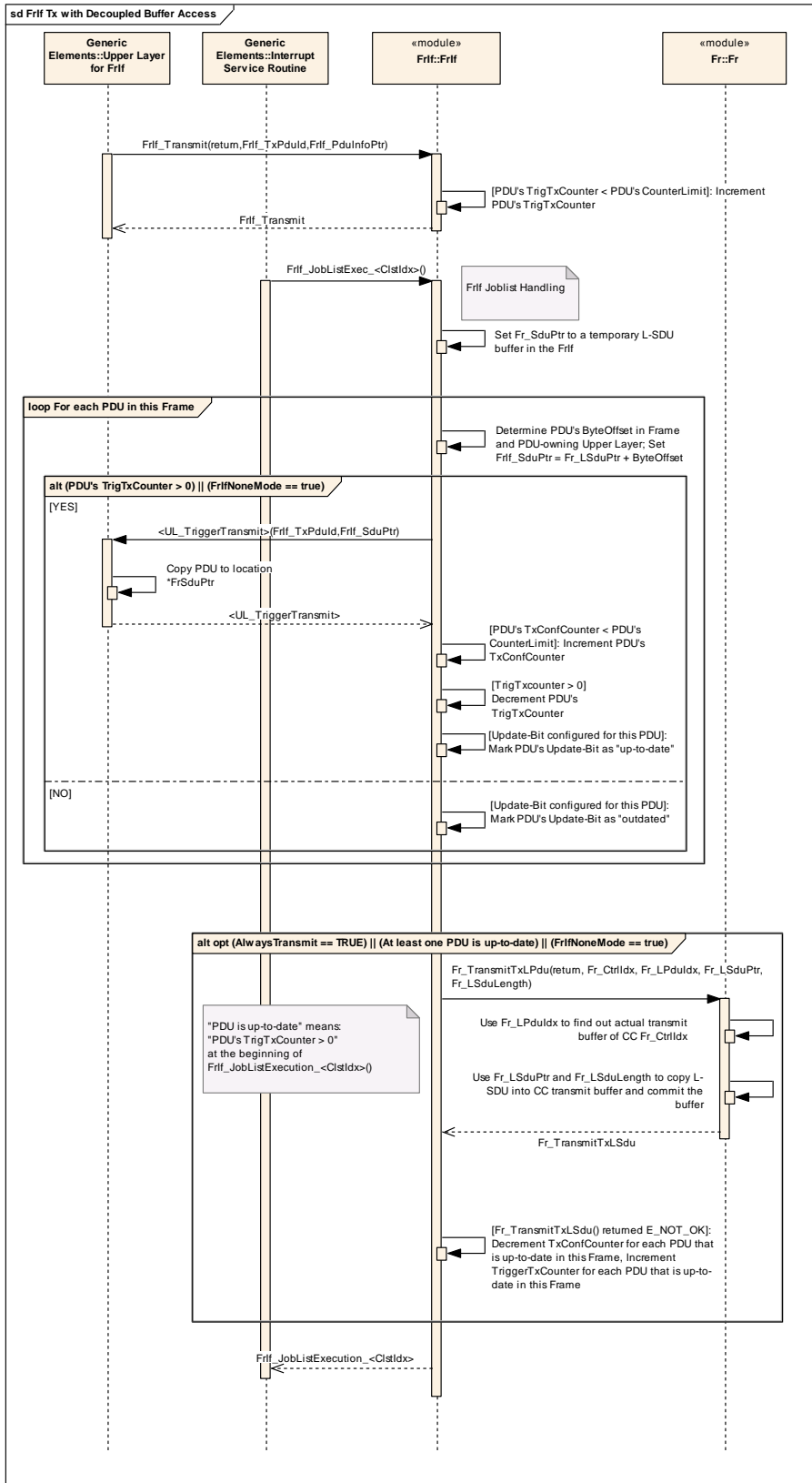


Figure 9-2: TransmitWithDecoupledBufferAccess

9.1.3 ProvideTxConfirmation

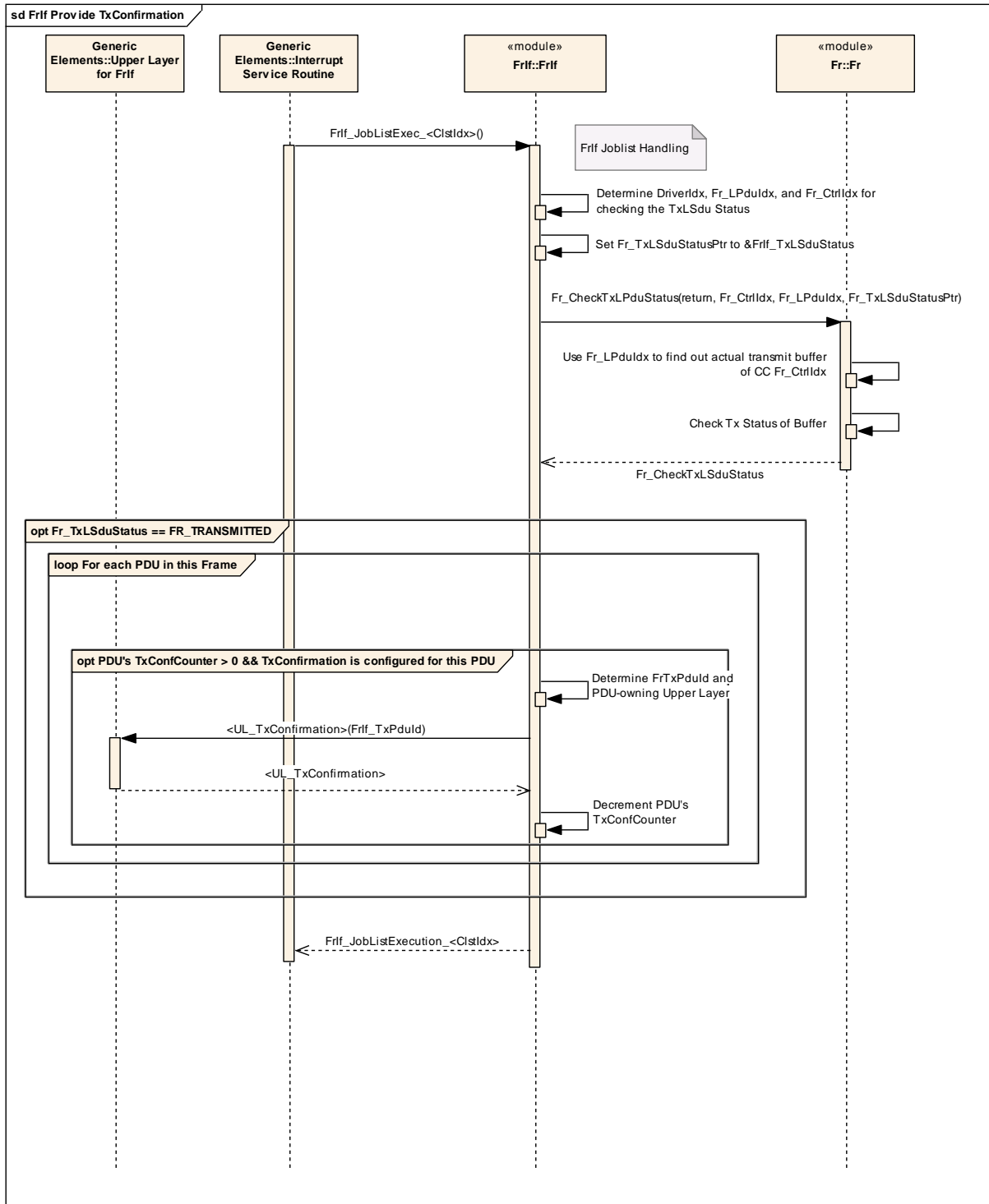


Figure 9-3: ProvideTxConfirmation

9.2 Data Reception

9.2.1 ReceiveAndIndicate

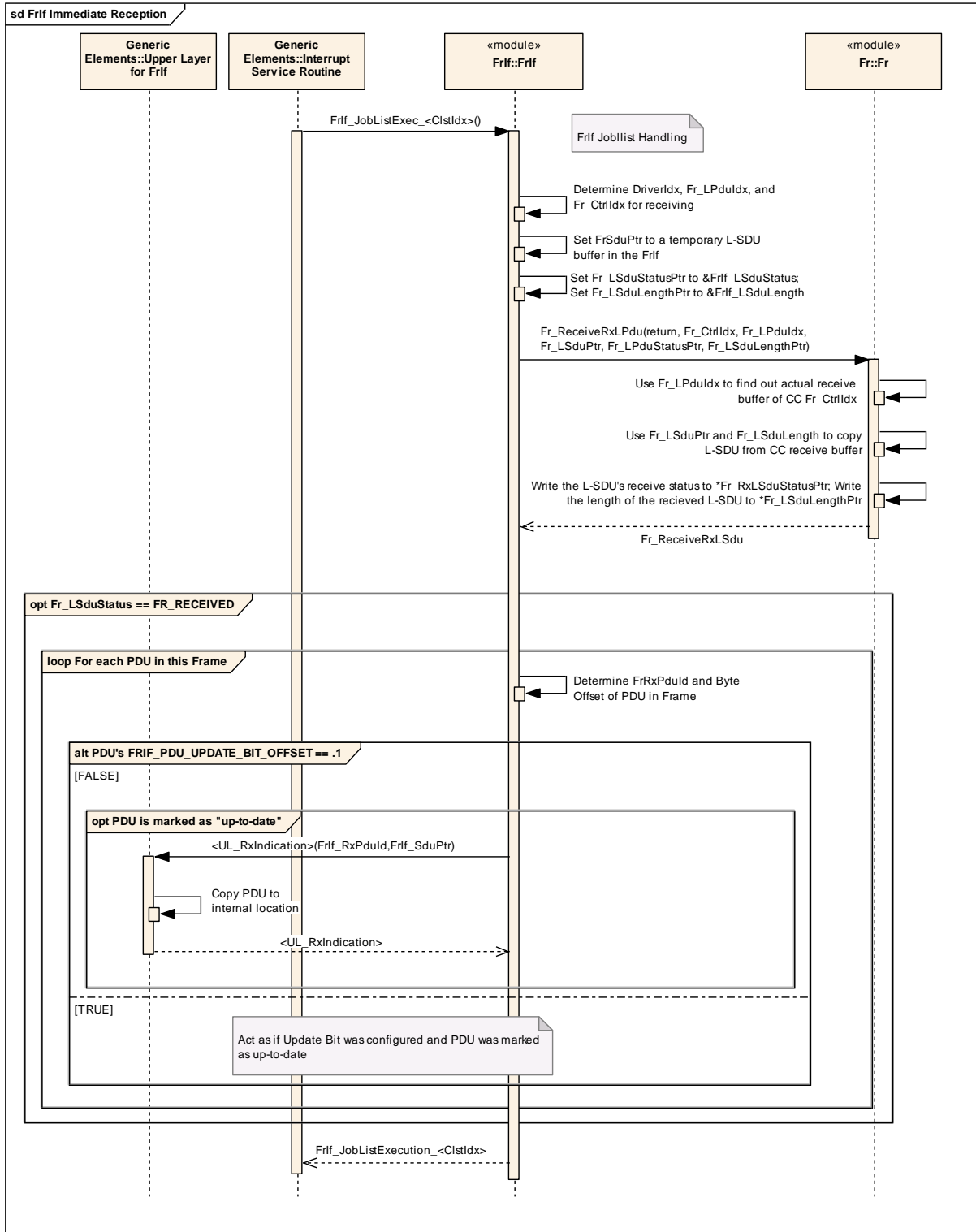


Figure 9-4: ReceiveAndIndicate

9.2.2 ReceiveAndStore

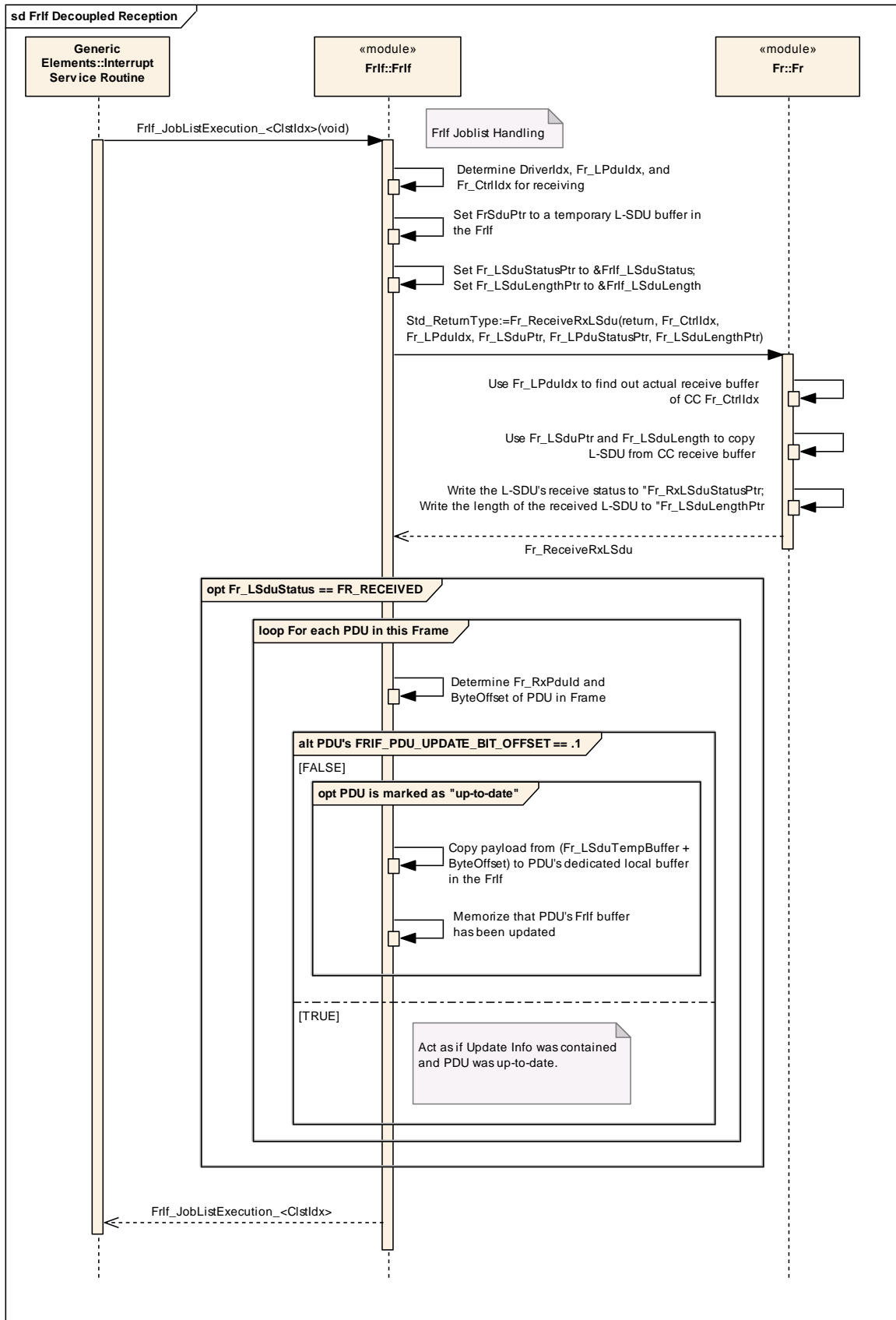


Figure 9-5: ReceiveAndStore

9.2.3 ProvideRxIndication

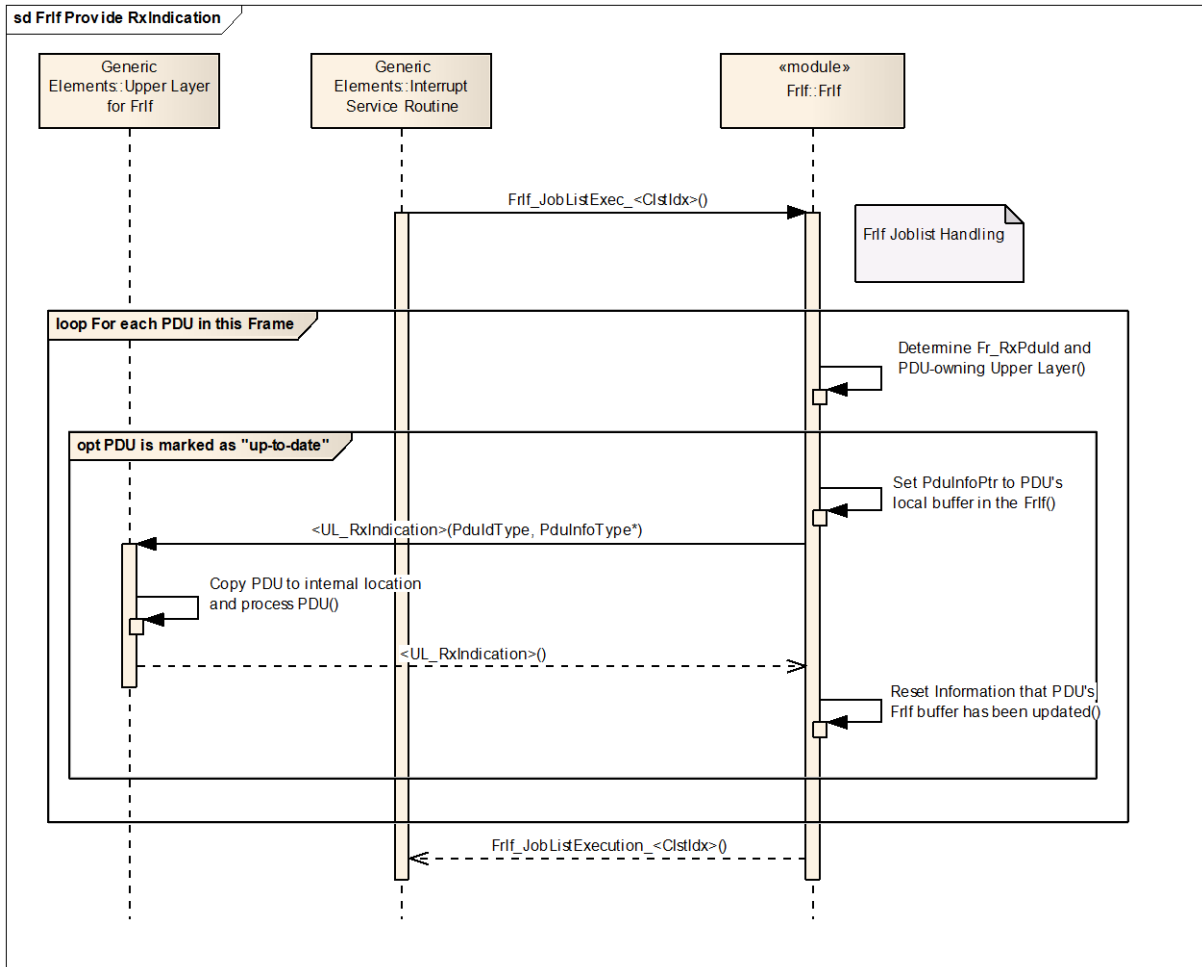


Figure 9-6: ProvideRxIndication

10 Configuration Specification

This chapter defines configuration parameters and their clustering into containers. Chapter 10.1 gives information to help understanding the subsequent chapters. Chapter 10.2 specifies the structure (containers) and the parameters of the FlexRay Interface. Chapter 10.3 specifies published information of the FlexRay Interface.

10.1 How to Read this Chapter

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

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [13]
This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

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

10.1.1 Configuration and Configuration Parameters

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

The configuration of parameters can be achieved at different times during the software process: [pre compile time](#), before [link time](#) or [post build time](#). In the following, the term "configuration class" (of a parameter) shall be used in order to refer to a specific configuration point in time.

Frlf05144:

All configuration data of the [Frlf](#) that is definable [at system configuration time](#) shall be re-loadable into the ECU [by a flashing process](#).

10.1.2 Variants

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

10.1.3 Containers

Frlf05077:

Containers structure the set of configuration parameters. This means:

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

10.1.4 Specification Template for Configuration Parameters

The following tables consist of three sections:

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

Pre compile time - specifies whether the configuration parameter shall be of configuration class Pre-compile time or not

Label	Description
X	The configuration parameter shall be of configuration class <i>Pre-compile time</i> .
--	The configuration parameter shall never be of configuration class <i>Pre-compile time</i> .

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

Label	Description
x	The configuration parameter shall be of configuration class <u>link time</u> .
--	The configuration parameter shall never be of configuration class <u>link time</u> .

Post build time - specifies whether the configuration parameter shall be of configuration class post build time or not

Label	Description
x	The configuration parameter shall be of configuration class <u>post build time</u> and no specific implementation is required.
L	<i>Loadable</i> - the configuration parameter shall be of configuration class <u>post build time</u> and only one configuration parameter set resides in the ECU.
M	<i>Multiple</i> - the configuration parameter shall be of configuration class <u>post build time</u> and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class <u>post build time</u> .

10.2 Containers and Configuration Parameters

Frlf05070:

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described in 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 shall extract all information to configure the [Frlf](#).

Frlf05071:

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

Frlf05072:

Configuration rules and constraints for plausibility checks shall be performed during configuration time, wherever possible.

Frlf05073:

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

10.2.1 Variants

VARIANT-POST-BUILD: All configuration parameters in container 'FrlfGeneral' shall be configurable at pre-compile time. All other configuration parameters shall be configurable at post-build-time.

Use case: Object code delivery, selectable configuration

VARIANT-PRE-COMPILE: All configuration parameters shall be configurable at pre-compile time.

Use case: Execution time optimizations

10.2.2 Frlf

SWS Item	ECUC_Frlf_06087 :
Module Name	<i>Frlf</i>
Module Description	Configuration of the Frlf (FlexRay Interface) module.

Included Containers		
Container Name	Multiplicity	Scope / Dependency
FrlfConfig	1	Configuration of the FlexRay Interface. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
FrlfGeneral	1	This container contains the general configuration parameters of the FlexRay Interface.

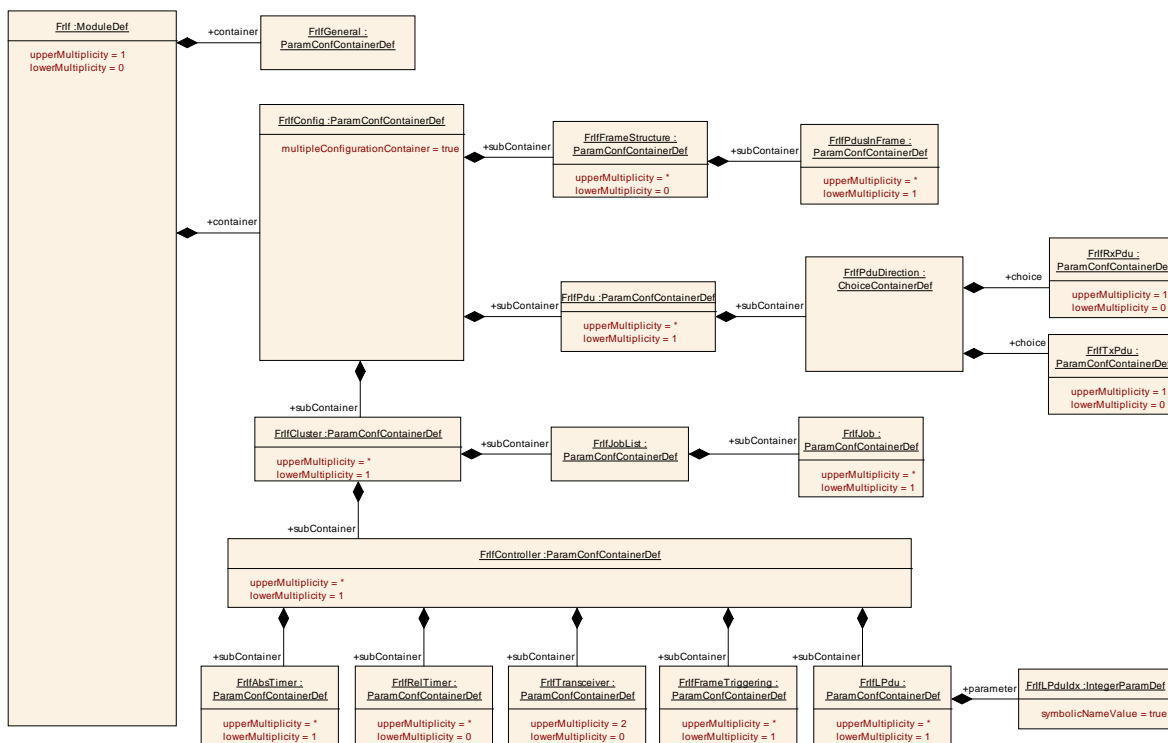


Figure 10-1: FrIf

10.2.3 FrIfGeneral

SWS Item	FrIf05360 :
Container Name	FrIfGeneral{FRIF_GENERAL_CONFIGURATION}
Description	This container contains the general configuration parameters of the FlexRay Interface.
Configuration Parameters	

SWS Item	FrIf06116_Conf :		
Name	FrIfCCReadbackApiSupport {FRIF_CC_READBACK_API_SUPPORT}		
Description	Enables/disables the "FrIf_ReadCCConfig" API (FrIf05313). This should not be mixed up with CCReadbackSupport functionality of FlexRay Driver during FrControllerInit (FR647).		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	FrIf06080 :		
Name	FrIfDevErrorDetect {FRIF_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off true: Development Error Detection and Notification on false: Development Error Detection and Notification off		

Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf06117_Conf :		
Name	FrlfFreeOpAApiName {FRIF_FREE_OP_A_API_NAME}		
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	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf06118_Conf :		
Name	FrlfFreeOpBApiName {FRIF_FREE_OP_B_API_NAME}		
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	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf06119_Conf :		
Name	FrlfFreeOpsHeader {FRIF_FREE_OPS_HEADER}		
Description	Defines header file for configurable FREE_OP_A / FREE_OP_B functions.		
Multiplicity	0..1		
Type	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf06106_Conf :		
Name	FrlfGetClockCorrectionSupport		

	{FRIF_GET_CLOCK_CORRECTION_SUPPORT}		
Description	Enables/disables the "FrIf_GetClockCorrection" API (FrIf05071).		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	FrIf06105_Conf :		
Name	FrIfGetGetChannelStatusSupport {FRIF_GET_GET_CHANNEL_STATUS_SUPPORT}		
Description	Enables/disables the "FrIf_GetChannelStatus" API (FrIf05030).		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	FrIf06114 :		
Name	FrIfGetNmVectorSupport {FRIF_GET_NM_VECTOR_SUPPORT}		
Description	Configuration parameter to enable/disable FrIf support to request the FlexRay hardware NMVector.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	FrIf06081 :		
Name	FrIfNumClstSupported {FRIF_CLST_IDX_MAX_SUPPORTED}		
Description	Maximum number of FlexRay Clusters that the FlexRay Interface supports.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	FrIf06082 :		
Name	FrIfNumCtrlSupported {FRIF_CTRL_IDX_MAX_SUPPORTED}		
Description	Maximum number of FlexRay CCs that the FlexRay Interface supports		
Multiplicity	1		

Type	IntegerParamDef		
Range	1 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf06116 :		
Name	FrlfPublicCddHeaderFile {FRIF_PUBLIC_CDD_HEADER_FILE}		
Description	Defines header files for callback functions which shall be included in case of CDDs. Range of characters is 1.. 32.		
Multiplicity	0..*		
Type	StringParamDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf00001_Conf :		
Name	FrlfUnusedBitValue {FRIF_UNUSED_BIT_VALUE}		
Description	Set unused bits to a defined value.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 1		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	Frlf06083 :		
Name	FrlfVersionInfoApi {FRIF_VERSION_INFO_API}		
Description	Enables/disables the existence of the Frlf_GetVersionInfo() API service true: Frlf_GetVersionInfo() API service exists false: Frlf_GetVersionInfo() API service does not exist		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

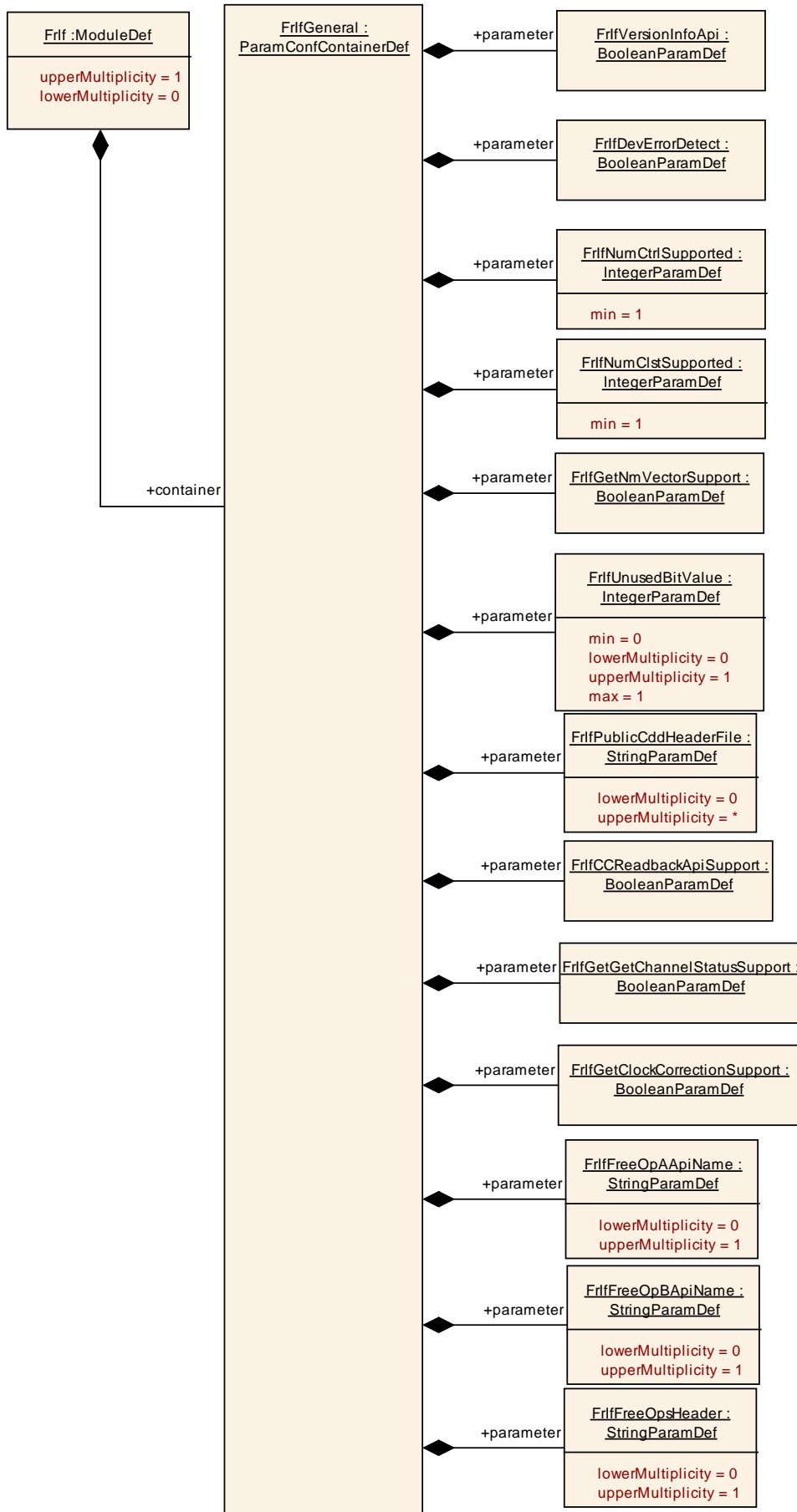


Figure 10-2 : FrIfGeneral

10.2.4 FrIfCluster

SWS Item	FrIf05366 :
Container Name	FrIfCluster{FRIF_CLUSTER}
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.
Configuration Parameters	

SWS Item	FrIf06002 :		
Name	FrIfClstIdx {FRIF_CLST_IDX}		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06005 :		
Name	FrIfGAssumedPrecision {G_ASSUMED_PRECISION}		
Description	This parameter is deprecated and will be removed in future. Old description: Assumed precision of the application network.		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06006 :		
Name	FrIfGChannels {G_CHANNELS}		
Description	The channels that are used by the cluster. Implementation Type: Fr_ChannelType		
Multiplicity	1		
Type	EnumerationParamDef		
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	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06007 :		
Name	FrIfGClusterDriftDamping {G_CLUSTER_DRIFT_DAMPING}		

Description	The cluster drift damping factor, based on the longest microtick gdMaxMicrotick used in the cluster. Used to compute the local cluster drift damping factor pClusterDriftDamping [Microticks].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 5		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06008 :		
Name	FrlfGColdStartAttempts {G_COLD_START_ATTEMPTS}		
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	IntegerParamDef		
Range	2 .. 31		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06102 :		
Name	FrlfGCycleCountMax {G_CYCLE_COUNT_MAX}		
Description	Maximum cycle counter value in a given cluster. Remark: Set to 63 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	7 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06009 :		
Name	FrlfGListenNoise {G_LISTEN_NOISE}		
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 cluster parameter pdListenTimeout.		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 16		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06010 :		
Name	FrlfGMacroPerCycle {G_MACRO_PER_CYCLE}		
Description	Number of macroticks in a communication cycle. Note: Lower limit 10 for FlexRay Protocol 2.1 Rev. A compliance		

Multiplicity	1		
Type	IntegerParamDef		
Range	8 .. 16000		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	FrIf06011 :		
Name	FrIfGMaxWithoutClockCorrectFatal {G_MAX_WITHOUT_CLOCK_CORRECTION_FATAL}		
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	IntegerParamDef		
Range	1 .. 15		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06012 :		
Name	FrIfGMaxWithoutClockCorrectPassive {G_MAX_WITHOUT_CLOCK_CORRECTION_PASSIVE}		
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	IntegerParamDef		
Range	1 .. 15		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope Dependency	scope: Module		

SWS Item	FrIf06013 :		
Name	FrIfGNetworkManagementVectLength {G_NETWORK_MANAGEMENT_VECTOR_LENGTH}		
Description	Length of the Network Management vector in a cluster [bytes]		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 12		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06014 :		
-----------------	--------------------	--	--

Name	FrIfGNumberOfMinislots {G_NUMBER_OF_MINISLOTS}		
Description	Number of minislots in the dynamic segment Remark: Upper limit 7986 for FlexRay Protocol 2.1 Rev. A compliance		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 7988		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06015 :		
Name	FrIfGNumberOfStaticSlots {G_NUMBER_OF_STATIC_SLOTS}		
Description	Number of static slots in the static segment		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 1023		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06016 :		
Name	FrIfGOffsetCorrectionMax {G_OFFSET_CORRECTION_MAX}		
Description	describes the maximum value which the offset correction should assume in seconds.		
Multiplicity	1		
Type	FloatParamDef		
Range	1.5E-7 .. 3.83567E-4		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06018 :		
Name	FrIfGPayloadLengthStatic {G_PAYLOAD_LENGTH_STATIC}		
Description	Payload length of a static frame [16 bit words]		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 127		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06019 :		
Name	FrIfGSyncFrameIDCountMax {G_SYNC_NODE_MAX}		
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	IntegerParamDef		
Range	2 .. 15		

Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06020 :		
Name	FrlfGdActionPointOffset {GD_ACTION_POINT_OFFSET}		
Description	Number of Macroticks the action point is offset from the beginning of a Static Slots or symbol window.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	Frlf06021 :		
Name	FrlfGdBit {GD_BIT}		
Description	Nominal bit time in seconds		
Multiplicity	1		
Type	EnumerationParamDef		
Range	T100NS		
	T200NS		
	T400NS		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06022 :		
Name	FrlfGdBitMax {GD_BIT_MAX}		
Description	Maximum bit time taking into account the allowable clock deviation of each node (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06023 :		
Name	FrlfGdBitMin {GD_BIT_MIN}		
Description	Minimum bit time taking into account the allowable clock deviation of each node (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: Module
---------------------------	---------------

SWS Item	Frlf06024 :		
Name	FrlfGdCasRxLowMax {GD_CAS_RX_LOW_MAX}		
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	IntegerParamDef		
Range	28 .. 254		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	Frlf06025 :		
Name	FrlfGdCycle {GD_CYCLE}		
Description	Length of the cycle, expressed in [s] Remark: Lower limit 0.000024 for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	FloatParamDef		
Range	1E-5 .. 0.016		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06026 :		
Name	FrlfGdDynamicSlotIdlePhase {GD_DYNAMIC_SLOT_IDLE_PHASE}		
Description	Duration of the idle phase within a dynamic slot [Minislots].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 2		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06101 :		
Name	FrlfGdIgnoreAfterTx {GD_IGNORE_AFTER_TX}		
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	IntegerParamDef		
Range	0 .. 15		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06027 :		
Name	FrlfGdMacrotick {GD_MACROTICK}		
Description	Duration of the cluster wide nominal macrotick, expressed in s		

Multiplicity	1		
Type	FloatParamDef		
Range	1E-6 .. 6E-6		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06028 :		
Name	FrlfGdMaxInitializationError {GD_MAX_INITIALIZATION_ERROR}		
Description	Maximum error that a node may have following integration in seconds.		
Multiplicity	1		
Type	FloatParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06029 :		
Name	FrlfGdMaxMicrotick {GD_MAX_MICROTICK}		
Description	Maximum Microtick length of all Microticks configured within a Cluster.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	T100NS		
	T12_5NS		
	T200NS		
	T25NS		
	T50NS		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06030 :		
Name	FrlfGdMaxPropagationDelay {GD_MAX_PROPAGATION_DELAY}		
Description	Maximum propagation delay of a Cluster (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06031 :		
Name	FrlfGdMinPropagationDelay {GD_MIN_PROPAGATION_DELAY}		
Description	Minimum propagation delay of a Cluster (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		

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

SWS Item	FrIf06032 :		
Name	FrIfGdMiniSlotActionPointOffset {GD_MINI_SLOT_ACTION_POINT_OFFSET}		
Description	Number of Macroticks the Minislot action point is offset from the beginning of a Minislot [Macroticks].		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 31		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06033 :		
Name	FrIfGdMinislot {GD_MINISLOT}		
Description	Duration of a minislot [Macroticks]		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	FrIf06034 :		
Name	FrIfGdNit {GD_NIT}		
Description	Duration of the Network Idle Time [Macroticks]. Remark: Upper limit 805 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 15978		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06035 :		
Name	FrIfGdSampleClockPeriod {GD_SAMPLE_CLOCK_PERIOD}		
Description	Sample clock period		
Multiplicity	1		
Type	EnumerationParamDef		
Range	T12_5NS		
	T25NS		
	T50NS		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06036 :		
Name	FrlfGdStaticSlot {GD_STATIC_SLOT}		
Description	Duration of a Static Slot [Macroticks]. Remark: Range 4 to 661 for FlexRay Protocol 2.1 Rev. A compliance		
Multiplicity	1		
Type	IntegerParamDef		
Range	3 .. 664		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06037 :		
Name	FrlfGdSymbolWindow {GD_SYMBOL_WINDOW}		
Description	Duration of the symbol window [Macroticks]. Remark: Range 0-142 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 162		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06100 :		
Name	FrlfGdSymbolWindowActionPointOffset {GD_SYMBOL_WINDOW_ACTION_POINT_OFFSET}		
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	IntegerParamDef		
Range	1 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06038 :		
Name	FrlfGdTssTransmitter {GD_TSS_TRANSMITTER}		
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	IntegerParamDef		
Range	1 .. 15		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06039 :		
Name	FrlfGdWakeupRxIdle {GD_WAKEUP_SYMBOL_RX_IDLE}		

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	IntegerParamDef		
Range	8 .. 59		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06040 :		
Name	FrlfGdWakeupRxLow {GD_WAKEUP_SYMBOL_RX_LOW}		
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	IntegerParamDef		
Range	8 .. 59		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06041 :		
Name	FrlfGdWakeupRxWindow {GD_WAKEUP_SYMBOL_RX_WINDOW}		
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	IntegerParamDef		
Range	76 .. 485		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06043 :		
Name	FrlfGdWakeupTxActive {GD_WAKEUP_SYMBOL_TX_LOW}		
Description	Number of bits used by the node to transmit the LOW phase of a wakeup 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	IntegerParamDef		
Range	15 .. 60		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06042 :		
Name	FrlfGdWakeupTxIdle {GD_WAKEUP_SYMBOL_TX_IDLE}		
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	IntegerParamDef		
Range	45 .. 180		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06003 :		
Name	FrlfMainFunctionPeriod {FRIF_MAINFUNCTION_PERIOD}		
Description	The execution cycle of the Frlf_MainFunction_<cluster>() 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	FloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
FrlfController	1..*	This container contains the configuration of FlexRay CC.
FrlfJobList	1	This container specifies a list of all FlexRay Jobs of the Cluster to be performed by Frlf_JobListExec_<ClstIdx>().

10.2.5 FrlfController

SWS Item	Frlf05363 :
Container Name	FrlfController
Description	This container contains the configuration of FlexRay CC.
Configuration Parameters	

SWS Item	Frlf06045 :		
Name	FrlfCtrlIdx {FRIF_CTRL_IDX}		
Description	This parameter provides a zero-based consecutive index of the FlexRay Communication Controllers. Upper layer BSW modules and the Frlf itself use this index to identify a FlexRay CC.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	Frlf06044 :		
Name	FrlfFrCtrlRef {FRIF_FR_CTRL}		
Description	Reference to a Controller, which is handled by a specific Driver. This reference is unique for the ECU.		
Multiplicity	1		
Type	Reference to [FrController]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	Multiplicity	Scope / Dependency
FrlfAbsTimer	1..*	This container contains the configuration of an absolute timer of a FlexRay CC.
FrlfFrameTriggering	1..*	A Frame triggering contains the communication parameters of the FlexRay Frame as well as a reference to the Frame Construction Plan.
FrlfLPdu	1..*	LPdu is an abstraction of all FlexRay Frames (L-PDUs) belonging to the same Frame Triggering FrlfFrameTriggering.
FrlfRelTimer	0..*	This container contains the configuration of a relative timer of a FlexRay CC.
FrlfTransceiver	0..2	Up to two FlexRay Transceivers may connect a Controller to a Cluster. This container realizes a Controller-Transceiver assignment.

10.2.6 FrlfAbsTimer

SWS Item	Frlf05361 :
Container Name	FrlfAbsTimer{FRIF_ABS_TIMER}
Description	This container contains the configuration of an absolute timer of a FlexRay

	CC.
Configuration Parameters	

SWS Item	Frlf06047 :		
Name	FrlfAbsTimerIdx {FRIF_ABS_TIMER_IDX}		
Description	This parameter provides a zero-based consecutive index of the absolute timers. Upper layer BSW modules use this index to identify an absolute timer.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	Frlf06046 :		
Name	FrlfFrAbsTimerRef {FRIF_FR_ABS_TIMER}		
Description	Reference to an absolute timer, which is handled by a specific FlexRay Driver. This reference is unique for the ECU, therefore, an explicit reference to the FlexRay Driver is not necessary.		
Multiplicity	1		
Type	Reference to [FrAbsoluteTimer]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.7 FrlfRelTimer

SWS Item	Frlf05362 :		
Container Name	FrlfRelTimer{FRIF_REL_TIMER_IDX}		
Description	This container contains the configuration of a relative timer of a FlexRay CC.		
Configuration Parameters			

SWS Item	Frlf06060 :		
Name	FrlfRelTimerIdx {FRIF_REL_TIMER_IDX}		
Description	This parameter provides a zero-based consecutive index of the relative timers. Upper layer BSW modules use this index to identify a relative timer.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	Frlf06059 :		
Name	FrlfFrRelTimerRef {FRIF_FR_REL_TIMER}		

Description	Reference to a relative timer, which is handled by a specific FlexRay Driver. This reference is unique for the ECU, therefore, an explicit reference to the FlexRay Driver is not necessary.		
Multiplicity	1		
Type	Reference to [FrRelativeTimer]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.8 FrlfFrameTriggering

SWS Item	ECUC_Frlf_06090 :		
Container Name	FrlfFrameTriggering		
Description	A Frame triggering contains the communication parameters of the FlexRay Frame as well as a reference to the Frame Construction Plan.		
Configuration Parameters			

SWS Item	Frlf06049 :		
Name	FrlfAllowDynamicLSduLength {FRIF_ALLOW_DYNAMIC_LSDU_LENGTH}		
Description	Allows L-PDU length reduction ('FrlfLSduLength' 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	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	ECUC_Frlf_00013 :		
Name	FrlfAlwaysTransmit {FRIF_ALWAYS_TRANSMIT}		
Description	Defines whether the driver's API function Fr_TransmitTxLPdu() shall always be called for this L-PDU.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06051 :		
Name	FrlfBaseCycle {FRIF_BASE_CYCLE}		
Description	This parameter contains the FlexRay Base Cycle used to transmit this FlexRay Frame.		

Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06052 :		
Name	FrIfChannel {FRIF_CHANNEL_ID}		
Description	This parameter contains the FlexRay Channel used to transmit this FlexRay Frame.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRIF_CHANNEL_A	Channel A	
	FRIF_CHANNEL_AB	Channel A and B	
	FRIF_CHANNEL_B	Channel B	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06053 :		
Name	FrIfCycleRepetition {FRIF_CYCLE_REPETITION}		
Description	This parameter contains the FlexRay Cycle Repetition used to transmit this FlexRay Frame.. possible Values: 1,2,4,8,16,32,64		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 64		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06054 :		
Name	FrIfLSduLength {FRIF_LSDU_LENGTH}		
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	IntegerParamDef		
Range	0 .. 254		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: The parameter depends on the low level parameters of the FlexRay CC.		

SWS Item	FrIf06055 :		
Name	FrIfPayloadPreamble {FRIF_PAYLOADPREAMBLE_ID}		
Description	Switching the Payload Preamble bit.		
Multiplicity	1		

Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06056 :		
Name	FrIfSlotId {FRIF_SLOT_ID}		
Description	This parameter contains the FlexRay Slot ID used to transmit this FlexRay Frame.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 2047		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06048 :		
Name	FrIfFrameStructureRef {FRIF_FRAME_STRUCTURE}		
Description	Reference to the Construction Plan of the FlexRay Frame.		
Multiplicity	1		
Type	Reference to [FrIfFrameStructure]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

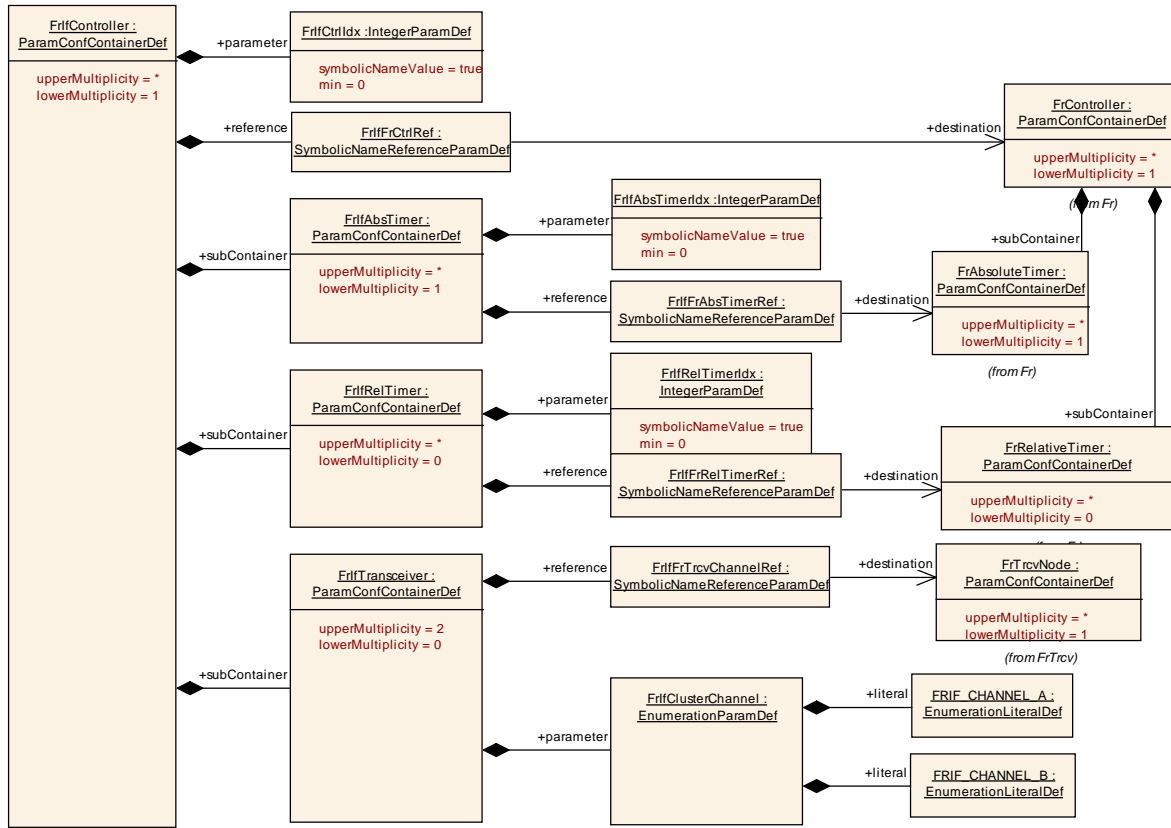


Figure 10-3: FrIfController

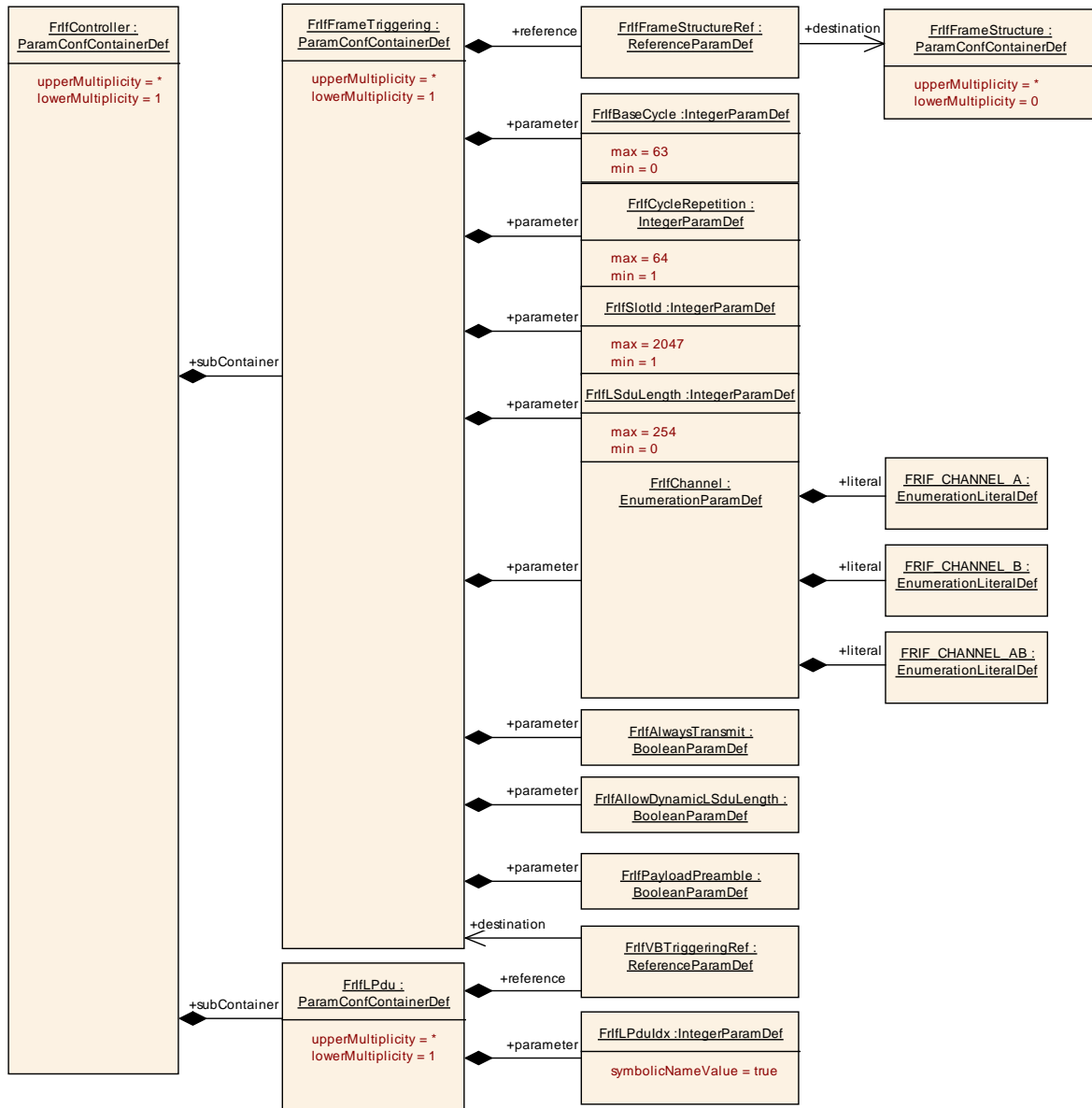


Figure 10-4: FrifController

10.2.9 FrIfJobList

SWS Item	FrIf05367 :
Container Name	FrIfJobList{FRIF_JOBLIST}
Description	This container specifies a list of all FlexRay Jobs of the Cluster to be performed by FrIf_JobListExec_<ClstIdx>().
Configuration Parameters	

SWS Item	FrIf06063 :		
Name	FrIfAbsTimerRef {FRIF_ABS_TIMER}		
Description	Reference to the absolute timer to be used to trigger the interrupt whose ISR contains the FrIf_JobListExec_<ClstIdx>() function.		
Multiplicity	1		
Type	Reference to [FrIfAbsTimer]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

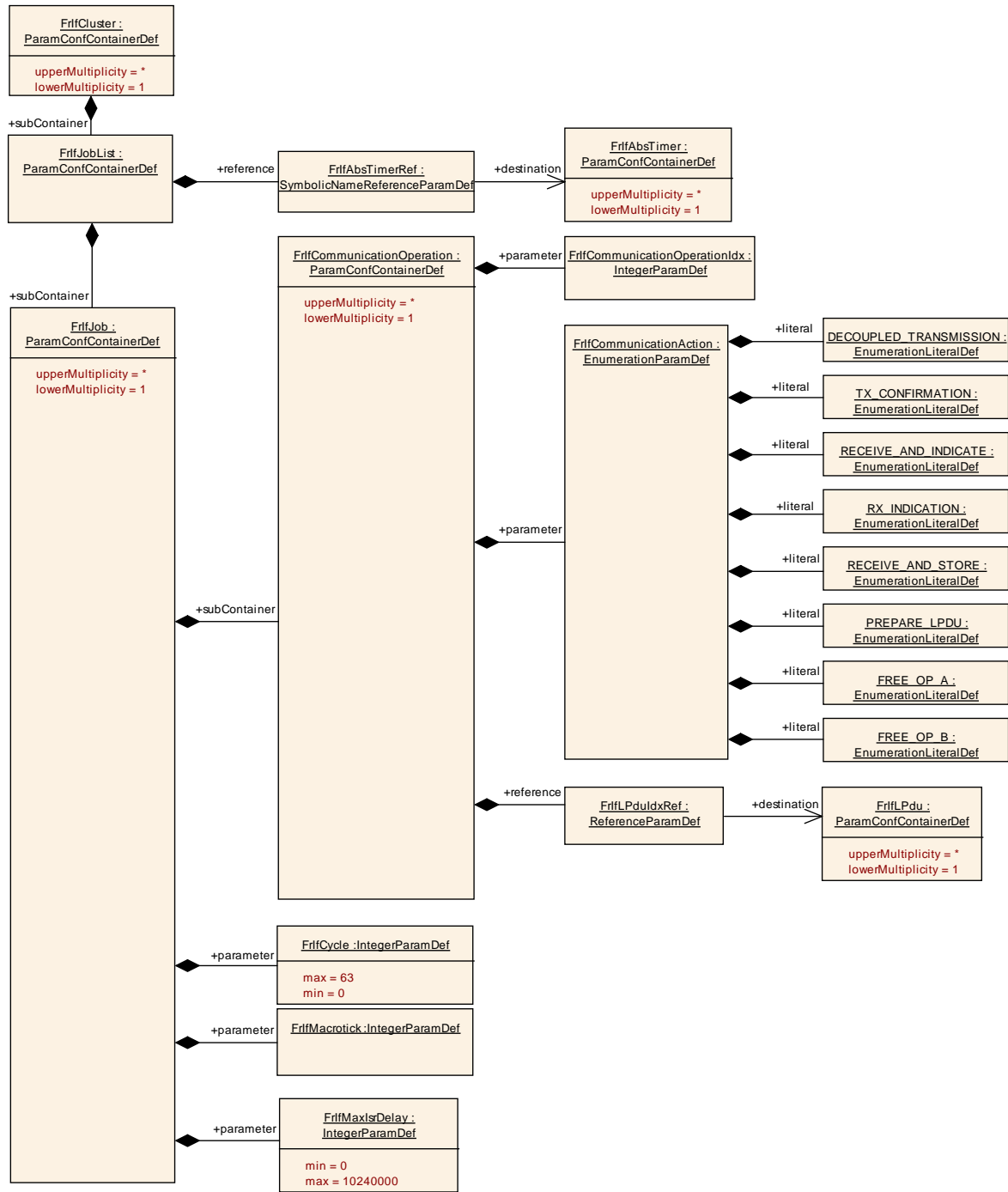


Figure 10-5: FrIfJobList

10.2.10 FrIfFrameStructure

SWS Item	FrIf05370 :
Container Name	FrIfFrameStructure{FRIF_FRAME_STRUCTURE}
Description	The Frame structure specifies a Construction Plan how a Frame is assembled with PDUs and their respective Update-Bits.
Configuration Parameters	

SWS Item	FrIf06113 :
Name	FrIfByteOrder {FRIF_BYTE_ORDER}
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 ByteOrder 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 FrlfPduOffset indicates the position of the least significant bit in the Frame.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BIG_ENDIAN		
	LITTLE_ENDIAN		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

10.2.11 FrlfPduInFrame

SWS Item	Frlf05371 :
Container Name	FrlfPduInFrame{FRIF_PDUS_IN_FRAME}
Description	This container holds all the information about a PDU in a FlexRay Frame.
Configuration Parameters	

SWS Item	Frlf06070 :		
Name	FrlfPduOffset {FRIF_PDU_OFFSET}		
Description	The value specifies the offset of the PDU within the Frame [bytes].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 253		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module 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.		

SWS Item	Frlf06071 :		
Name	FrlfPduUpdateBitOffset {FRIF_PDU_UPDATE_BIT_OFFSET}		
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	IntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module 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.
--	-----------------------

SWS Item	Frlf06069 :		
Name	FrFrlfPduRef {FRIF_PDU}		
Description	This is the reference to the local definition of a PDU.		
Multiplicity	1		
Type	Reference to [FrlfPdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.12 FrlfLPdu

SWS Item	Frlf05364 :		
Container Name	FrlfLPdu		
Description	LPdu is an abstraction of all FlexRay Frames (L-PDUs) belonging to the same Frame Triggering FrlfFrameTriggering.		
Configuration Parameters			

SWS Item	Frlf06058 :		
Name	FrlfLPduldx {FRIF_LPDU_IDX}		
Description	This parameter identifies the L-PDU in the interaction between FlexRay Interface and FlexRay Driver.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06057 :		
Name	FrlfVBTriggeringRef {FRIF_FRAME_TRIGGERING}		
Description	Reference to the assigned Frame triggering.		
Multiplicity	1		
Type	Reference to [FrlfFrameTriggering]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.13 FrlfPdu

SWS Item	Frlf05372 :		
Container Name	FrlfPdu{FRIF_PDU}		

Description	Contains PDU information. A PDU may be either a transmission PDU or a reception PDU.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / Dependency
FrlfPduDirection	1	A PDU is either transmit or receive

10.2.14 FrlfTxPdu

SWS Item	Frlf05374 :		
Container Name	FrlfTxPdu{FRIF_TX_PDU}		
Description	This container specifies transmission PDUs.		
Configuration Parameters			

SWS Item	Frlf06075 :		
Name	FrlfConfirm {FRIF_CONFIRM}		
Description	Defines whether the transmission of a PDU should be checked and confirmed to the PDU owning BSW module.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06076 :		
Name	FrlfCounterLimit {FRIF_COUNTER_LIMIT}		
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	1		
Type	IntegerParamDef		
Range	1 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06077 :		
Name	FrlfImmediate		
Description	Defines whether the the PDU is transmitted immediate or decoupled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf06050 :		
Name	FrlfNoneMode {FRIF_NONE_MODE}		
Description	Using the "None-Mode" which means that there is no API Frlf_Transmit call of the upper layer for this PDU.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	Frlf00014_Conf :		
Name	FrlfTxConfirmationName		
Description	This parameter defines the name of the <User_TxConfirmation>. This parameter depends on the parameter FrlfUserTxUL. If FrlfUserTxUL equals FR_TP, FR_NM, PDUR or XCP, the name of the <User_TxConfirmation> is fixed. If FrlfUserTxUL equals CDD, the name of the <User_TxConfirmation> is selectable.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Frlf06078 :		
Name	FrlfTxPduId {FRIF_TX_PDU_ID}		
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	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 ..		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Frlf and PduR/FrNm/FrTp		

SWS Item	Frlf06084_Conf :		
Name	FrlfUserTriggerTransmitName		
Description	This parameter defines the name of the <User_TriggerTransmit>. This parameter depends on the parameter FrlfUserTxUL. If FrlfUserTxUL equals FR_TP, FR_NM, PDUR or XCP, the name of the <User_TriggerTransmit> is fixed. If FrlfUserTxUL equals CDD, the name of the <User_TriggerTransmit> is selectable.		
Multiplicity	0..1		
Type	FunctionNameDef		
Default value	--		
regularExpression	--		

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

SWS Item Frlf00015_Conf :			
Name FrlfUserTxUL			
Description This parameter defines the upper layer (UL) module to which the trigger of the to be transmitted FRIFTXPDUID (via the <User_TriggerTransmit>) or the confirmation of the successfully transmitted FRIFTXPDUID has to be routed (via the <User_TxConfirmation>).			
Multiplicity 1			
Type EnumerationParamDef			
Range	CDD	Complex Device Driver	
	FR_NM	FR NM	
	FR_TP	FR TP	
	PDUR	PDU Router	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency scope: ECU			

SWS Item Frlf06074 :			
Name FrlfPduRef {FRIF_PDU_REF}			
Description Reference to the external PDU definition.			
Multiplicity 1			
Type Reference to [Pdu]			
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency scope: ECU			

No Included Containers

10.2.15 FrlfRxPdu

SWS Item Frlf05373 :	
Container Name	FrlfRxPdu{FRIF_RX_PDU}
Description	Receive PDU
Configuration Parameters	

SWS Item Frlf00016_Conf :	
Name FrlfRxIndicationName	
Description This parameter defines the name of the <User_RxIndication>. This parameter depends on the parameter FRIF_USERRXINDICATION_UL. If FRIF_USERRXINDICATION_UL equals FR_TP, FR_NM, or PDUR the name of the <User_RxIndication> is fixed. If FRIF_USERRXINDICATION_UL equals CDD, the name of the <User_RxIndication> is selectable.	
Multiplicity 0..1	
Type FunctionNameDef	
Default value --	

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

SWS Item	Frlf00017_Conf :		
Name	FrlfUserRxIndicationUL		
Description	This parameter defines the upper layer (UL) module to which the indication of the successfully received FRIFRXPDU has to be routed via <User_RxIndication>. This <User_RxIndication> has to be invoked when the indication of the configured FRIFRXPDU will be received by a Rx indication event from the FR Driver module. If no upper layer (UL) module is configured, no <User_RxIndication> has to be called in case of a Rx indication event of the FRIFRXPDU from the FR Driver module.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	CDD	Complex Device Driver	
	FR_NM	FR NM	
	FR_TP	FR TP	
	PDUR	PDU Router	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	Frlf06073 :		
Name	FrlfPduRef {FRIF_PDU_REF}		
Description	Reference to the external PDU definition.		
Multiplicity	1		
Type	Reference to [Pdu]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.16 FrlfPduDirection

SWS Item	Frlf06072 :
Choice container Name	FrlfPduDirection
Description	A PDU is either transmit or receive

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

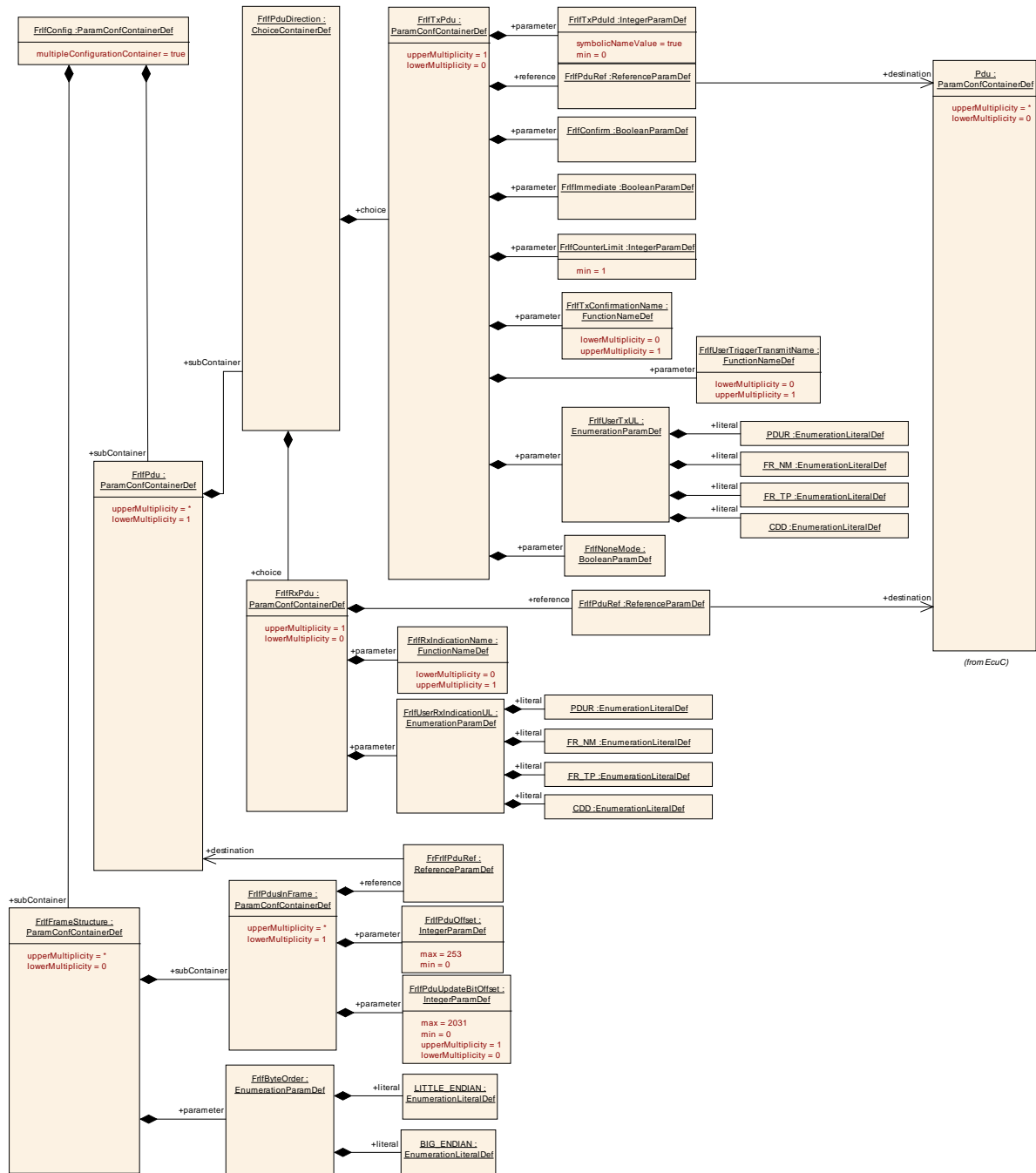


Figure 10-6: FrIfPdu

10.2.17 FrIfTransceiver

SWS Item	FrIf05391 :
Container Name	FrIfTransceiver{FRIF_TRANSCEIVER}
Description	Up to two FlexRay Transceivers may connect a Controller to a Cluster. This container realizes a Controller-Transceiver assignment.
Configuration Parameters	

SWS Item	FrIf06062 :
Name	FrIfClusterChannel {FRIF_TRCV_CLUSTER_CHANNEL}

Description	This parameter identifies to which one of the two Channels "A" or "B" of the Cluster the Transceiver is connected.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRIF_CHANNEL_A	Channel A	
	FRIF_CHANNEL_B	Channel B	
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	Frlf06061 :		
Name	FrlfFrTrcvChannelRef {FRIF_TRCV_CHANNEL}		
Description	Reference to a Transceiver Driver Channel. This reference is unique for the ECU.		
Multiplicity	1		
Type	Reference to [FrTrcvNode]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.18 FrlfConfig

SWS Item	Frlf06001 :
Container Name	FrlfConfig [Multi Config Container]
Description	Configuration of the FlexRay Interface. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

Included Containers		
Container Name	Multiplicity	Scope / 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	0..*	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.

10.3 Published Information

Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information. In addition, the published information contains some [link time](#) configuration data that is needed by the configuration tools.

The standard common published information like

```
vendorId (<Module>_VENDOR_ID),  
moduleId (<Module>_MODULE_ID),  
arMajorVersion (<Module>_AR_MAJOR_VERSION),  
arMinorVersion (<Module>_AR_MINOR_VERSION),  
arPatchVersion (<Module>_AR_PATCH_VERSION),  
swMajorVersion (<Module>_SW_MAJOR_VERSION),  
swMinorVersion (<Module>_SW_MINOR_VERSION),  
swPatchVersion (<Module>_SW_PATCH_VERSION),  
vendorApiInfix (<Module>_VENDOR_API_INFIX)
```

is provided in the BSW Module Description Template (see [15] Figure 4.1 and Figure 7.1).

Additional published parameters are listed below if applicable for this module.