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

Document Version	2.3.0
Document Status	Draft
Part of Release	2.1
Revision	20

Document Change History

Date	Version	Changed by	Change Description
07.06.2010	2.3.0	AUTOSAR Administration	• Provide the Cycle length in nanoseconds directly to avoid systematic rounding error • If the Joblist Execution Function loses synchronization it has to be resynchronized with the next mainfunction call • Legal disclaimer revised
17.04.2008	2.2.0	AUTOSAR Administration	• 7.4.1: Added description • 7.4.3.1: Added description • 10.2.2.1: Added parameter „FrIfUnusedBitValue“ • 10.2.2.2.1.5: Added parameter “FrIfAllowDynamicLSduLength • Legal disclaimer revised
06.02.2008	2.1.0	AUTOSAR Administration	Harmonization improved to Communication Manager and FlexRay Driver
06.02.2007	2.0.1	AUTOSAR Administration	• “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

Release Notes

Errata and known deficiencies

The wakeup concept is currently neither harmonized nor consistent throughout all wakeup related specification documents and therefore subject to change.

Known and potential problems resulting from known deficiencies

Due to the fact that the wakeup concept is not harmonized, inconsistent assumptions may lead to

- the duplication of functionalities across multiple modules
- proprietary implementation extensions
- difficulties during integration

Changes planned for next release

The harmonized wakeup concept throughout all wakeup related documents will result in

- adapted specification texts in Chapter 7 of the specification documents
- adapted APIs in Chapter 8 of the specification documents
- adapted wakeup sequences in Chapter 9 of the specification documents

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and Functional Overview

This specification 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).

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

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

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

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

- initialization
- data transmission (sending and reception)
- start/halt/abort communication
- FlexRay specific functions (e.g. send wake-up pattern)
- set operation mode
- get status information
- various timer functions

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

In addition to the above-mentioned AUTOSAR BSW modules, the FrIf shall, with the functionality described within the specification in hand, also support other non-AUTOSAR upper layer software modules, provided that these modules interact with the FrIf in the same manner as the upper layer AUTOSAR BSW modules. In particular, those non-AUTOSAR modules need to provide APIs according to chapter 8.7.3 of this specification. 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)
SchM	BSW Scheduler (AUTOSAR BSW module)
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** *which operations* this executable code performs on *which data* and at *which points in time*.

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

3 Related Documentation

3.1 Input Documents

[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

3.2 Related Standards and Norms

[14] FlexRay Communications System Protocol Specification Version 2.1
Revision A

[15] FlexRay Communications System Electrical Physical Layer Specification Version 2.1 Revision A

4 Constraints and Assumptions

4.1 Limitations

Several API services of the FlexRay BSW modules allow for selecting one of possibly **multiple configuration sets**, e.g. FrIf_Init(). However, for the current release, it is only necessary for the implementation of the FlexRay BSW modules and their respective configuration tools to support **one configuration set** for each possible multitude. Nevertheless, the respective API services are already specified to support multiple configuration sets in order to avoid a change of the API services' signatures in future releases of this specification.

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.

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

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

Cycle Number = $(B + n * 2^R)_{mod64}$

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

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

In the dynamic segment of each FlexRay Communication Cycle, a transmit/receive buffer of a FlexRay Communication Controller shall be used at the most once. This limits the reconfiguration possibilities and thus restricts the number of transmittable (sent and received) LPdus per dynamic segment to the accumulated number (over all CCs on one ECU) of transmit/receive buffers connected to one cluster. This limitation results from the unpredictability of the time of transmission of an LPdu within the dynamic segment. Because of that a point in time for the reconfiguration of a certain buffer for multiple usages within the dynamic segment cannot be predetermined.

4.2 Applicability to Car Domains

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

5 Dependencies to Other Modules

5.1 AUTOSAR Operating System

There is one dedicated FlexRay Job List Execution Function for each FlexRay Cluster that is controlled by the FlexRay Interface. Each FlexRay Job List Execution Function must be registered in the AUTOSAR OS as the ISR of an absolute timer of a FlexRay CC connected to the respective Cluster.

5.2 AUTOSAR BSW Scheduler

There is one dedicated FlexRay Interface Main Function for each FlexRay Cluster that is controlled by the FlexRay Interface. 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 according to the Cluster's Cycle length.

5.3 AUTOSAR Communication Manager

By means of some dedicated API services, the AUTOSAR ComM controls the communication state of the FlexRay Clusters as defined by the FlexRay Interface State Machine.

5.4 All Upper Layer AUTOSAR BSW Modules

The calling of the FlexRay Job List Execution Function synchronously to the FlexRay Global Time ensures 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. 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 has to allow access to its PDU buffers at all times and it also has to ensure data consistency in its buffers.

5.5 AUTOSAR PDU-Router

The FrIf declares and calls some callback functions of the PDU-Router in order to confirm transmission and notify reception of PDUs.

5.6 AUTOSAR FlexRay Network Management

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

5.7 AUTOSAR FlexRay Transport Protocol

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

5.8 AUTOSAR FlexRay Driver

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

5.9 AUTOSAR FlexRay Transceiver Driver

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

5.10 AUTOSAR Development Error Tracer

In order to be able to report development errors, the FrIf has to have access to the error hook of the Development Error Tracer.

5.11 AUTOSAR Diagnostic Event Manager

In order to be able to report production errors, the FrIf has to have access to the Diagnostic Event Manager.

5.12 File Structure

5.12.1 Code File Structure

The code file structure shall not completely be defined within this specification.
However, the code-file structure shall include the following files:

FrIf.c	general source code file of the FlexRay Interface
FrIf_Cfg.c	contains pre-compile time configurable parameters
FrIf_Lcfg.c	contains link time configurable parameters
FrIf_PBcfg.c	contains post build time configurable parameters

5.12.2 Header File Structure

The header file structure shall contain the following header files:

FrIf.h	general header file of the FlexRay Interface
FrIf_Cfg.h	contains the FrIf pre-compile-time configurable parameters (pre processor constants)
FrIf_Cbk.h	contains the declarations of the callback functions provided by the FrIf 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_GeneralTypes.h	contains declarations shared by all AUTOSAR FlexRay BSW modules
ComStack_Types.h	contains the communication module abstracted datatypes shared by AUTOSAR communication BSW.
PduR_FrIf.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_FrIf.h	Contains the declaration of the API services the SchM offers to the FlexRay Interface
MemMap.h	

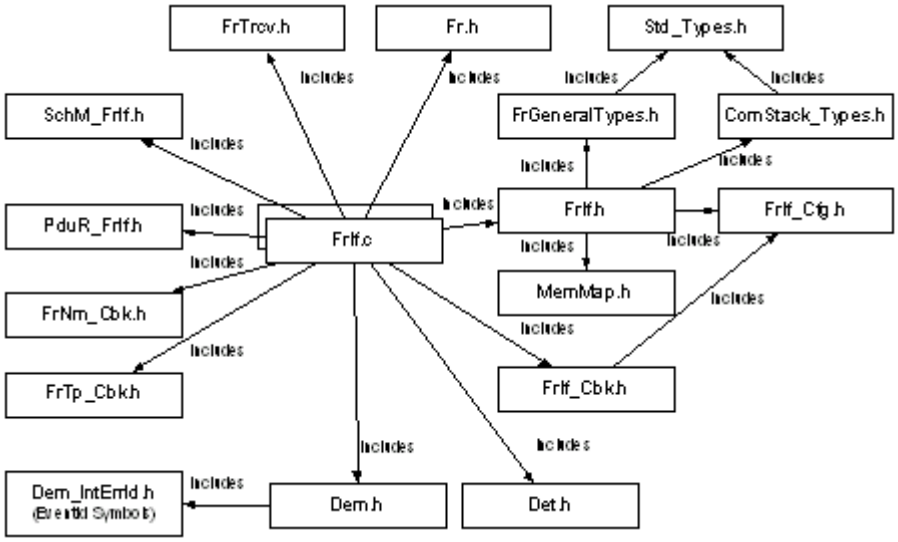

Figure 5-1: FlexRay Interface Header File Structure

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.

6 Requirements Traceability

6.1 General Requirements on Basic Software Modules

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

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

BSW00371	Do not pass function pointers via API	
BSW00358	Return type of init() functions	
BSW00414	Parameter of init function	
BSW00376	Return type and parameters of main processing functions	
BSW00359	Return type of callback functions	
BSW00360	Parameters of callback functions	
BSW00329	Avoidance of generic interfaces	
BSW00330	Usage of macros / inline functions instead of functions	
BSW00331	Separation of error and status values	
BSW009	Module User Documentation	
BSW00401	Documentation of multiple instances of configuration parameters	
BSW172	Compatibility and documentation of scheduling strategy	
BSW010	Memory resource documentation	
BSW00333	Documentation of callback function context	
BSW00374	Module vendor identification	
BSW00379	Module identification	
BSW003	Version identification	
BSW00318	Format of module version numbers	
BSW00321	Enumeration of module version numbers	
BSW00341	Microcontroller compatibility documentation	
BSW00334	Provision of XML file	
BSW00435	Header File Structure for the Basic Software Scheduler	
BSW00436	Module Header File Structure for the Memory Mapping	

6.2 Requirements on FlexRay

Requirement		***Satisfied by***
BSW05000	Support of Synchronous SW Modules	5.3, 7.4.4
BSW05001	Support of Asynchronous SW Modules	5.3, 7.4.3.1
BSW05002	FlexRay Interface and FlexRay Driver as Only Necessarily Synchronous SW Modules	5.3
BSW05003	Support of Slot/Cycle Multiplexing	4.1
BSW05169	Avoid Timer Interrupts during Start-up	8.3.2
BSW05055	Avoid Timer Interrupts during Shutdown	8.3.8, 8.3.9
BSW05064	Abstraction of FlexRay CC-specific Implementation	7.2.1
BSW05065	Number of FlexRay CCs per Driver	7.2.1
BSW05005	Support of Hardware FIFO Mechanism	7.4.3.3, 7.4.3.4, 7.4.3.5
BSW05024	Abstraction from CC Buffer Configuration	10.2.2.2.1.4
BSW05066	L-SDU-Based API	8.7.3.3
BSW05058	Configuration of FlexRay Driver at System Configuration Time	7.4
BSW05059	Transmit/Receive Buffer Configuration	7.4, 10.2.2.2.1.4
BSW05116	Initialization of FlexRay CC	7.3, 7.3.1, 8.3.3
BSW05011	Initialize Low-Level Parameters	8.3.2, 8.3.3
BSW05012	Initialize FlexRay CC Transmit/Receive Buffers	8.3.3
BSW05109	Start-up of a FlexRay CC	7.3.1, 8.3.7
BSW05117	Sending of Wake-Up Pattern	8.3.15
BSW05120	Get FlexRay CC POC Status	8.3.18
BSW05121	Get FlexRay CC Sync State	8.3.16
BSW05106	Buffer Reconfiguration in Normal Active Mode	7.4.3.6
BSW05107	MTS Sending	8.3.4
BSW05111	Get MTS Reception Status	8.3.6
BSW05125	Interrupt Handling	8.3.26, 8.3.27, 8.3.32, 8.3.33

BSW05114	Abortion of FlexRay CC Communication	8.3.9
BSW05115	Halt of FlexRay CC Communication	8.3.8
BSW05033	Tick Conversion	8.3.21, 8.3.22
BSW05156	Controller External Clock Synchronization	8.3.17
BSW05044	Set Absolute Timer	8.3.23
BSW05045	Set Relative Timer	8.3.24
BSW05046	Enable Absolute Alarms	8.3.27
BSW05047	Disable Absolute Alarms	8.3.33
BSW05048	Acknowledge Absolute Alarms	8.3.31
BSW05049	Enable Relative Alarms	8.3.28
BSW05050	Disable Relative Alarms	
BSW05051	Acknowledge Relative Alarms	8.3.32
BSW05052	Get Cycle Length in Macroticks	8.3.20
BSW05019	Get FlexRay Global Time	8.3.19
BSW05072	FlexRay Time Services Access if CC is Out of Sync	8.3.19

Remarks:

(1) This is not a requirement for the FlexRay Interface, but for the FlexRay Driver.
(2) This is not a requirement for the FlexRay Interface, but for the FlexRay Transceiver Driver.
(3) This is not a requirement for the FlexRay Interface, but for the FlexRay Transport Layer.

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 FrIf accesses several CCs using the FlexRay Driver layer, which can be made up of several FlexRay Driver modules. The FlexRay Transceiver modules are not shown in this figure; however, the structure that applies to the FlexRay Driver modules and the FlexRay CCs analogously applies to the FlexRay Transceiver Driver modules and the FlexRay Transceivers.

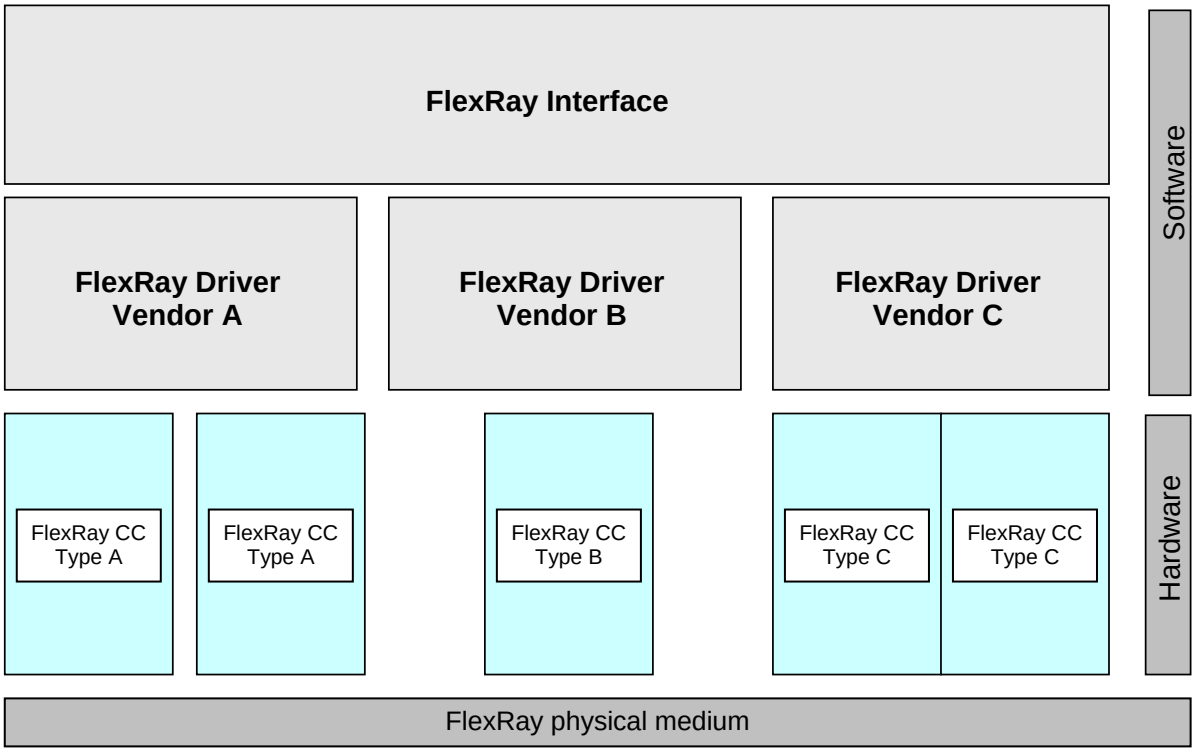

Figure 7-1: Basic Structure of the FlexRay BSW Stack

7.2 Indexing Scheme

7.2.1 Principle

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

In order to select those resources spread over the various entities[2] accessed via the FrIf, the FlexRay-related AUTOSAR BSW modules use an indexing scheme that is exemplarily described in Figure 7-2 and Figure 7-3.

[1] E.g. controller, timers, configuration data sets, etc.

The FrIf achieves the abstraction by providing to the upper layer BSW modules an abstract, unique, zero-based consecutive index [3] for each sort of resource, independent from their type, location, and access method.
In general, the FrIf 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 FrIf[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.

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

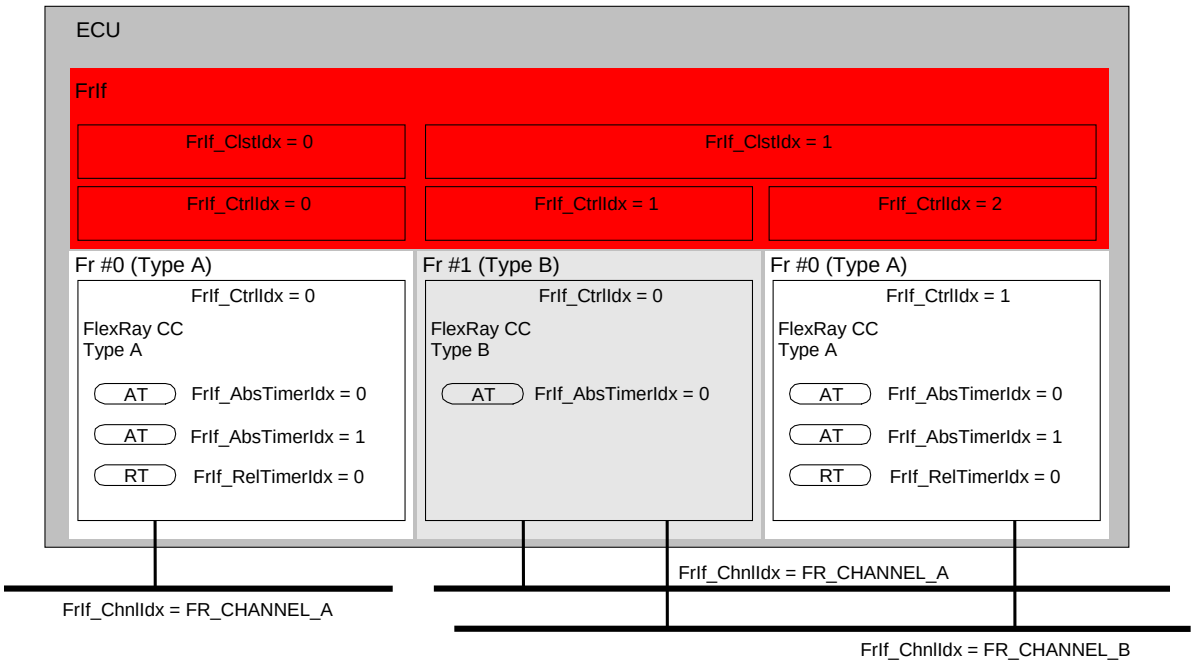

Figure 7-2: FlexRay Controller Indexing Scheme of the FlexRay Interface

In order to abstract for upper layer BSW modules the various CCs which the FrIf controls via the FlexRay Drivers, the FrIf generates an abstract, unique, zero-based consecutive index FrIf_CtrlIdx (“FlexRay Interface Controller Index”), which maps to a tuple of FlexRay Driver API Service function pointer and CC index Fr_CtrlIdx (“FlexRay Driver Controller Index”).

[2] FlexRay Drivers, FlexRay Communication Controllers, FlexRay Transceiver Drivers, and FlexRay Transceivers
[3] Like Controller Index, Cluster Index, Channel Index, 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.

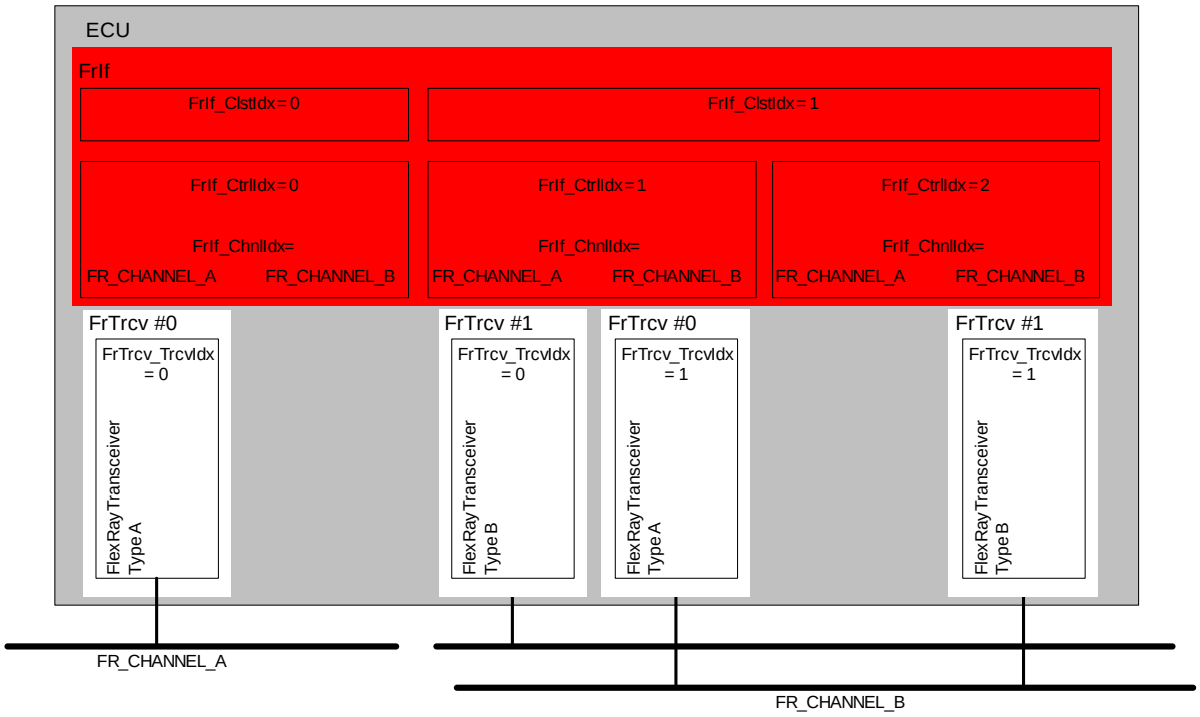

Figure 7-3: FlexRay Transceiver Indexing Scheme of the FlexRay Interface

In order to abstract for upper layer BSW modules the various FlexRay Transceivers which the FrIf controls via the FlexRay Transceiver Drivers, the FrIf takes advantage of the fact that each FlexRay Transceiver is unambiguously assigned to a specific Channel on a specific FlexRay CC.

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

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

The FlexRay Clusters are numbered with the index FrIf_ClstIdx ("FlexRay Interface Cluster Index"). In order to provide Cluster-based API services, the static configuration data of the FrIf contains a data structure that specifies which FlexRay CC and which FlexRay Transceivers are connected to each Cluster, or in other words, that maps each value of FrIf_ClstIdx to (one, or in general) a set of values for FrIf_CtrlIdx and tuples of (FrIf_CtrlIIdx, FrIf_ChnlIdx).

The FrIf numbers all PDUs to be transmitted with an abstract, unique, zero-based consecutive index FrIf_TxPduId (“FlexRay Interface Transmit PDU ID”). This index is used in the FrIf API service FrIf_Transmit() and allows the FrIf 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

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

The FrIf can be configured to support at least four (possibly different) **FlexRay CCs**.

The FrIf can be configured to support one of both or both **FlexRay Channels** as specified in [14].

The FrIf can be configured to support at least four **FlexRay Clusters**.

The FrIf can be configured to support at least one **absolute timer** per FlexRay CCs.

The FrIf can be configured to support at least one **relative timer** per FlexRay CCs.

7.3 FlexRay Interface State Machine

In order to allow ComM to control the communication states of the FlexRay system, the FrIf implements a simplified state machine called **FlexRay Interface State Machine,** and provides the APIs necessary for starting up und shutting down the FlexRay communication. This simplified state machine hides from the upper layers the FlexRay-specific complex mechanisms carried out during communication start-up and communication shutdown, and the **substates** passed through during these processes.

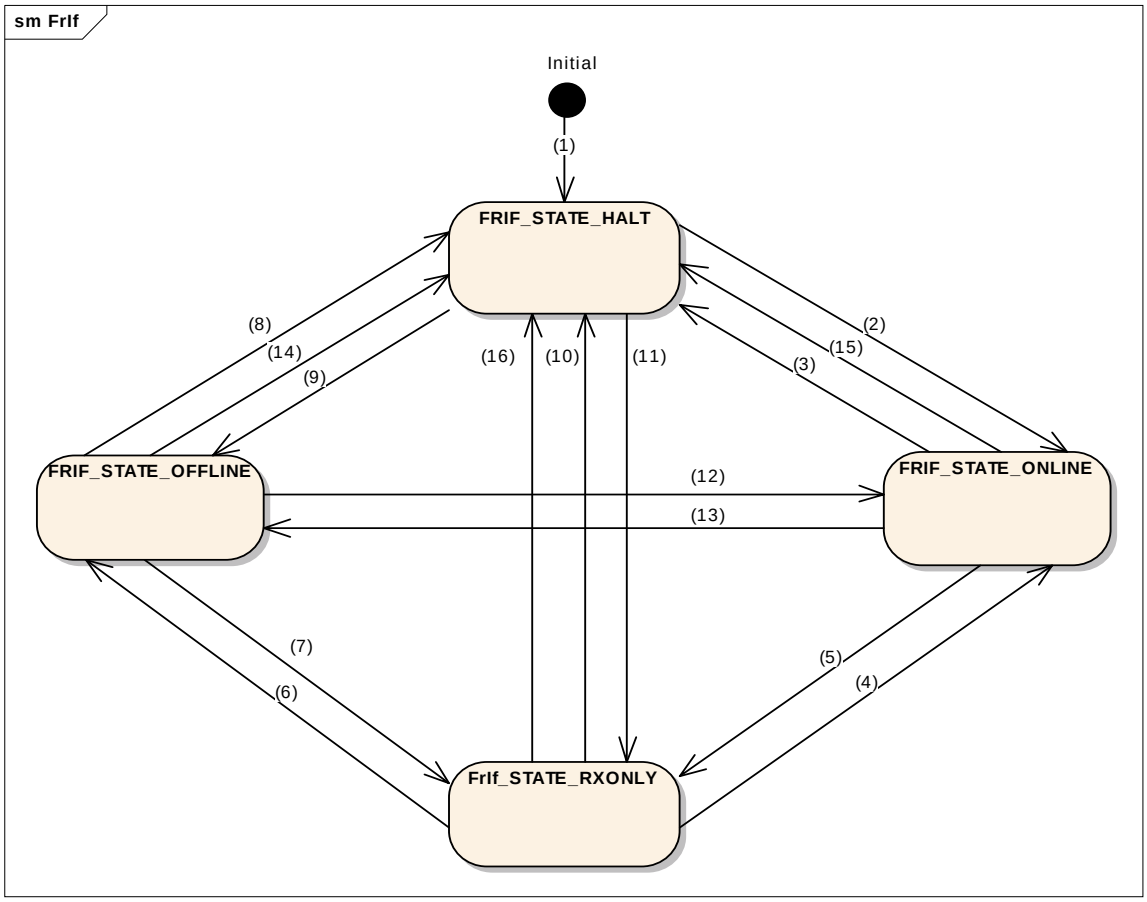

Figure 7-4: FlexRay Interface State Machine

Figure 7-4: FlexRay Interface State Machine shows the states and transistions that are visible to the user of the FrIf.
The four different states represent the communication capabilities of the FrIf.

For each CC controlled by the FrIf:

- a separate FlexRay Interface State Machine, and
- a separate state varialble **FrIf_State**

is maintained by the FrIf.

The state varialble FrIf_State can assume the values given in Table 7-1.

State	*Description*
FRIF_STATE_HALT	The CC is halted and no communication operations are executed (see chapter 7.4 for details).
FRIF_STATE_OFFLINE	The CC is synchronized to the cluster, but no communication services are executed (see chapter 7.4 for details).
FRIF_STATE_RXONLY	The CC is synchronized to the cluster and only reception services are executed (see chapter 7.4 for details).
FRIF_STATE_ONLINE	The CC is synchronized to the cluster and all communication services (reception, transmission, transmission confirmation) are executed (see chapter 7.4 for details).

Table 7-1: FrIf_State Values

7.3.1 Controlling State Transitions

During initialization of the FrIf by executing FrIf_Init() (transistion (1)), all CCs are halted (set into *POC:halt*) and the FrIf_State for each CC is initialized with state FRIF_STATE_HALT.

All other state transistions shown in Figure 7-4 are requested by API services FrIf_RequestControllerStateTransition(). The current value of FrIf_State can be retrieved by the API service FrIf_GetControllerState() and does also affect the execution of the various Communication Operations on the respective CC. Further details can be found in chapter 7.4.3.

A cluster-based view on the FlexRay Interface State Machine is provided by the API services FrIf_RequestClusterStateTransition() and FrIf_GetClusterState() as described in chapters 8.3.11 and 8.3.13.

The following table lists the transition names that can be used as parameter when calling FrIf_RequestControllerStateTransition() and FrIf_RequestClusterStateTransition(). When using the Cluster-based API service FrIf_RequestClusterStateTransition(), the state of all FlexRay CCs belonging to the respective Cluster is controlled.
Please note that certain sets of transistions are combined into a single transistion name, since they result in the same target state, even if they originate from different states.

Transistion Name	*Transitions (see* Figure 7-4*)*	*Description*
FRIF_GOTO_OFFLINE	(9)	Asynchronous transistion resulting in FrIf_State FRIF_STATE_OFFLINE.
	(6) (13)	This is a software state transistion that can be executed synchronously within FrIf_RequestControllerStateTransition().
FRIF_GOTO_RXONLY	(11)	Asynchronous transistion resulting in FrIf_State FRIF_STATE_RXONLY.
	(5) (7)	This is a software state transistion that can be executed synchronously within FrIf_RequestControllerStateTransition().
FRIF_GOTO_ONLINE	(2)	Asynchronous transistion resulting in FrIf_State FRIF_STATE_ONLINE.
	(4) (12)	This is a software state transistion that can be executed synchronously within FrIf_RequestControllerStateTransition().
FRIF_GOTO_HALT_IMMEDIATELY	(3) (8) (10)	This is a synchronous state transistion that immediatly halts the FlexRay communication by calling FrIf_AbortCommunication(), which also sets the FrIf_State to FRIF_STATE_HALT.
FRIF_GOTO_HALT_AT_EOC_NO_COM	(14) (15) (16)	The FlexRay communication is halted at the end of the current communication cylce by calling FrIf_HaltCommunication(), which also sets the FrIf_State to FRIF_STATE_HALT. The software transistion to FRIF_STATE_HALT is executed synchonously within FrIf_RequestControllerStateTransition(), i.e. the communication operation are stopped immediately.

Table 7-2: FlexRay Interface State Machine Transitions

Besides the above-mentioned API services used to control the state transitions of the FlexRay Interface State Machine, the FrIf provides other API services that can be used to directly control the POC state of a FlexRay CC, and thus the communication states of the FlexRay system.
However, please note that the user of the FrIf may **either** use the API services

- FrIf_RequestControllerStateTransition()
- FrIf_RequestClusterStateTransition()
- FrIf_GetControllerState()
- FrIf_GetClusterState()

or the API services

- FrIf_ControllerInit()
- FrIf_StartCommunication()
- FrIf_HaltCommunication()
- FrIf_AbortCommunication()
- FrIf_SetWakeupChannel()
- FrIf_SendWUP()
- FrIf_GetSyncState()
- FrIf_GetPOCStatus()

to control the communication states of the FlexRay system.
Using both sets of API services concurrently will result in undefined behavior of the FlexRay system.

For starting up the FlexRay communication, several tasks have to be performed inside the FlexRay system. Since those tasks in general need a relative long time period to execute, the start-up transitions (2), (9) and (11) are **asynchronous**, i.e. they are generally not completed within the execution of the respective API service call FrIf_RequestControllerStateTransition() or FrIf_RequestClusterStateTransition(),, and the requested target state will therefore not be reached immediately.
In order to ensure that the requested asynchronous state transitions are completed, a mechanism has to be implemented within the FlexRay Interface Main Function that brings the respective state transition to completion. Further details can be found in chapter 7.3.2.

7.3.2 FlexRay Interface Main Function

The FlexRay Interface Main Function needs to be called cyclically from a task body provided by the SchM with a calling period according to the FlexRay Cycle length and configurable at system configuration time (configuration parameter FrIfMainFunctionCycle). The initialization API service of the FrIf, ensures that the FlexRay Interface Main Function is cyclically called by SchM with the configured period.

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, there is one dedicated FlexRay Interface Main Function for each FlexRay Cluster that is controlled by the FrIf.
The API names of the FlexRay Interface Main Functions therefore are:

- FrIf_MainFunction_0() for Cluster # 0 (FrIf_ClstIdx = 0)
- FrIf_MainFunction_1() for Cluster # 1 (FrIf_ClstIdx = 1)
- FrIf_MainFunction_2() for Cluster # 2 (FrIf_ClstIdx = 2)
- FrIf_MainFunction_3() for Cluster # 3 (FrIf_ClstIdx = 3)
- and so on, if more than 4 FlexRay Clusters are supported.

Upon each invocation, the FlexRay Interface Main Function performs for its respective Cluster (or all the CCs belonging to this Cluster, respectively) the following **major tasks**:

- If any of the asynchronous state transistions (2), (9) or (11) of the FlexRay Interface State Machine has been requested for a CC, drive the respective state transition.
- If necessary, update FrIf_State for each CC from the actual hardware state of the respective CC.
- Check, if a loss of the JobList's synchronization (see JobListAsyncFlag) or a miss of execution was detected and if necessary (re-)establish the synchronous execution of the FlexRay Job List Execution Function.

The following chapters provide more details about these major tasks of the FlexRay Interface Main Function.

7.3.2.1 Driving the Asynchronous State Transistions

The state transistions (2), (9) or (11) of the FlexRay Interface State Machine occur during the start-up of a FlexRay CC or Cluster. During these transitions, a FlexRay CC is taken from *POC:halt* to an operational state (*POC:normal active* or *POC:normal passive*) via a multitude of substates.
The exact behaviour of the FlexRay system during these start-up transistions heavily depends on the intended use case and its constraints. However, the number of conceivable use cases and their respective constrains is very high. Therefore, in this document it is impossible to provide a detailed and all-embracing description of how the asynchronous transitions (2), (9) and (11) are realized within the FlexRay Interface State Machine including all the substates a FlexRay CC has to assume during these transitions. It lies within the responsibility of the implementer of the FrIf to define and implement a behavior that satisfies all relevant use cases, and offers the configuration possibilities which allow the user of the FrIf to adapt the FrIf's behavior to his/her needs. This includes e.g.

- the option to transmit or not transmit a wake-up pattern before executing the start-up,
- the option to act as coldstart node or as integrating node,
- the option to use or not use single slot mode,
- etc.

The following list serves as guideline to the implementer on the tasks that might have to be performed in order to drive the asynchronous state transitions. Please note that this sequence of tasks is **not** completely executed within **one call** of the FlexRay Interface Main Function, since it contains asynchronous API service calls. Upon each invocation, the FlexRay Interface Main Function has to monitor the progress in this sequence of tasks and apply adequate means to advance to the next step at the appropriate point in time.

1. Initialize the FlexRay CC (use FrIf_ControllerInit()).
2. Optionally transmit a wakeup pattern (use FrIf_SetWakeupChannel() and FrIf_SendWUP()).
3. Start FlexRay Communication (use FrIf_StartCommunication()).
4. Wait until FlexRay Communication is established (use FrIf_GetSyncState() or FrIf_GetPOCStatus()).
5. Setup the Job List Execution: Set the flag JobListAsyncFlag indicating to the FlexRay Interface Main Function that the execution of the FlexRay Job List of this Cluster needs to be synchronized.

Further tasks are necessary to allow for the above-mentioned optional behavior, e.g. single slot mode. For details on the FlexRay Communications System's start-up mechnanism, please refer to [14].

7.3.2.2 Updating FrIf_State

Since a FlexRay CC might autonomously perform transistions (3), (8) and (10) of the FlexRay Interface State Machine due to a loss of synchronicity[5] with the FlexRay Cluster, the FrIf_State of each CC might have to be updated from the actual hardware state of the respective CC, which can be retrieved by means of the API service FrIf_GetPOCStatus().

7.3.2.3 Synchronizing the FlexRay Job List Execution Function

If the flag JobListAsyncFlag indicates that the execution of the FlexRay Job List by the FlexRay Job List Execution Function of the Cluster needs to be synchronized, the FlexRay Interface Main Function (re-)establishes the synchronicity by performing the following steps:

1. Get the current FlexRay Global Time (by calling FrIf_GetGlobalTime()) and add some "safety margin".
2. Search the Cluster's FlexRay Job List for the subsequent FlexRay Communication Job, i.e. the first Job with an invocation time greater than the current Global Time + safety margin. If the search has reached the end of the FlexRay Job List, continue the search at the beginning of the FlexRay Job List.
3. Set the JobListPointer to that FlexRay Communication Job and program the absolute timer used for this Cluster's FlexRay Job List Execution Function (configuration parameter FrIfAbsTimerRef) with this Job's invocation time by calling FrIf_SetAbsoluteTimer().

[5] Note that both *POC:normal active* and *POC:normal passive* are considered synchronous states.

4. Clear the JobListAsyncFlag to indicate that this Cluster's FlexRay Job List Execution Function has been (re-)synchronized to the FlexRay Global Time.
5. Enabled the interrupt of the absolute timer used for this Cluster's FlexRay Job List Execution Function (by calling FrIf_EnableAbsoluteTimerIRQ())

This mechanism is used for the initial synchronization of the FlexRay Job List Execution Function (i.e. after the initialization of the FrIf by means of FrIf_Init()) as well as any subsequent re-synchronizations that might become necessary if the FlexRay Job List Execution Function loses synchronicity with the FlexRay Global Time of the Cluster.

7.4 Data Communication via FlexRay

FlexRay in general is a deterministic time-driven communication system. Each datum that should be transmitted or received has to be scheduled at system configuration time. This even holds true for data that - from the application's point of view - are considered *event-driven*.

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

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).
In other words, 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.

7.4.1 PDU Packing, Frame Construction Plans, and PDU Update-Bits

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

Since bus-independent AUTOSAR PDUs have a maximal length of 8 bytes, and since according to [14] a FlexRay Frame can contain as many as 254 bytes of payload data, the FrIf is capable of packing multiple PDUs into one FlexRay Frame. The rules defining how to pack PDUs into FlexRay Frames, the so-called **Frame Construction Plans**, are defined at system configuration time and stored in the static configuration of the FrIf (configuration parameter FrIfFrameStructure).

In cases where multiple PDUs are packed into a single FlexRay Frame, this FlexRay Frame has to be transmitted by the FrIf if **at least one** of the contained PDUs has been updated by an upper layer BSW module (by a call of FrIf_Transmit() and <UL_TriggerTransmit>()). As a result, those PDUs in this FlexRay Frame that have not been updated by an upper layer BSW module will also be transmitted. Please note that this does not necessarily mean that the previous values of those PDUs are transmitted. On the contrary, in case the parameter 'FrIfUnusedBitValue' does not exist, arbitrary values for those PDUs will be transmitted.

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

In order for the receiving FrIf to 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, additional information, so called **PDU Update-Bits**, within the FlexRay Frame may be transmitted to the receiving FrIf.

For each PDU, a dedicated PDU Update-Bit in the FlexRay Frame can be configured. This PDU Update-Bit can be located at an arbitrary bit position in the Frame Construction Plan that is not occupied by any PDU.
The configuration of Update-Bits for the PDUs and the definition of the location of the Update-Bits within the FlexRay Frame are performed at system configuration time (configuration parameter FrIfPduUpdateBitOffset).

If an Update-Bit is configured for a specific PDU, the receiving FrIf will only indicate the reception of this PDU to an upper layer BSW module, if the PDU's Update-Bit in the Frame is set to "1". If the PDU's Update-Bit is set to "0", no reception will be indicated. If no Update-Bit is configured for a specific PDU, the receiving FrIf assumes this PDU to always be valid, and this PDU's receiption is always indicated to the upper layer BSW module.

If the parameter 'FrIfUnusedBitValue' 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 'FrIfUnusedBitValue'.

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

According to [14], a FlexRay CC might **only** support the so-called "continuous transmission mode" where a message is transmitted continuously until the host explicitly invalidates the transmit buffer. If such a FlexRay CC is being used for transmission, and the receiving FrIf should still be able to determine which of the PDUs in a received FlexRay Frame have actually been updated by an upper layer BSW module on the transmitter side, a special mechanism is needed in the transmitting FrIf, called **AlwaysTransmit** (configuration parameter FrIfAlwaysTransmit). If AlwaysTransmit is enabled for an L-PDU that is transmitted using the Communication Operation DECOUPLED_TRANSMISSION, the FlexRay Driver's API service Fr_TransmitTxLPdu() is always called for this L-PDU, independent from any PDUs in this L-PDU having been updated by an upper layer BSW module. This enables resetting the PDU Update-Bits in the FlexRay CC's transmit buffer, even if none of the PDUs in the FlexRay Frame have actually been updated by an upper layer BSW module, and thus ensures the correct interpretation of the received Frame contents by the receiving FrIf.

If Transmission with Immediate Buffer Access is used for a FlexRay Frame, only one PDU is allowed in this Frame. Therefore, PDU Update-Bits can in most cases be omitted for Transmission with Immediate Buffer Access.

Since:

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

- Update-Bits can be configured independently per PDU,

the FrIf 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.

7.4.2 Realization of the Time-Driven FlexRay Schedule

According to [14], a FlexRay CC is **not** required to provide mechanisms in hardware to ensure synchronous access to its transmit and receive buffers e.g. by providing shadow buffers that may be accessed asynchronously by the AUTOSAR FlexRay software modules.

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 time[6] and thus avoid concurrent access to the buffers by the hardware and the software.

In order to provide this necessary synchronicity, the FrIf defines for each Cluster a FlexRay Job List (configuration parameter FrIfJobList).

The Cluster's FlexRay Job List is executed by its FlexRay Job List Execution Function (see also chapter 8.4.1) in the ISR of an absolute timer (configuration parameter FrIfAbsTimerRef) of a FlexRay CC connected to the respective Cluster.

7.4.2.1 FlexRay Job List

A FlexRay Job List is a list of **FlexRay Communication Jobs** sorted according to their respective execution start time.
Each Communication Job (configuration parameter FrIfJob) contains the following properties:

- Job start time by means of
  - FlexRay Communication Cycle (configuration paramer FrIfCycle)
  - Macrotick Offset within the Communication Cycle (configuration parameter FrIfMacrotick).
- A list of Communication Operations (configuration parameter FrIfCommunicationOperation) sorted according to a configureable Communication Operation index (configuration parameter FrIfCommunicationOperationIdx). The sorting order defines the order of execution of the Communication Operations within a FlexRay Communication Job.

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.

[6] In FlexRay Global Time

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

Each Communication Operation contains the following properties:

- Communication Operation Index (configuration parameter FrIfCommunicationOperationIdx), which determines the execution order of the Communication Operations within the Communication Job.
- Communication Action (configuration parameter FrIfCommunicationAction), which speficies the actual action to perform (see 7.4.3):
    - DECOUPLED_TRANSMISSION
    - TX_CONFIRMATION
    - RECEIVE_AND_STORE
    - RX_INDICATION
    - RECEIVE_AND_INDICATE
    - PREPARE_LPDU
- A reference to an L-PDU that is associated with the Communication Action to perform (configuration parameter FrIfLPduRef).

7.4.2.2 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.

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

- FrIf_JobListExec_0() for Cluster # 0 (FrIf_ClstIdx = 0)
- FrIf_JobListExec_1() for Cluster # 1 (FrIf_ClstIdx = 1)
- FrIf_JobListExec_2() for Cluster # 2 (FrIf_ClstIdx = 2)
- FrIf_JobListExec_3() for Cluster # 3 (FrIf_ClstIdx = 3)
- and so on, if more than 4 FlexRay Clusters are supported.

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

For its operation, the FlexRay Job List Execution Function utilizes a dedicated pointer, the **JobListPointer**, to select the next Job to execute, as well as a dedicated flag, the **JobListAsyncFlag**, to inform the FlexRay Interface Main Function about a loss of synchronicity with the FlexRay Cluster.

Upon invocation, the FlexRay Job List Execution Function performs the following steps:

1. Retrieve the FlexRay Global Time from the FlexRay CC that provides the Cluster's absolute timer interrupt.
2. If the FlexRay Global Time cannot be retrieved or the Global Time delay compared to the Communication Job's configured start time is larger than a

maximum delay (configuration parameter FrIfMaxIsrDelay), 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 the flag JobListAsyncFlag indicating to the FlexRay Interface Main Function that the execution of the FlexRay Job List of this Cluster is asynchronous and needs to be (re-)synchronized.
- Call Dem_ReportErrorStatus(FRIF_E_JLE_SYNC, DEM_EVENT_STATUS_FAILED).
- Disabled the absolute timer interrupt (using FrIf_DisableAbsoluteTimerIRQ()).
- Terminate the execution of this FlexRay Communication Job.

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

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

7.4.3 Communication Operations

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

7.4.3.1 DECOUPLED_TRANSMISSION

DECOUPLED_TRANSMISSION is exectued if the related CC is in FrIf_State FRIF_STATE_ONLINE. Otherwise, this Communication Operation is not executed

For Communication Operation DECOUPLED_TRANSMISSION the following steps are performed:

1. In case the parameter 'FrIfUnusedBitValue' exists, all the unused bits within the Frame Construction Plan shall be set to the configured value 'FrIfUnusedBitValue' while assembling the frame.
2. Iterate over all PDUs contained in the FrIfFrameStructure of the associated Trame triggering of this Comunication Operation and
3. Check whether TrigTxCounter is > 0 for the PDU. If not, clear the update-bit for this PDU (configuration parameter FrIfPduUpdateBitOffset) and proceed with the next PDU.
4. Call the upper layer's function <UL>_TriggerTransmit() with the asociated PDUId (FrIf_TxPduId) and pass a pointer to a temporary buffer within the FrIf that assembles the L-SDU. The pointer shall consider the byte offset (configuration parameter FrIfPduOffset) of the PDU within the Frame.
5. Decrement TrigTxCounter.

6. Remember that a transmission for this PDU is pending if a transmission confirmation is needed for this PDU (configuration parameter FrIfConfirm) (increment TxConfCounter[7]).
7. Set the update-bit if configured for this PDU (configuration parameter FrIfPduUpdateBitOffset).
8. If at least one PDU was requested for transmission, or the FlexRay Driver's API service Fr_TransmitTxLPdu() shall always be called for this L-PDU (configuration parameter FrIfAlwaysTransmit), the FlexRay Driver's API service Fr_TransmitTxLPdu() is called:
    a. Fr_CtrlIdx is derived according to the indexing scheme descibed in 7.2
    b. Fr_LPduIdx is set to the configured L-PDU index (configuration parameter FrIfLPduIdx) associated with the Communication Operation
    c. Fr_LSduPtr is set to the temporary FrIf L-SDU assembling buffer.
    d. Fr_LSduLength is set to the L-SDU length (configuration parameter FrIfLSduLength)
9. 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.

In case the parameter 'FrIfAllowDynamicLSduLength' exists and is set to TRUE for the associated frame triggering, the actual L-SDU length, that is passed to the driver (Fr_TransmitTxLPdu()), shall be determined taking into account the following for those PDUs only, which have been indicated via FrIf_Transmit():
- the position of the respective PDU within the L-SDU
- the actual length of the respective PDU as passed via FrIf_Transmit()
- the position of the update-bit of the respective PDU (if configured)

7.4.3.2 TX_CONFIRMATION

TX_CONFIRMATION is exectued if the related CC is in FrIf_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 service Fr_CheckTxLPduStatus():
    a. Fr_CtrlIdx is derived according to the indexing scheme descibed in 7.2
    b. Fr_LPduIdx is set to the configured L-PDU index (configuration parameter FrIfLPduIdx) 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 FrIfFrameStructure of the associated Frame Triggering. If TxConfCounter for a PDU is 0 proceed with the next PDU, otherwise

[7] Limited by static configuration [Configuration Parameter FrIfCounterLimit]

a. Call the upper layer's function <UL>_TxConfirmation() with the asociated PDUId (FrIf_TxPduId).
b. Decrement TxConfCounter.

7.4.3.3 RECEIVE_AND_STORE

RECEIVE_AND_STORE is exectued if the related CC is in FrIf_State FRIF_STATE_ONLINE and FRIF_STATE_RXONLY. Otherwise, this Communication Operation is ignored.

For Communication Operation RECEIVE_AND_STORE the following steps are performed:

1. Call the FlexRay Driver's API service Fr_ReceiveRxLPdu():
   a. Fr_CtrlIdx is derived according to the indexing scheme descibed in 7.2
   b. Fr_LPduIdx is set to the configured L-PDU index (configuration parameter FrIfLPduIdx) 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 FrIfFrameStructure of the associated Frame Triggering and:
   a. If an update bit was configured for the PDU (configuration parameter FrIfPduUpdateBitOffset) and the update bit for the PDU is not set, continue with the next PDU. Otherwise,
   b. Copy the PDU Payload from the temporary buffer considering the PDU offset within the L-SDU (configuration parameter FrIfPduOffset) into a FrIf PDU-related static buffer.
   c. Mark the PDU-related static buffer as up-to-date.

7.4.3.4 RX_INDICATION

RX_INDICATION is exectued if the related CC is in FrIf_State FRIF_STATE_ONLINE and FRIF_STATE_RXONLY. 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 FrIfFrameStructure 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 <UL>_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.

7.4.3.5 RECEIVE_AND_INDICATE

RECEIVE_AND_INDICATE is exectued if the related CC is in FrIf_State FRIF_STATE_ONLINE and FRIF_STATE_RXONLY. Otherwise this Communication Operation is ignored.

For Communication Operation RECEIVE_AND_INDICATE the following steps are performed:

1. Call the FlexRay Driver's API service Fr_ReceiveRxLPdu():
   a. Fr_CtrlIdx is derived according to the indexing scheme descibed in 7.2
   b. Fr_LPduIdx is set to the configured L-PDU index (configuration parameter FrIfLPduIdx) 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 FrIfFrameStructure of the associated Frame Triggering and:
   a. If an update bit was configured for the PDU (configuration parameter FrIfPduUpdateBitOffset) and the update bit for the PDU is not set, continue with the next PDU. Otherwise,
   c. Call the upper layer's function <UL>_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 FrIfPduOffset) as parameters.

7.4.3.6 PREPARE_LPDU

PREPARE_LPDU is exectued in every FrIf_State.

For Communication Operation PREPARE_LPDU the following steps are performed:

1. Call the FlexRay Driver's API service Fr_PrepareLPdu():
   a. Fr_CtrlIdx is derived according to the indexing scheme descibed in 7.2
   b. Fr_LPduIdx is set to the configured L-PDU index (configuration parameter FrIfLPduIdx) 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 harwdare resources if available at appropriate point of times. However, it is the responsibility of the FlexRay Driver to decide and validate ressource 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.

7.4.4 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 FrIf_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.

Note:
Continuouse mode shall not be used in combination with Immediate Buffer Access.

If an immediate PDU transmission is indicated by calling FrIf_Transmit() with FrIf_TxPduId beeing configured for an immediate PDU, the following steps are performed within the context of the FrIf_Transmit() API service:

1. Call the FlexRay Driver's API service Fr_TransmitTxLPdu():
   a. Fr_CtrlIdx is derived according to the indexing scheme descibed in 7.2
   b. Fr_LPduIdx is set to the configured L-PDU index (configuration parameter FrIfLPduIdx) associated with the FrIf_TxPduId.
   c. Fr_LSduPtr is set to the Pdu Payload pointer contained in the PduInfoPtr passed as parameter to FrIf_Transmit.
   d. Fr_LSduLength is set to the L-SDU length (configuration parameter FrIfLSduLength)
2. In case the Driver's API Fr_TransmitTxLPdu() returned E_OK (indicating that the transmission request succeeded) the TxConfCounter is incremented[8] for the respective PDU.

7.5 Error Classification

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.

Development error values are of type uint8.

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 Cycle value	Development	FRIF_E_INV_CYCLE	0x06

[8] Limited by static configuration [Configuration Parameter FrIfCounterLimit]

Type or error	*Relevance*	*Related error code*	*Value [hex]*
Invalid Macrotick value	Development	FRIF_E_INV_MACROTICK	0x07
Invalid FrIf_TxPdu Index	Development	FRIF_E_INV_TXPDUID	0x08
Invalid configuration index	Development	FRIF_E_INV_FRIFCONF_IDX	0x09
FrIf not initialized	Development	FRIF_E_NOT_INITIALIZED	0x0A
Job List Execution lost synchronicity to the FlexRay Global Time	Production	FRIF_E_JLE_SYNC	Assigned by DEM

Table 7-3: Definition of Error Codes

7.6 Error Detection

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 7.5 and chapter 8.

If the FRIF_DEV_ERROR_DETECT switch is set to ON, all FrIf API services other than FrIf_Init() and FrIf_GetVersionInfo() shall:

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

unless the FrIf has been initialized with a preceding call of FrIf_Init().

7.7 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

8.1 Imported types

8.1.1 Standard Types

The data types:

- Std_ReturnType
- Std_VersionInfoType
- PduIdType
- PduInfoType
- BusTrcvErrorType

shall be included from the files:

- Std_Types.h
- ComStackTypes.h

8.1.2 Types imported from Fr_GeneralTypes.h

The following types are shared between the FrIf and the various FlexRay Drivers and FlexRay Transceiver Drivers controlled by it. The types are defined in the file Fr_GeneralTypes.h from which the FlexRay Interface imports:

- Fr_ConfigType
- Fr_MTSStatusType
- Fr_ChannelType
- Fr_SyncStateType
- Fr_OffsetCorrectionType
- Fr_RateCorrectionType
- Fr_POCStatusType
- Fr_TxLPduStatusType
- Fr_RxLPduStatusType
- FrTrcv_TrcvModeType
- FrTrcv_TrcvWUReasonType

8.2 Type Definitions

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

8.2.1 FrIf_ConfigType

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

8.2.2 FrIf_StateType

Type:	Enumeration	
Range:	`FRIF_STATE_HALT`	The FlexRay CC is not ready for communication, the FlexRay cluster is not synchronized.
	`FRIF_STATE_OFFLINE`	The FlexRay CC ready for communication, the FlexRay cluster is synchronized. The software disables the communication capabilities.
	`FRIF_STATE_RXONLY`	The FlexRay CC is ready for communication, the FlexRay cluster is synchronized. The software disables the transmission communication capabilities.
	`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 FrIf_State according to the FlexRay Interface State Machine (see chapter 7.3).	

8.2.3 FrIf_StateTransitionType

Type:	Enumeration	
Range:	`FRIF_GOTO_ONLINE`	Literal for requesting transition into FRIF_STATE_ONLINE state.
	`FRIF_GOTO_RXONLY`	Literal for requesting transition into FRIF_STATE_RXONLY state.
	`FRIF_GOTO_OFFLINE`	Literal for requesting transition into FRIF_STATE_OFFLINE state.
	`FRIF_GOTO_HALT_IMMEDIATELY`	Literal for requesting transition into FRIF_STATE_HALT state by immediately aborting communication on the network (using FrIf_AbortCommunication()).
	`FRIF_GOTO_HALT_AT_EOC_STOP_COM`	Literal for requesting transition into FRIF_STATE_HALT state by halting communication on the network at the end of the communication cycle (using FrIf_HaltCommunication()).
Description:	Variables of this type are used to represent the FrIf_State transition according to the FlexRay Interface State Machine (see chapter 7.3).	

8.3 Function Definitions

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

Note:
Several API services of the FrIf exist as CC-based API service and in addition as Cluster-based API services. In general, in this chapter, the description of the Cluster-based API-services contains a call of the respective CC-based API service in a loop over all CCs belonging to the respective Cluster.

However, an implementation of the FrIf is allowed to replace the explicit call of the respective CC-based API service "inline" by the required code of this CC-based API service.
The same rule applies accordingly to Cluster-based Transceiver API services.

8.3.1 FrIf_GetVersionInfo

Service name:	FrIf_GetVersionInfo	
Syntax:	`void FrIf_GetVersionInfo ( Std_VersionInfoType *FrIf_VersionInfoPtr )`	
Service ID [hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	None	
Parameters (out):	`FrIf_VersionInfoPtr`	Pointer to a memory location where the FlexRay Interface version information shall be stored. 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_VersionInfoPtr equals NULL_PTR.
Return value:	None	

Description:	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 FrIfVersionInfoApi).
	
	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 FrIf_Cfg.h.
Caveats:	--
Configuration:	If pre-compile-time configuration parameter 'FRIF_VERSION_INFO_API' is 'ON' this API service is included in the compilation process. If pre-compile-time configuration parameter 'FRIF_VERSION_INFO_API' is 'OFF' this API service is excluded from the compilation process.

8.3.2 FrIf_Init

Service name:	FrIf_Init	

Syntax:	`void FrIf_Init`
	`(`
	`  uint8  FrIf_ConfigIdx,`
	`  const FrIf_ConfigType  *FrIf_ConfigPtr`
	`)`	

Service ID [hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non reentrant	
Parameters (in):	`FrIf_ConfigIdx`	**For the current release, this parameter has to be set to 0.** 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_FRIFCONF_IDX) if FrIf_ConfigIdx has an invalid value.
	`FrIf_ConfigPtr`	Base pointer to the configuration structure of the FlexRay Interface. 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_ConfigPtr equals NULL_PTR.
Parameters (out):	`None`	
Return value:	`None`	
Description:	This API service of the FlexRay Interface is used to initialize the FlexRay Interface, all FlexRay Drivers and all FlexRay Transceivers. This initialization is carried out by the following actions: 1. Configure the FlexRay Interface: initialize the FlexRay Interface State Machine for all CCs controlled by the FrIf, and initialize the local memory space used to store the PDU data, the PDU properties and state variables. 2. Ensure that the FlexRay Interface Main Function is called by SchM with the configured period (configuration parameter FrIfMainFunctionCycle). 3. Initialize all FlexRay Drivers by calling all FlexRay Drivers' API service Fr_Init(Fr_ConfigPtr) with: • Fr_ConfigPtr pointing to the configuration of the respective FlexRay Driver. The actual value of the pointer Fr_ConfigPtr used for configuring each respective FlexRay Driver is contained in the configuration set of the FrIf.	

	4. Initialize and configure all FlexRay Transceiver devices by calling all FlexRay Transceiver Drivers' API service FrTrcv_TrcvInit(FrTrcv_TrcvIdx) with: • FrTrcv_TrcvIdx looping over all Transceiver indices of the respective FlexRay Transceiver Driver (➔ multiple calls of FrTrcv_TrcvInit() per FlexRay Transceiver Driver). All configuration data needed to initialize each respective FlexRay Transceiver is contained in the static configuration of the FlexRay Transceiver Driver, and therefore do not have to be passed as parameter in the FrTrcv_TrcvInit() call. The AUTOSAR ComM calls this FlexRay Interface API service with the address of the static configuration structure of the FrIf in parameter FrIf_ConfigPtr. After the execution of this API service, the FlexRay Interface, all FlexRay BSW modules below, and the respective hardware devices controlled by these Drivers are in a defined state. This, however, does not mean that communication via FlexRay has already started up.
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).
Configuration:	--

8.3.3 FrIf_ControllerInit

Service name:	FrIf_ControllerInit	
Syntax:	`Std_ReturnType FrIf_ControllerInit ( uint8 FrIf_CtrlIdx )`	
Service ID [hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of FrIf_CtrlIdx reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_ControllerInit() 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_LowLevelConfSetIdx and Fr_BufConfSetIdx to 0. 3. Calling Fr_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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.4 FrIf_SendMTS

Service name:	FrIf_SendMTS	
Syntax:	`Std_ReturnType FrIf_SendMTS ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx )`	
Service ID [hex]:	0x14	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_SendMTS() 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_ChnlIdx to FrIf_ChnlIdx 3. Calling Fr_SendMTS() 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].	
Configuration:	--	

8.3.5 FrIf_StopMTS

Service name:	FrIf_StopMTS	
Syntax:	`Std_ReturnType FrIf_StopMTS ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx )`	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_StopMTS() 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_ChnlIdx to FrIf_ChnlIdx 3. Calling Fr_StopMTS() 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].	
Configuration:	--	

8.3.6 FrIf_CheckMTS

Service name:	FrIf_CheckMTS	
Syntax:	`Std_ReturnType FrIf_CheckMTS ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, Fr_MTSStatusType *FrIf_MTSStatusPtr )`	
Service ID [hex]:	0x15	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme.
Parameters (out):	`FrIf_MTSStatusPtr`	Pointer to a memory location where output value will be stored. 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_MTSStatusPtr equals NULL_PTR.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_CheckMTS() 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_ChnlIdx to FrIf_ChnlIdx • Fr_MTSStatusPtr to FrIf_MTSStatusPtr 3. Calling Fr_CheckMTS() 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].	
Configuration:	--	

8.3.7 FrIf_StartCommunication

Service name:	FrIf_StartCommunication	
Syntax:	`Std_ReturnType FrIf_StartCommunication ( uint8 FrIf_CtrlIdx )`	
Service ID [hex]:	0x07	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_StartCommunication() 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. 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].	
Configuration:	--	

8.3.8 FrIf_HaltCommunication

Service name:	FrIf_HaltCommunication	
Syntax:	`Std_ReturnType FrIf_HaltCommunication ( uint8 FrIf_CtrlIdx )`	
Service ID [hex]:	0x09	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_HaltCommunication() 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. Calling Fr_HaltCommunication() of the determined FlexRay Driver with the parameters determined as described above. 3. Set FRIF_STATE to FRIF_STATE_HALT for the FlexRay CC with index FrIf_CtrlIdx.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.9 FrIf_AbortCommunication

Service name:	FrIf_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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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). 2. Calling Fr_AbortCommunication() of the determined FlexRay Driver with the parameters determined as described above. 3. Set FRIF_STATE to FRIF_STATE_HALT for the FlexRay CC with index FrIf_CtrlIdx.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.10 FrIf_RequestControllerStateTransition

Service name:	FrIf_RequestControllerStateTransition	
Syntax:	`Std_ReturnType FrIf_RequestControllerStateTransition ( uint8 FrIf_CtrlIdx, FrIf_StateTransitionType FrIf_StateTransition )`	
Service ID [hex]:	0x35	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the CC addressed. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_StateTransition`	Requested FrIf state transition.
Parameters (out):	`None`	
Return value:	`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:	These API services of the FlexRay Interface requests for FlexRay CC with index FrIf_CtrIdx the FrIf_State reached by the state transition according to FrIf_StateTransistion.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.11 FrIf_RequestClusterStateTransition

Service name:	FrIf_RequestClusterStateTransition	
Syntax:	`Std_ReturnType FrIf_RequestClusterStateTransition ( uint8 FrIf_ClstIdx, FrIf_StateTransitionType FrIf_StateTransition )`	
Service ID [hex]:	0x36	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant	
Parameters (in):	`FrIf_ClstIdx`	Index of the FlexRay Cluster addressed. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ClstIdx has an invalid value.
	`FrIf_StateTransistion`	Requested state transition.
Parameters (out):	`None`	
Return value:	`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:	These API services of the FlexRay Interface requests for all FlexRay CCs within the addressed FlexRay Cluster FrIf_Clstdx the FrIf_State reached by the state transition according to FrIf_StateTransistion.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.12 FrIf_GetControllerState

Service name:	FrIf_GetControllerState	
Syntax:	`Std_ReturnType FrIf_GetControllerState ( uint8 FrIf_CtrlIdx, FrIf_StateType *FrIf_StatePtr )`	
Service ID [hex]:	0x37	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the CC of which the FrIf_State shall be retrieved. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`FrIf_StatePtr`	Pointer to a memory location where the retrieved FrIf_State will be stored. 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.
Return value:	`E_OK`	Function was successfully executed.
	`E_NOT_OK`	Function execution failed due to detected errors.
Description:	This API service of the FlexRay Interface retrieves the current FrIf_State of the FlexRay CC with index FrIf_CtrlIdx.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.13 FrIf_GetClusterState

Service name:	FrIf_GetClusterState	
Syntax:	`Std_ReturnType FrIf_GetClusterState ( uint8 FrIf_ClstIdx, FrIf_StateType *FrIf_StatePtr )`	
Service ID [hex]:	0x38	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_ClstIdx`	Index of the FlexRay Cluster of which the aggregated FrIf_State shall be retrieved. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ClstIdx has an invalid value.
Parameters (out):	`FrIf_StatePtr`	Pointer to a memory location where the retrieved FrIf_State will be stored. 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.
Return value:	`E_OK`	Function was successfully executed.
	`E_NOT_OK`	Function execution failed due to detected errors.
Description:	This API service of the FlexRay Interface retrieves the aggregated FrIf_State of the FlexRay CCs connected to the Cluster with index FrIf_ClstIdx by: 1. Retrieving the FrIf_State of each FlexRay CC contained in cluster FrIf_ClstIdx by e.g. executing FrIf_GetControllerState(). 2. Logically aggregate the retrieved FrIf_States of all FlexRay CCs to determine the FrIf_State of the FlexRay Cluster according the following rule: Always return the highest available communication capability. The communication capability order from highest to lowest is: a. FRIF_STATE_ONLINE b. FRIF_STATE_RXONLY c. FRIF_STATE_OFFLINE d. FRIF_STATE_HALT .	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.14 FrIf_SetWakeupChannel

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 FrIf_CtrlIdx reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_SetWakeupChannel() 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_ChnlIdx to FrIf_ChnlIdx 3. 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].	
Configuration:	--	

8.3.15 FrIf_SendWUP

Service name:	FrIf_SendWUP	
Syntax:	`Std_ReturnType FrIf_SendWUP ( uint8 FrIf_CtrlIdx )`	
Service ID [hex]:	0x12	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_SendWUP() 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. 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].	
Configuration:	--	

8.3.16 FrIf_GetSyncState

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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`FrIf_SyncStatePtr`	Pointer to a memory location where output value will be stored. 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 equals NULL_PTR.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_GetSyncState() 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_SyncStatePtr to FrIf_SyncStatePtr 3. 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].	
Configuration:	--	

8.3.17 FrIf_SetExtSync

Service name:	FrIf_SetExtSync	
Syntax:	`Std_ReturnType FrIf_SetExtSync ( uint8 FrIf_CtrlIdx, Fr_OffsetCorrectionType FrIf_Offset, Fr_RateCorrectionType FrIf_Rate )`	
Service ID [hex]:	0x17	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_Offset`	Offset correction that shall be applied
	`FrIf_Rate`	Rate correction that shall be applied
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_SetExtSync() 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_Rate to FrIf_Rate • Fr_Offset to FrIf_Offset 3. 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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.18 FrIf_GetPOCStatus

Service name:	FrIf_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 FrIf_CtrlIdx reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`FrIf_POCStatusPtr`	Pointer to a memory location where output value will be stored. 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_POCStatusPtr equals NULL_PTR.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_GetPOCStatus() 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_POCStatusPtr to FrIf_POCStatusPtr 3. 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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.19 FrIf_GetGlobalTime

Service name:	FrIf_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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`FrIf_CyclePtr`	Pointer to a memory location where output value will be stored. 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_CyclePtr equals NULL_PTR.
	`FrIf_MacroTickPtr`	Pointer to a memory location where output value will be stored. 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_MacroTickPtr equals NULL_PTR.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_GetGlobalTime() 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_CylcePtr to FrIf_CyclePtr • Fr_MacroTickPtr to FrIf_MacroTickPtr 3. 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].	
Configuration:	--	

8.3.20 FrIf_GetMacroticksPerCycle

Service name:	FrIf_GetMacroticksPerCycle	
Syntax:	`uint16 FrIf_GetMacroticksPerCycle ( uint8 FrIf_CtrlIdx )`	
Service ID [hex]:	0x1B	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`uint16`	Number of Macroticks per FlexRay Cycle of the FlexRay CC.
Description:	This API service of the FlexRay Interface retrieves the number of Macroticks per FlexRay Cycle of the FlexRay CC 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].	
Configuration:	GMacroPerCycle	

8.3.21 FrIf_GetCycleLength

Service name:	FrIf_GetCycleLength	
Syntax:	`uint32 FrIf_GetCycleLength ( uint8 FrIf_CtrlIdx, )`	
Service ID [hex]:	0x1D	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`uint32`	Time in unit of nanoseconds.
Description:	This API returns the configured time of the parameter “GdCycle” in nanoseconds for the FlexRay controller with index FrIf_CtrlIdx.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	GdCycle	

8.3.22 FrIf_ConvertNanosecToMacroticks

Service name:	FrIf_ConvertNanosecToMacroticks	
Syntax:	`uint32 FrIf_ConvertNanosecToMacroticks ( uint8 FrIf_CtrlIdx, uint32 FrIf_Nanosec )`	
Service ID [hex]:	0x1C	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_Nanosec`	Time in unit of nanoseconds.
Parameters (out):	`None`	
Return value:	`uint32`	Time in unit of macrotickes.
Description:	This API service of the FlexRay Interface converts out of static configuration the given time in nanoseconds to the corresponding time in macroticks for the FlexRay controller with index FrIf_CtrlIdx.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.23 FrIf_ConvertMacroticksToNanosec

Service name:	FrIf_ConvertMacroticksToNanosec	
Syntax:	`uint32 FrIf_ConvertMacroticksToNanosec ( uint8 FrIf_CtrlIdx, uint32 FrIf_MacroTick )`	
Service ID [hex]:	0x1D	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_MacroTick`	Time in unit of macroticks. 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_MACROTICK) if FrIf_MacroTick has an invalid value.
Parameters (out):	`None`	
Return value:	`uint32`	Time in unit of nanoseconds.
Description:	This API service of the FlexRay Interface converts out of static configuration the given time in macroticks to the corresponding time in nanoseconds for the FlexRay controller with index FrIf_CtrlIdx.	
Caveats:	The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.24 FrIf_SetAbsoluteTimer

Service name:	FrIf_SetAbsoluteTimer	
Syntax:	`Std_ReturnType FrIf_SetAbsoluteTimer ( uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx, uint8 FrIf_Cycle, uint16 FrIf_Offset )`	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_AbsTimerIdx`	Index of the absolute timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_AbsTimerIdx has an invalid value.
	`FrIf_Cycle`	FlexRay Cycle number to be set. 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_CYCLE) if FrIf_Cycle has an invalid value.
	`FrIf_Offset`	Number of Macroticks to be set. 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_MACROTICK) if FrIf_Offset has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_SetAbsoluteTimer() 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_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].
Configuration:	--

8.3.25 FrIf_SetRelativeTimer

Service name:	FrIf_SetRelativeTimer	
Syntax:	`Std_ReturnType FrIf_SetRelativeTimer ( uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx, uint16 FrIf_Offset )`	
Service ID [hex]:	0x1F	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC reentrant for different FlexRay CCs	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_RelTimerIdx`	Index of the relative timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_RelTimerIdx has an invalid value.
	`FrIf_Offset`	Number of Macroticks the relative timer shall be set to.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_SetRelativeTimer() 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_Offset to FrIf_Offset 3. 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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.26 FrIf_CancelAbsoluteTimer

Service name:	FrIf_CancelRelativeTimer	
Syntax:	`Std_ReturnType FrIf_CancelAbsoluteTimer ( uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx )`	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_AbsTimerIdx`	Index of the absolute timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_AbsTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_CancelAbsoluteTimer() 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_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].	
Configuration:	--	

8.3.27 FrIf_CancelRelativeTimer

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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_RelTimerIdx`	Index of the relative timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_RelTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_CancelRelativeTimer() by: 4. Translating (based on static FrIf configuration) the FlexRay CC index FrIf_CtrlIdx into a tuple (FlexRay Driver | Driver-specific CC index Fr_CtrlIdx). 5. Setting parameters • Fr_RelTimerIdx to FrIf_RelTimerIdx 6. 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].	
Configuration:	--	

8.3.28 FrIf_EnableAbsoluteTimerIRQ

Service name:	FrIf_EnableAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType FrIf_EnableAbsoluteTimerIRQ ( uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx )`	
Service ID [hex]:	0x22	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_AbsTimerIdx`	Index of the absolute timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_AbsTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_EnableAbsoluteTimerIRQ() 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_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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.29 FrIf_EnableRelativeTimerIRQ

Service name:	FrIf_EnableRelativeTimerIRQ	
Syntax:	`Std_ReturnType FrIf_EnableRelativeTimerIRQ ( uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx )`	
Service ID [hex]:	0x23	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC reentrant for different FlexRay CCs	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_RelTimerIdx`	Index of the relative timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_RelTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service Fr_EnableRelativeTimerIRQ() 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_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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.30 FrIf_GetAbsoluteTimerIRQStatus

Service name:	FrIf_GetAbsoluteTimerIRQStatus	
Syntax:	`Std_ReturnType FrIf_GetAbsoluteTimerIRQStatus ( uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx, boolean *FrIf_IRQStatusPtr )`	
Service ID [hex]:	0x39	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_AbsTimerIdx`	Index of the absolute timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_AbsTimerIdx has an invalid value.
Parameters (out):	`FrIf_IRQStatusPtr`	Pointer to a memory location where output value will be stored.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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].	
Configuration:	--	

8.3.31 FrIf_GetRelativeTimerIRQStatus

Service name:	FrIf_GetRelativeTimerIRQStatus	
Syntax:	`Std_ReturnType FrIf_GetRelativeTimerIRQStatus ( uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx, boolean *FrIf_IRQStatusPtr )`	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_RelTimerIdx`	Index of the relative timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_RelTimerIdx has an invalid value.
Parameters (out):	`FrIf_IRQStatusPtr`	Pointer to a memory location where output value will be stored.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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].	
Configuration:	--	

8.3.32 FrIf_AckAbsoluteTimerIRQ

Service name:	FrIf_AckAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType FrIf_AckAbsoluteTimerIRQ ( uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx )`	
Service ID [hex]:	0x24	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_AbsTimerIdx`	Index of the absolute timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_AbsTimerIdx has an invalid value.
Parameters (out):	None	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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].	
Configuration:	--	

8.3.33 FrIf_AckRelativeTimerIRQ

Service name:	FrIf_AckRelativeTimerIRQ	
Syntax:	`Std_ReturnType FrIf_AckRelativeTimerIRQ ( uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx )`	
Service ID [hex]:	0x25	
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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_RelTimerIdx`	Index of the relative timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_RelTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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].	
Configuration:	--	

8.3.34 FrIf_DisableAbsoluteTimerIRQ

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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_AbsTimerIdx`	Index of the absolute timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_AbsTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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].	
Configuration:	--	

8.3.35 FrIf_DisableRelativeTimerIRQ

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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_RelTimerIdx`	Index of the relative timer to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_TIMER_IDX) if FrIf_RelTimerIdx has an invalid value.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Driver API service 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].	
Configuration:	--	

8.3.36 FrIf_Transmit

Service name:	FrIf_Transmit	
Syntax:	`Std_ReturnType FrIf_Transmit ( PduIdType FrIf_TxPduId, const PduInfoType *FrIf_PduInfoPtr )`	
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. 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.
	`FrIf_PduInfoPtr`	Pointer to a structure with FlexRay PDU related data. 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_PduInfoPtr or SduDataPtr in *FrIf_PduInfoPtr equals NULL_PTR.
Parameters (out):	`None`	
Return value:	`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: • TrigTxCounter has reached it's limit • FlexRay Driver reported an error in case of immediate transmission • An error has been detected in development mode
Description:	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_TxPduId 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 FrIfCommunicationAction 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 date 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 [FrIf05389].	
Configuration:	--	

[9] Limited by static configuration [Configuration Parameter FrIfCounterLimit]

8.3.37 FrIf_SetTransceiverMode

Service name:	FrIf_SetTransceiverMode	
Syntax:	`Std_ReturnType FrIf_SetTransceiverMode ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcvModeType FrIf_TrcvMode )`	
Service ID [hex]:	0x28	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
	`FrIf_TrcvMode`	Transceiver mode to be set.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service 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_TrcvIdx). 2. Setting parameters • FrTrcv_TrcvMode to FrIf_TrcvMode 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].	
Configuration:	--	

8.3.38 FrIf_GetTransceiverMode

Service name:	FrIf_GetTransceiverMode	
Syntax:	`Std_ReturnType FrIf_GetTransceiverMode ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcvModeType *FrIf_TrcvModePtr )`	
Service ID [hex]:	0x2A	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`FrIf_TrcvModePtr`	Pointer to a memory location where output value will be stored.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service 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].	
Configuration:	--	

8.3.39 FrIf_GetTransceiverWUReason

Service name:	FrIf_GetTransceiverWUReason	
Syntax:	`Std_ReturnType FrIf_GetTransceiverWUReason ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx, FrTrcv_TrcvWUReasonType *FrIf_TrcvWUReasonPtr )`	
Service ID [hex]:	0x2B	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`FrIf_TrcvWUReasonPtr`	Pointer to a memory location where output value will be stored.
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service FrTrcv_GetTransceiverWUReason() 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_TrcvWUReasonPtr to FrIf_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 FrIf_Init() before this API service may be called, see [FrIf05389].	
Configuration:	--	

8.3.40 FrIf_EnableTransceiverWakeup

Service name:	FrIf_EnableTransceiverWakeup	
Syntax:	`Std_ReturnType FrIf_EnableTransceiverWakeup ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx )`	
Service ID [hex]:	0x2C	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service FrTrcv_EnableTransceiverWakeup() 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. 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].	
Configuration:	--	

8.3.41 FrIf_DisableTransceiverWakeup

Service name:	FrIf_DisableTransceiverWakeup	
Syntax:	`Std_ReturnType FrIf_DisableTransceiverWakeup ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx )`	
Service ID [hex]:	0x2E	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service FrTrcv_DisableTransceiverWakeup() 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. 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 [FrIf05389].	
Configuration:	--	

8.3.42 FrIf_ClearTransceiverWakeup

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. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	`None`	
Return value:	`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:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service FrTrcv_EnableTransceiverWakeup() 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. 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].	
Configuration:	--	

8.4 Interrupt Service Routines

8.4.1 FrIf_JobListExec_<ClstIdx>

Service name:	FrIf_JobListExec_<ClstIdx>

Syntax:	`void FrIf_JobListExec_<ClstIdx>`
	`(`
	`  void`
	`)`

Service ID [hex]:	0x32
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	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 7.4.2.2.
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. The FlexRay Interface has to be initialized with a call of FrIf_Init() before this API service may be called, see [FrIf05389].
Configuration:	--

8.5 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

8.5.1 FrIf_Cbk_WakeupByTransceiver

Service name:	FrIf_Cbk_WakeupByTransceiver	
Syntax:	`void FrIf_Cbk_WakeupByTransceiver ( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx )`	
Service ID [hex]:	0x3B	
Sync/Async:	Synchronous	
Reentrancy:	reentrant	
Parameters (in):	`FrIf_CtrlIdx`	Index of the FlexRay CC to address. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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_ChnlIdx`	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx. Please refer to chapter 7.2 for more information on the Indexing Scheme. 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.
Parameters (out):	None	
Return value:	None	
Description:	This API service of the FlexRay Interface wraps the FlexRay Transceiver Driver API service FrTrcv_Cbk_WakeupByTransceiver() 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. 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].	
Configuration:	--	

8.6 Scheduled Functions

8.6.1 FrIf_MainFunction_<ClstIdx>

Service name:	FrIf_MainFunction_<ClstIdx>

Syntax:	`Void FrIf_MainFunction_<ClstIdx>`
	`(`
	`  void`
	`)`

Service ID [hex]:	0x33
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	None
Parameters (out):	None
Return value:	None
Description:	This cyclically executed API service of the FlexRay Interface serves the following purposes: • Maintenance (processign requested transtitions (see 8.3.10) and monitoring the actual FlexRay CC POC state of all CCs connected to the Cluster with index ClstIdx. • Initially program the absolute timer interrupt in order to start the execution of FrIf_JobListExec_<ClstIdx>(). • Monitoring the proper (in time) execution of the FrIf_JobListExec_<ClstIdx>(). Please refere to chapter 7.3 for a detailed description.
Timing:	variable cyclic, independently configurable per Cluster (configuration parameter FrIfMainFunctionCycle).
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 (configuration parameter FrIfMainFunctionCycle).
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 [FrIf05389].
Configuration:	--

8.7 Expected Interfaces

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

8.7.1 Mandatory Interfaces

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

API service	*Module*	*Description*
Fr_*	Fr	All API services of the AUTOSAR FlexRay Driver.
FrTrcv_*	FrTrcv	All API services of the AUTOSAR FlexRay Transceiver Driver.
Dem_ReportErrorStatus	Dem	API service to report production error status information.

8.7.2 Optional Interfaces

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

API service	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
Det_ReportError	Det	Development error notification	FRIF_DEV_ERROR_DETECT

8.7.3 Configurable Interfaces

This chapter lists all interfaces where the target API service of any upper layer 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 [1]. 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 FrIf 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 FrIf 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.

8.7.3.1 <UL_RxIndication>

Name:	<UL_RxIndication>

Syntax:	`void <UL_RxIndication>`
	`(`
	`  PduIdType  FrIf_RxPduId,`
	`  const uint8  *FrIf_SduPtr`
	`)`

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
	`FrIf_SduPtr` Pointer to FlexRay SDU (buffer of received payload)
Parameters (out):	None
Return value:	None
Description:	This API service of an upper layer BSW module <UL> (e.g. PduR, FrTp, FrNm) is called by the FlexRay Interface to indicate to this upper layer BSW module that the PDU with index FrIf_RxPduId has been received via the FlexRay Communication System. During the execution of this API service, the upper layer BSW module that is the final recipient of this PDU is expected to retireve (i.e. copy) the SDU (i.e. the payload of the PDU) by means of the pointer FrIf_SduPtr and the SDU length known from this BSW module's static configuration.
Caveats:	This API service is called during the execution of the FlexRay Job List Execution Function.
Configuration:	--

8.7.3.2 <UL_TxConfirmation>

Name:	<UL_TxConfirmation>

Syntax:	`void <UL_TxConfirmation>`
	`(`
	`  PduIdType  FrIf_TxPduId`
	`)`

Reentrancy:	Reentrant for different FrIf_TxPduId Non reentrant for the same FrIf_TxPduId
Parameters (in):	`FrIf_TxPduId` PDU-ID of the FlexRay PDU of which the transmission is being confirmed
Parameters (out):	None
Return value:	None
Description:	This API service of an upper layer BSW module <UL> (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.
Configuration:	--

8.7.3.3 <UL_TriggerTransmit>

Name:	<UL_TriggerTransmit>	

Syntax:	`void <UL_TriggerTransmit>`
	`(`
	`  PduIdType  FrIf_TxPduId,`
	`  uint8  *FrIf_SduPtr`
	`)`	

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
	`FrIf_SduPtr`	Pointer to a memory location in a temporary L-SDU buffer of the FrIf
Parameters (out):	None	
Return value:	None	
Description:	This API service of an upper layer BSW module <UL> (e.g. PduR, FrTp, FrNm) is called by the FlexRay Interface to request from this upper layer BSW module that the PDU with index FrIf_TxPduId be copied to the location in a temporary L-SDU buffer of the FrIf to which FrIf_SduPtr points. The configuration tools of the FrIf and the respective upper layer BSW module have to ensure that the length of the PDU with index FrIf_TxPduId is equally defined in both BSW modules. Therefore, the PDU length does not need to be passed in this API service.	
Caveats:	This API service is called during the execution of the FlexRay Job List Execution Function.	
Configuration:	--	

9 Sequence Diagrams

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

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

9.1 Data Transmission

9.1.1 Transmit With Immediate Buffer Access

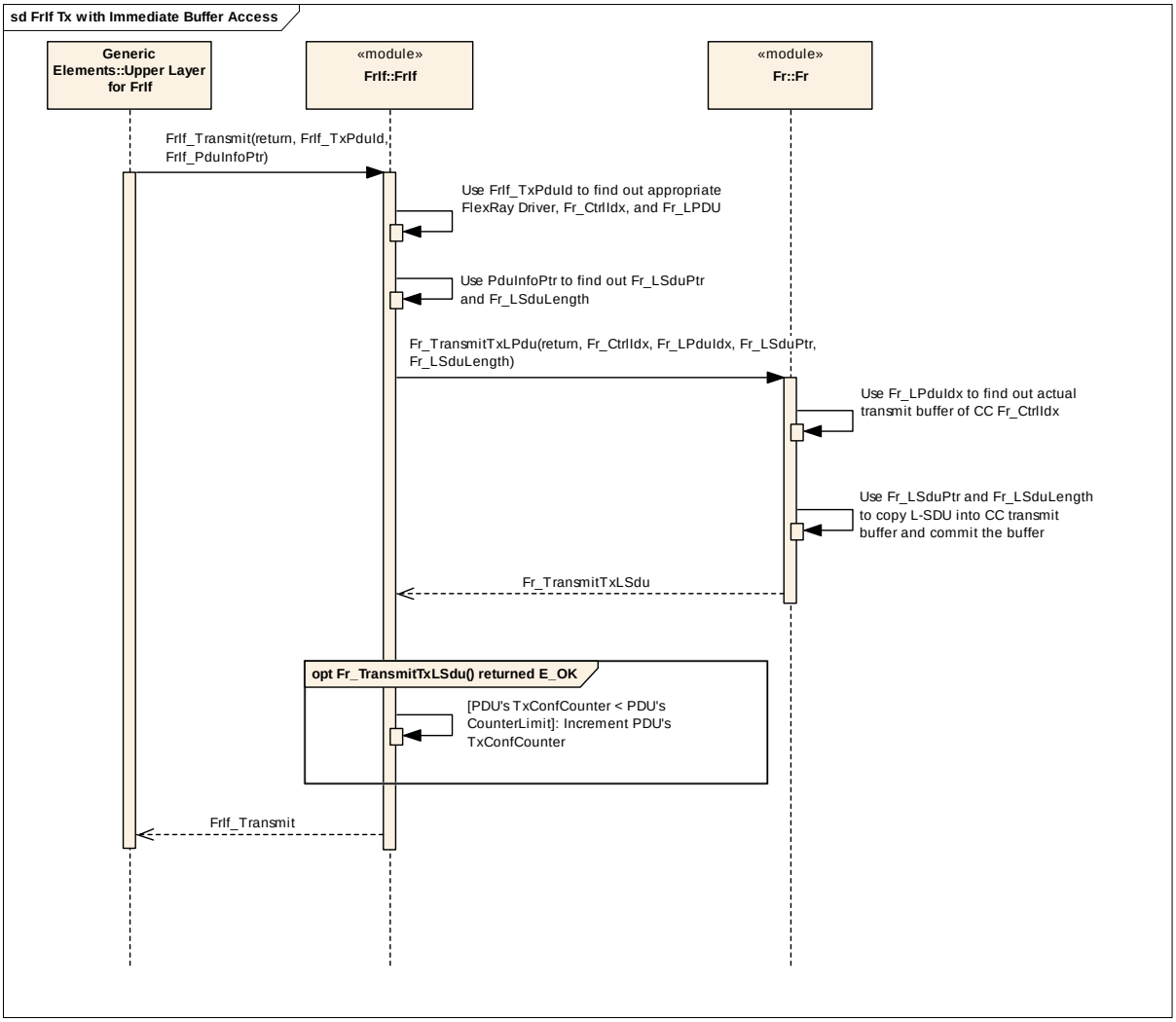

Figure 9-1: Transmit With Immediate Buffer Access

9.1.2 TransmitWithDecoupledBufferAccess

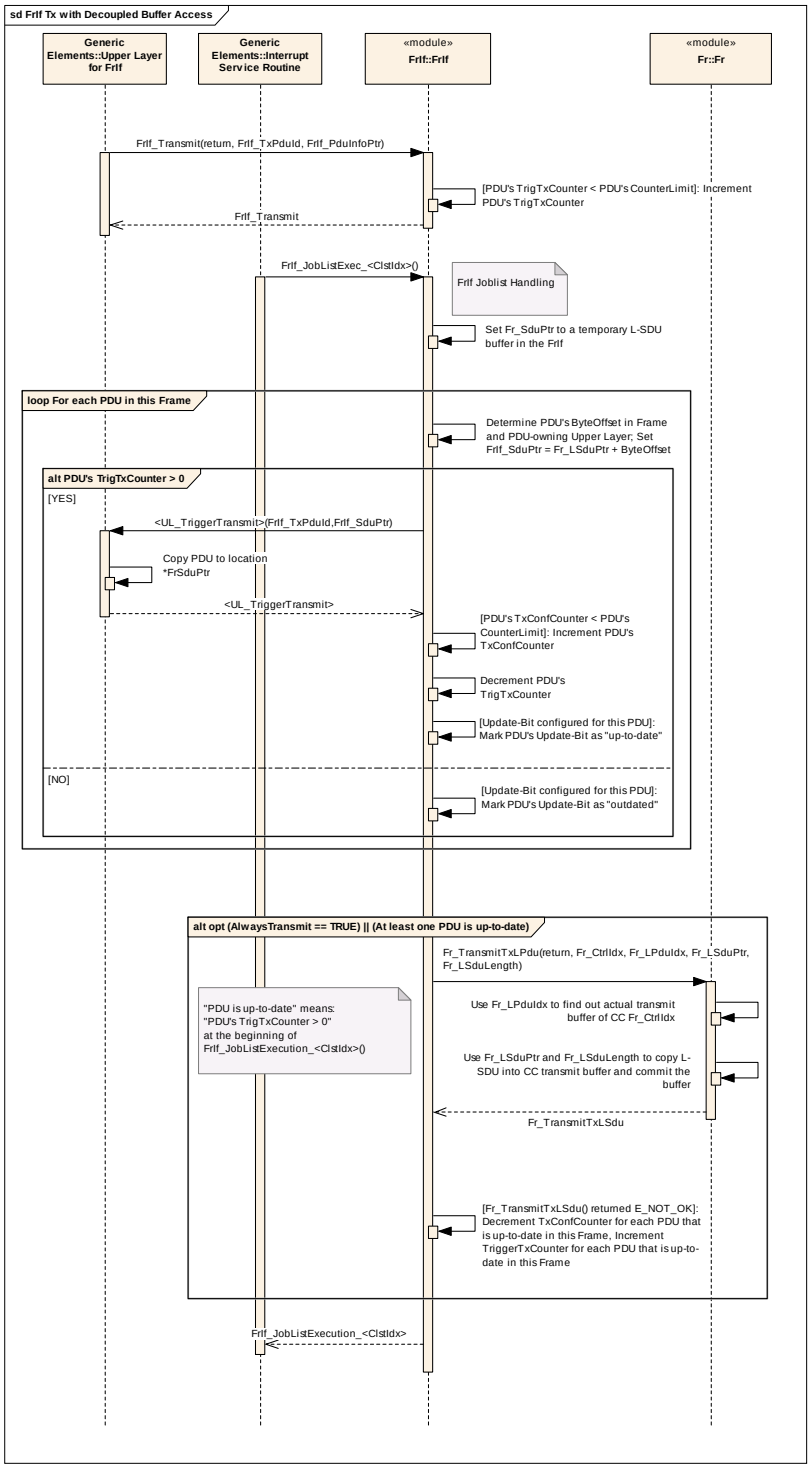

Figure 9-2: TransmitWithDecoupledBufferAccess

9.1.3 ProvideTxConfirmation

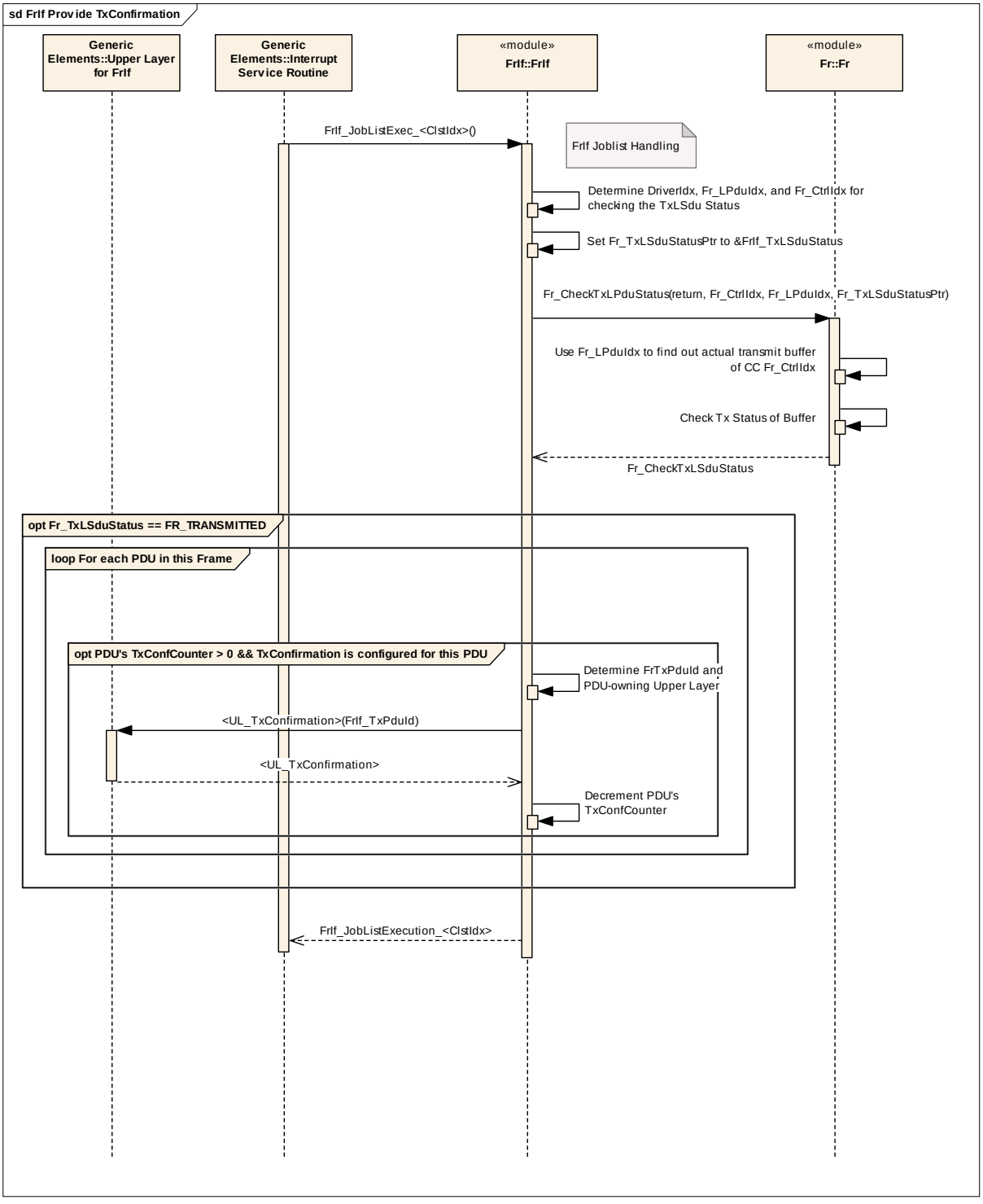

Figure 9-3: ProvideTxConfirmation

9.2 Data Reception

9.2.1 ReceiveAndIndicate

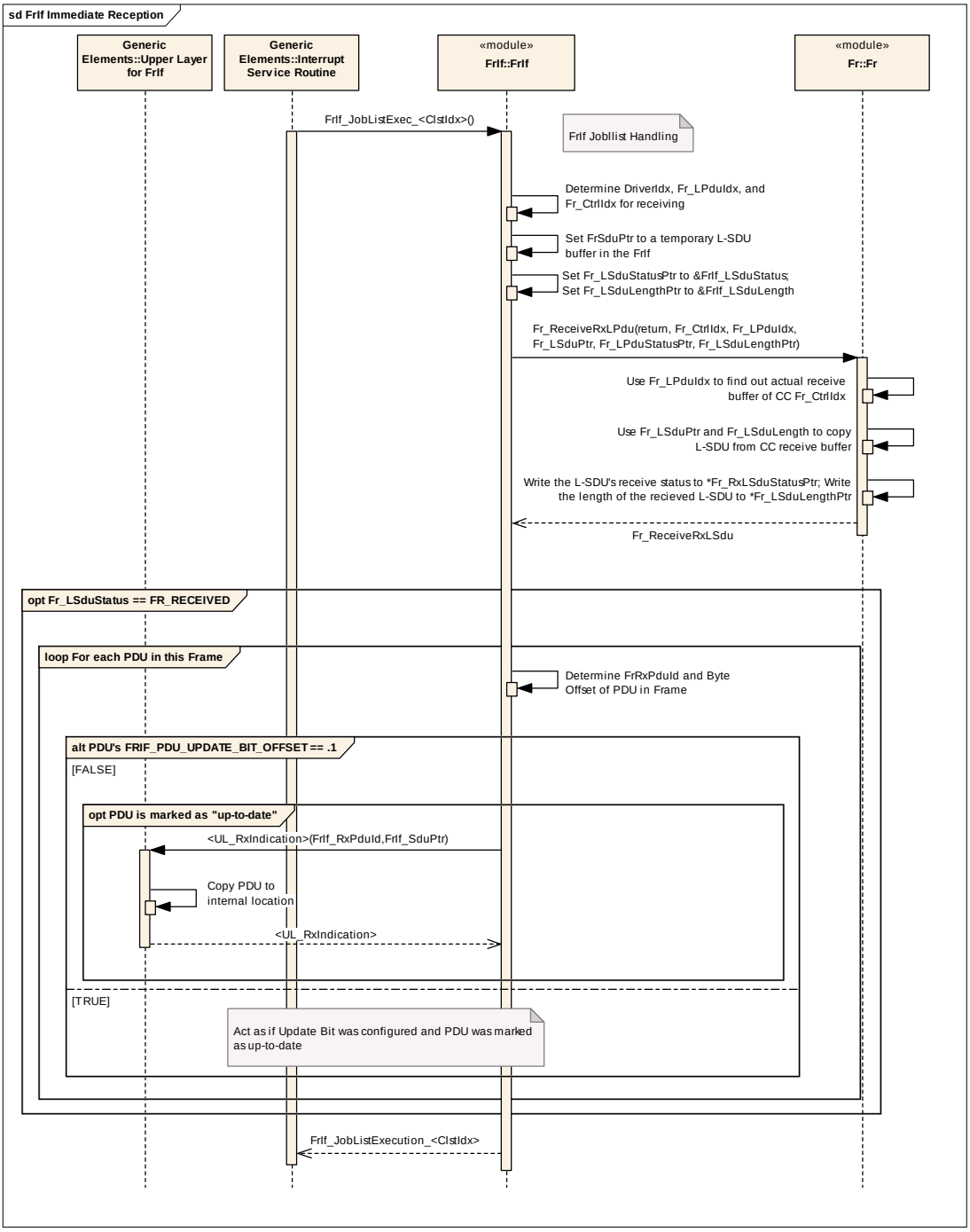

Figure 9-4: ReceiveAndIndicate

9.2.2 ReceiveAndStore

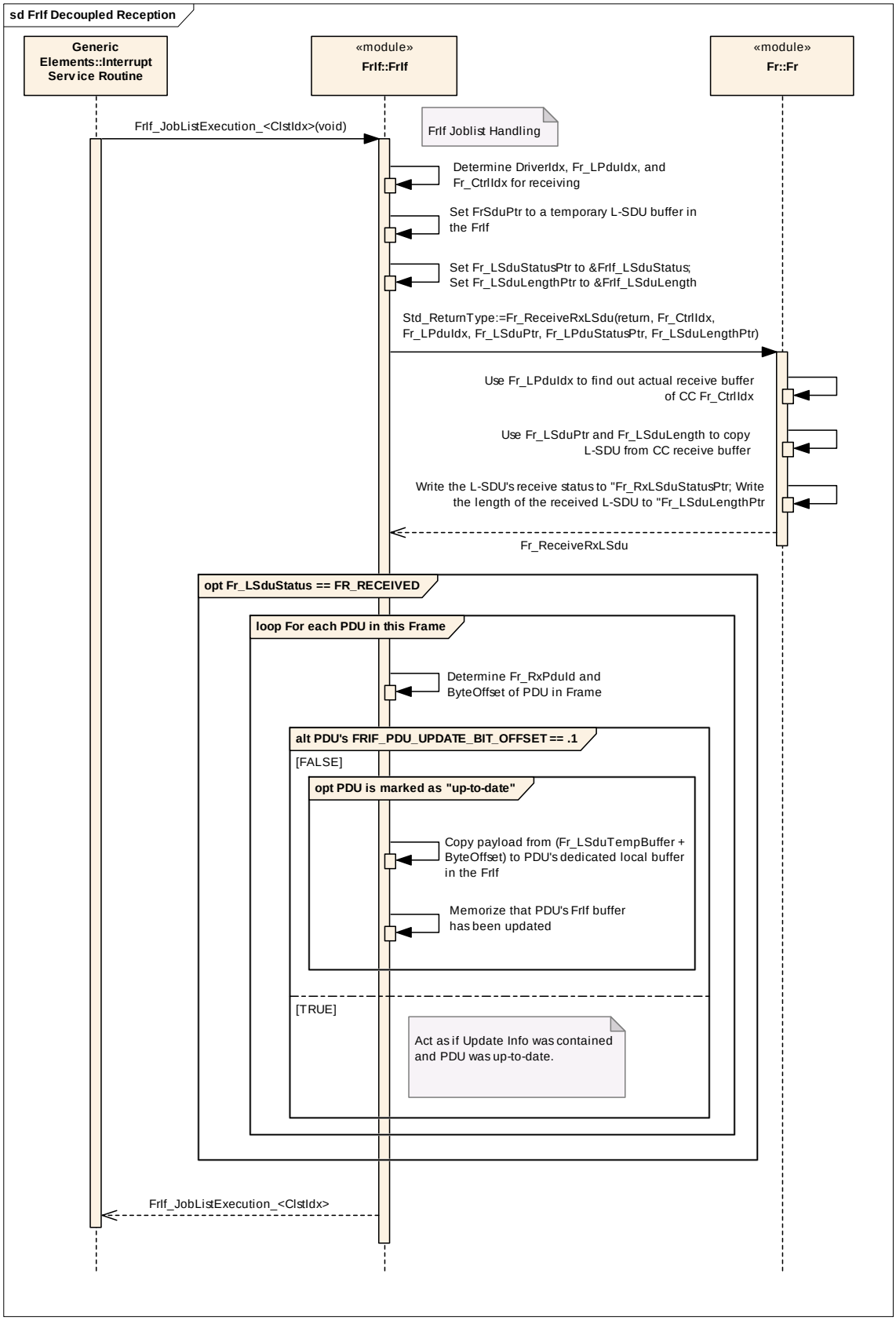

Figure 9-5: ReceiveAndStore

9.2.3 ProvideRxIndication

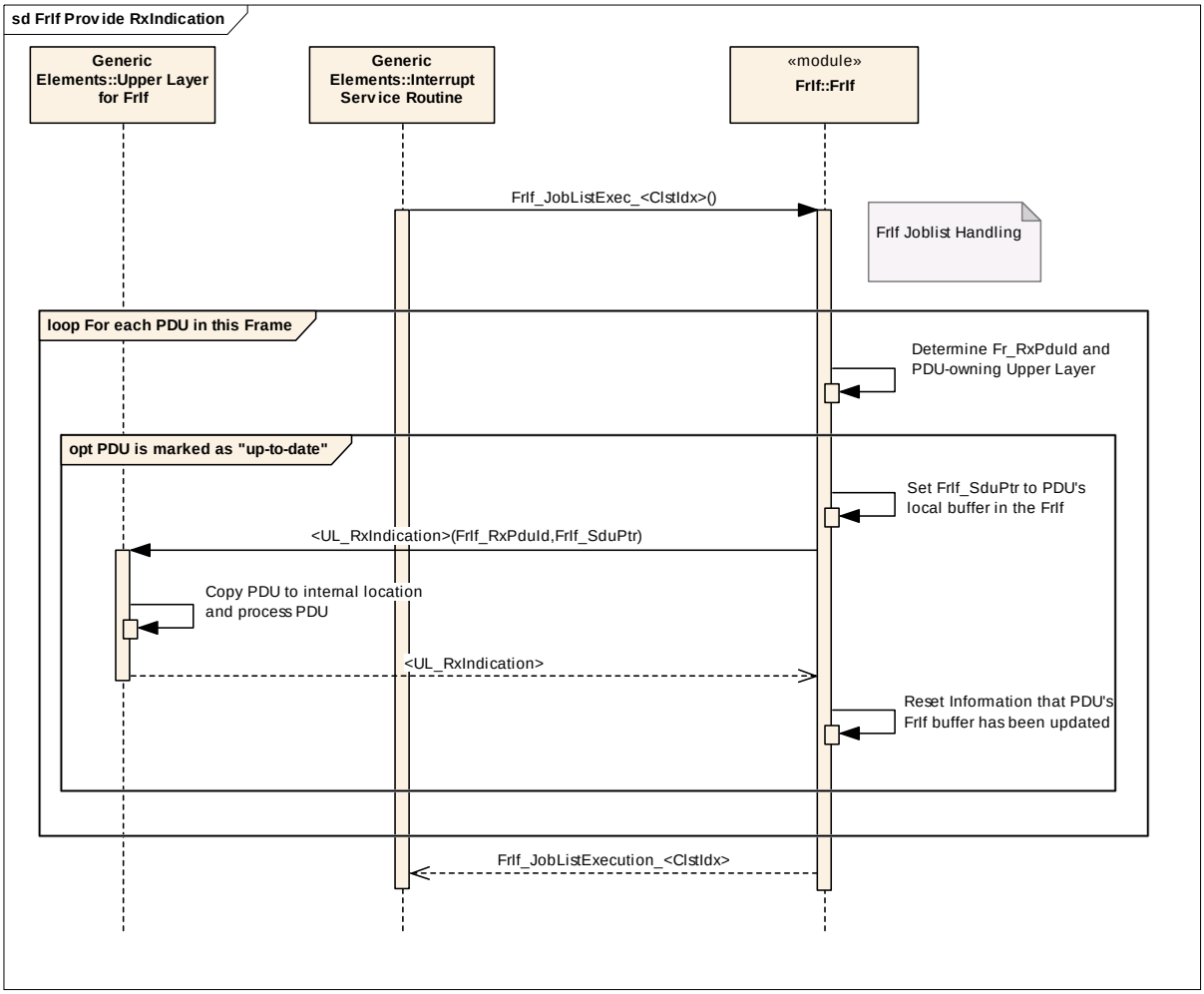

Figure 9-6: ProvideRxIndication

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.

All configuration data of the FrIf 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

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

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

10.1.4 Specification Template for Configuration Parameters

The following tables consist of three sections:

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

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

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

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

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

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

10.2 Containers and Configuration Parameters

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 FrIf.

The configuration tool must check the consistency of the configuration at configuration time. Configuration rules and constraints for plausibility checks shall be performed during configuration time, wherever possible.

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

10.2.1 Variants

There is only one variant of the FlexRay Interface, which is described hereunder.

10.2.2 FlexRay Interface Configuration

SWS Item	
Container Name	FrIf
Description	This container contains the general configuration of the FlexRay Interface.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfGeneral	1	Modul
FrIfCluster	1 … *	Module/Fr
FrIfFrameStructure	0 … *	Modul
FrIfPdu	0 … *	Modul

10.2.2.1 FlexRay Interface General Configuration

SWS Item	
Container Name	FrIfGeneral
Description	This container contains the general configuration of the FlexRay Interface.
Configuration Parameters	

Name	FrIfVersionInfoApi		
Description	Enables/disables the existence of the FrIf_GetVersionInfo() API service. The pre compile time switch FRIF_VERSION_INFO_API is derived from this configuration parameter.		
Type	BOOLEAN-PARAM-DEF		
Range	true	FrIf_GetVersionInfo() API service exists, FRIF_VERSION_INFO_API is set to ON	
	false	FrIf_GetVersionInfo() API service does not exist, FRIF_VERSION_INFO_API is set to OFF	
Configuration Class	***Pre-compile***	x	
	Link time	--	
	Post build time	--	
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfDevErrorDetect		
Description	Switches the Development Error Detection and Notification on or off. The pre compile time switch FRIF_DEV_ERROR_DETECT is derived from this configuration parameter.		
Type	BOOLEAN-PARAM-DEF		
Range	true	Development Error Detection and Notification on, FRIF_DEV_ERROR_DETECT is set to ON	
	false	Development Error Detection and Notification off, FRIF_DEV_ERROR_DETECT is set to OFF	
Configuration Class	***Pre-compile***	x	
	Link time	--	
	Post build time	--	
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfNumCtrlSupported		
Description	Maximum number of FlexRay CCs that the FlexRay Interface supports		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	1 … *		
Configuration Class	***Pre-compile***	--	
	Link time	X	
	Post build time	--	
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfUnusedBitValue		
Description	Set unused bits to a defined value		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0...1		
Configuration Class	***Pre-compile***	X	
	Link time	--	
	Post build time	--	
Scope	Module		
Dependency	--		
Multiplicity	0..1		

Name	FrIfNumClstSupported		
Description	Maximum number of FlexRay Clusters that the FlexRay Interface supports		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	1 … *		
Configuration Class	***Pre-compile***	--	
	Link time	X	
	Post build time	--	
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.2 FlexRay Cluster

SWS Item	
Container Name	FrIfCluster
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	

Name	FrIfClstIdx		
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.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfMaxIsrDelay		
Description	The maximum allowable delay in macroticks between the scheduled and the actual invocation time of FrIf_JobListExec_<ClstIdx>().		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	0 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfMainFunctionCycle		
Description	Main function cycle length		
Type	FLOAT-PARAM-DEF		
Unit	s		
Range	-		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	GColdStartAttempts		
Description	Maximum number of times a node in the Cluster is permitted to attempt to start the Cluster by initiating schedule synchronization.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	2 … 31		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdActionPointOffset		
Description	Number of Macroticks the action point is offset from the beginning of a Static Slots or symbol window		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	1 … 63		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdCasRxLowMax		
Description	Upper limit of the CAS acceptance window		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	67 … 99		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdDynamicSlotIdlePhase		
Description	Duration of the idle phase within a Dynamic Slot		
Type	INTEGER-PARAM-DEF		
Unit	Minislots		
Range	0 … 2		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdMinislot		
Description	Duration of a Minislot		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	2 … 63		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdMiniSlotActionPointOffset		
Description	Number of Macroticks the Minislot action point is offset from the beginning of a Minislot		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	1 … 31		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdStaticSlot		
Description	Duration of a Static Slot		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	4 … 659		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdSymbolWindow		
Description	Duration of the symbol window		
Unit	Macroticks		
Type	INTEGER-PARAM-DEF		
Range	0 … 139		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdTssTransmitter		
Description	Number of bits in the Transmission Start Sequence		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	3 … 15		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdWakeupSymbolRxIdle		
Description	Number of bits used by the node to test the duration of the 'idle' portion of a received wakeup symbol. Duration is equal to (gdWakeupSymbolTxIdle - gdWakeupSymbolTxLow)/2 minus a safe part. (Collisions, clock differences, and other effects can deform the Tx-wakeup pattern.)		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	14 … 59		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdWakeupSymbolRxLow		
Description	Number of bits used by the node to test the LOW portion of a received wakeup symbol. This lower limit of zero bits has to be received to detect the LOW portion by the receiver. The duration is equal to gdWakeupSymbolTxLow minus a safe part. (Active stars, clock differences, and other effects can deform the Tx-wakeup pattern.)		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	10 … 55		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdWakeupSymbolRxWindow		
Description	The size of the window used to detect wakeups. Detection of a wakeup requires a low and idle period (from one WUS) and a low period (from another WUS) to be detected entirely within a window of this size. The duration is equal to gdWakeupSymbolTxIdle + 2 * gdWakeupSymbolTxLow plus a safe part. (Clock differences and other effects can deform the Tx-wakeup pattern.)		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	76 … 301		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdWakeupSymbolTxIdle		
Description	Number of bits used by the node to transmit the 'idle' part of a wakeup symbol. The duration is equal to cdWakeupSymbolTxIdle		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	45 … 180		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdWakeupSymbolTxLow		
Description	Number of bits used by the node to transmit the LOW part of a wakeup symbol. The duration is equal to cdWakeupSymbolTxLow		
Type	INTEGER-PARAM-DEF		
Unit	gdBit		
Range	15 … 60		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GListenNoise		
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.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	2 … 16		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GMacroPerCycle		
Description	Number of Macroticks in a Communication Cycle		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	10 … 16000		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GMaxWithoutClockCorrectFatal		
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*.		
Type	INTEGER-PARAM-DEF		
Unit	Even/Odd cycle pairs		
Range	1 … 15		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GMaxWithoutClockCorrectPassive		
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		
Type	INTEGER-PARAM-DEF		
Unit	Even/Odd cycle pairs		
Range	1 … 15		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GNumberOfMinislots		
Description	Number of Minislots in the Dynamic Segment		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … 7986		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GNumberOfStaticSlots		
Description	Number of Static Slots in the static segment		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	2 … 1023		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GPayloadLengthStatic		
Description	Payload length of a static Frame		
Type	INTEGER-PARAM-DEF		
Unit	16bit words		
Range	0 … 127		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GSyncNodeMax		
Description	Maximum number of nodes that may send Frames with the sync Frame indicator bit set to one		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	2 … 15		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GChannels		
Description	The Channels that are used by the Cluster		
Type	Fr_ChannelType		
Range	[FR_CHANNEL_A, FR_CHANNEL_B, FR_CHANNEL_AB]		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GClusterDriftDamping		
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		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0 … 5		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdBit		
Description	Nominal bit time		
Type	ENUMERATION-PARAM-DEF		
Range	[t100ns, t200ns, t400ns]		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdCycle		
Description	Length of the Cycle		
Type	[FLOAT-PARAM-DEF		
Unit	s		
Range	0.000001 … 0.016		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdMacrotick		
Description	Duration of the Cluster wide nominal Macrotick		
Type	FLOAT-PARAM-DEF		
Unit	s		
Range	0.000001 ... 0.000006		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdMaxInitializationError		
Description	Maximum error that a node may have following integration		
Type	FLOAT-PARAM-DEF		
Unit	s		
Range	0.0 … 0.0000117		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdNit		
Description	Duration of the Network Idle Time		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	2 … 767		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GdSampleClockPeriod		
Description	Sample clock period		
Type	ENUMERATION-PARAM-DEF		
Range	[t12_5ns, t25ns, t50ns]		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GNetworkManagementVectLength		
Description	Length of the Network Management vector in a Cluster		
Type	INTEGER-PARAM-DEF		
Unit	bytes		
Range	0 … 12		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GOffsetCorrectionMax		
Description	describes the maximum value which the offset correction should assume		
Type	FLOAT-PARAM-DEF		
Unit	s		
Range	0.0000005 … 0.0003811		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Name	GOffsetCorrectionStart		
Description	Start of the offset correction phase within the NIT, expressed as the number of Macroticks from the start of Cycle		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	9 … 15999		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module/Fr		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfController	1 … *	
FrIfJobList	0 … 1	

10.2.2.2.1 FlexRay Controller

SWS Item	
Container Name	FrIfController
Description	This container contains the configuration of a FlexRay CC.
Configuration Parameters	

Name	FrIfCtrlIdx		
Description	This parameter provides a zero-based consecutive index of the FlexRay Communication Controllers. Upper layer BSW modules and the FrIf itself use this index to identify a FlexRay CC.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Name	FrIfFrCtrlRef		
Description	Reference to a Controller, which is handled by a specific Driver. This reference is unique for the ECU.		
Type	SYMBOLIC-NAME-REFERENCE-DEF to /AUTOSAR/Fr/FrController		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfAbsTimer	1 … *	--
FrIfRelTimer	0 … *	--
FrIfTransceiver	0 … 2	--
FrIfLPdu	0 … *	--
FrIfFrameTriggering	0 … *	--

10.2.2.2.1.1 Absolute Timer

SWS Item	
Container Name	FrIfAbsTimer
Description	This container contains the configuration of an absolute timer of a FlexRay CC.
Configuration Parameters	

Name	FrIfAbsTimerIdx		
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.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Name	FrIfFrAbsTimerRef		
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.		
Type	SYMBOLIC-NAME-REFERENCE-DEF to /AUTOSAR/Fr/FrController/FrAbsoluteTimer		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.2.1.2 Relative Timer

SWS Item	
Container Name	FrIfRelTimer
Description	This container contains the configuration of a relative timer of a FlexRay CC.
Configuration Parameters	

Name	FrIfRelTimerIdx		
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.		
Type	INTEGER-PARAM-DEF		
Unit			
Range	0 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Name	FrIfFrRelTimerRef		
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.		
Type	SYMBOLIC-NAME-REFERENCE-DEF to /AUTOSAR/Fr/FrController/FrRelativeTimer		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.2.1.3 Transceiver

SWS Item	
Container Name	FrIfTransceiver
Description	Up to two FlexRay Transceivers may connect a Controller to a Cluster. This container realizes a Controller–Transceiver assignment.
Configuration Parameters	

Name	FrIfFrTrcvChannelRef		
Description	Reference to a Transceiver Driver Channel. This reference is unique for the ECU.		
Type	SYMBOLIC-NAME-REFERENCE-DEF to /AUTOSAR/FrTrcv/FrTrcvChannel		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Name	FrIfClusterChannel		
Description	This parameter identifies to which one of the two Channels "A" or "B" of the Cluster the Transceiver is connected.		
Type	ENUMERATION-PARAM-DEF		
Unit	--		
Range	FRIF_CHANNEL_A	Transceiver is connected to channel A	
	FRIF_CHANNEL_B	Transceiver is connected to channel B	
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.2.1.4 LPdu

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

Name	FrIfLPduIdx		
Description	This parameter identifies the L-PDU in the interaction between FlexRay Interface and FlexRay Driver.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		

Name	FrIfFrameTriggeringRef		
Description	Reference to the assigned Frame Triggering.		
Type	REFERENCE-DEF to /AUTOSAR/FrIf/FrIfCluster/FrIfController/FrIfFrameTriggering		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
--	--	--

10.2.2.2.1.5 Frame Triggering

SWS Item	
Container Name	FrIfFrameTriggering
Description	A Frame Triggering contains the communication parameters of the FlexRay Frame as well as a reference to the Frame Construction Plan.
Configuration Parameters	

Name	FrIfFrameStructureRef		
Description	Reference to the Frame Construction Plan of the FlexRay Frame.		
Type	REFERENCE-DEF to /AUTOSAR/FrIf/FrIfFrameStructure		
Unit	--		
Range	--		
Configuration Class	**Pre-compile**	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfSlotId		
Description	This parameter contains the FlexRay Slot ID used to transmit this FlexRay Frame.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	1 … 2047		
Configuration Class	**Pre-compile**	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfBaseCycle		
Description	This parameter contains the FlexRay Base Cycle used to transmit this FlexRay Frame.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … 63		
Configuration Class	**Pre-compile**	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfCycleRepetition		
Description	This parameter contains the FlexRay Cycle Repetition used to transmit this FlexRay Frame.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	$2^R = 2^0 \ldots 2^6 = 1, 2, 4, 8, \ldots 64$		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfChannel		
Description	This parameter contains the FlexRay Channel used to transmit this FlexRay Frame.		
Type	ENUMERATION-PARAM-DEF		
Unit	--		
Range	FRIF_CHANNEL_A		Channel A
	FRIF_CHANNEL_B		Channel B
	FRIF_CHANNEL_AB		Channel A and Channel B
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfAlwaysTransmit		
Description	Defines whether the FlexRay Driver's API service Fr_TransmitTxLPdu() shall always be called for this L-PDU, independent from any PDUs in this L-PDU being marked as up-to-date.		
Type	BOOLEAN-PARAM-DEF		
Unit	--		
Range	TRUE / FALSE		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfAllowDynamicLSduLength		
Description	Allows L-PDU length reduction ('FrIfLSduLength' defines max. length) and indicates that the related CC buffer has to be reconfigured for the actual length and Header-CRC before transmission of the L-PDU.		
Type	BOOLEAN-PARAM-DEF		
Unit	--		
Range	TRUE / FALSE		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	0..1		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
--	--	--

10.2.2.2.2 FlexRay Job List

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

Name	FrIfAbsTimerRef		
Description	It is a reference to the absolute timer to be used to trigger the interrupt whose ISR contains the FrIf_JobListExec_<ClstIdx>() function.		
Type	REFERENCE-DEF to /AUTOSAR/FrIf/FrIfCluster/FrIfController/FrIfAbsTimer		
Unit	--		
Range	--		
Configuration Class	**Pre-compile**	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
FrIfJob	1 … *	--

10.2.2.2.2.1 FlexRay Job

SWS Item	
Container Name	FrIfJob
Description	A Job may contain more than one operation that are executed at a specific point in time.
Configuration Parameters	

Name	FrIfCycle		
Description	Number of the FlexRay Cycle in which to execute this Job.		
Type	INTEGER-PARAM-DEF		
Unit	FlexRay Cycle		
Range	0 … 63		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfMacrotick		
Description	Macrotick offset within the Cycle when to execute this Job		
Type	INTEGER-PARAM-DEF		
Unit	Macrotick		
Range	0 … 16000		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfCommunicationOperation	1 … *	

10.2.2.2.2.2 FlexRay Communication Operation

SWS Item	
Container Name	FrIfCommunicationOperation
Description	A separate operation which is part of a FlexRay Job and defines what type of action is executed.
Configuration Parameters	

Name	FrIfCommunicationOperationIdx		
Description	For each Communication Operation, this index spans a range of zero-based consecutive values of communication operations.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfCommunicationAction		
Description	This parameter defines the sction to be performed for each communication operation.		
Type	ENUM-PARAM-DEF		
Unit	--		
Range	DECOUPLED_TRANSMISSION		Decoupled transmission
	TX_CONFIRMATION		Transmission confirmation
	RECEIVE_AND_STORE		Decoupled reception
	RX_INDICATION		Reception indication
	RECEIVE_AND_INDICATE		Immediate reception
	PREPARE_LPDU		Prepare LPdu for operation
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfLPduRef		
Description	Reference to an LPdu		
Type	REFERENCE-DEF to /AUTOSAR/FrIf/FrIfCluster/FrIfController/FrIfLPdu		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.3 Frame Structure

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

Name	FrIfLSduLength		
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.		
Type	INTEGER-PARAM-DEF		
Unit	Number of bytes		
Range	0 … 254		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfPdusInFrame	1 … *	

10.2.2.3.1 PDU in Frame

SWS Item	
Container Name	FrIfPdusInFrame
Description	This container holds all the information about a PDU in a FlexRay Frame.
Configuration Parameters	

Name	FrIfPduOffset		
Description	The value specifies the offset of the PDU within the Frame in units of bytes.		
Type	INTEGER-PARAM-DEF		
Unit	Byte		
Range	0 … 253		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	X	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfPduRef		
Description	This is the reference to the local definition of a PDU.		
Type	REFERENCE-DEF to /AUTOSAR/FrIf/FrIfPdu		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfPduUpdateBitOffset		
Description	This value specifies where the PDU's Update-Bit is stored in the Frame. If the value is marked as invalid, no Update-Bit is used for the PDU.		
Type	INTEGER-PARAM-DEF		
Unit	Bit		
Range	0 … 2031		Bit location of PDU's Update-Bit in the FlexRay Frame, counted from the first transmitted payload bit on the bus.
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	0..1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.4 FlexRay PDU

SWS Item	
Container Name	FrIfPdu
Description	FrIfPdu contains PDU information. A PDU may be either a transmission PDU or a reception PDU. So, this container holds exactly one sub container (FrIfTxPdu or FrIfRxPdu)
Configuration Parameters	

Name	FrIfPduLength		
Description	This value specifies the length of a Pdu in units of bytes.		
Type	INTEGER-PARAM-DEF		
Unit	Byte		
Range	0 … 254		Pdu length in unit sof bytes
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfTxPdu	0/1	
FrIfRxPdu	1/0	

10.2.2.4.1 Transmission PDU

SWS Item	
Container Name	FrIfTxPdu
Description	This container specifies transmission PDUs.
Configuration Parameters	

Name	FrIfTxPduId		
Description	The FrIf-specific TxPDU identifier, which has to be used by the upper layer BSW module. The identifier has to be zero based and consecutive.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0 … (n-1)		n = number of Tx PDUs in FrIf
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	upper layer modules transmitting via FrIf		
Dependency	--		
Multiplicity	1		

Name	FrIfConfirm		
Description	Defines whether the transmission of a PDU should be checked and confirmed to the PDU owning BSW module.		
Type	BOOLEAN-PARAM-DEF		
Unit	--		
Range	TRUE / FALSE		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfImmediate		
Description	Defines whether the this is an immediate or a decoupled Pdu.		
Type	BOOLEAN-PARAM-DEF		
Unit	--		
Range	TRUE / FALSE		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfCounterLimit		
Description	This value states the maximum number of indication of ready PDU data to the FrIf (i.e. maximum number of invocations of FrIf_Transmit) without an intermediate transmission of the PDU.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	1 … *		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	Module		
Dependency	--		
Multiplicity	1		

Name	FrIfPduRef		
Description	Reference to the external PDU definition		
Type	REFERENCE-DEF to /AUTOSAR/EcuC/PduCollection/Pdu		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

10.2.2.4.2 Reception PDU

SWS Item	
Container Name	FrIfRxPdu
Description	This container specifies reception PDUs.
Configuration Parameters	

Name	FrIfPduRef		
Description	Reference to the external PDU definition		
Type	REFERENCE-DEF to /AUTOSAR/EcuC/PduCollection/Pdu		
Unit	--		
Range	--		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post build time	x	--
Scope	ECU		
Dependency	--		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
--	--	--

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.

SWS Item		
Information elements		
Information element name	***Type / Range***	***Information element description***
FRIF_VENDOR_ID	#define/ uint16	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
FRIF_MODULE_ID	#define/ uint8	Module ID of this module from Module List
FRIF_AR_MAJOR_VERSION	#define/ uint8	Major version number of AUTOSAR specification on which the appropriate implementation is based on.
FRIF_AR_MINOR_VERSION	#define/ uint8	Minor version number of AUTOSAR specification on which the appropriate implementation is based on.
FRIF_AR_PATCH_VERSION	#define/ uint8	Patch level version number of AUTOSAR specification on which the appropriate implementation is based on.
FRIF_SW_MAJOR_VERSION	#define/ uint8	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
FRIF_SW_MINOR_VERSION	#define/ uint8	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
FRIF_SW_PATCH_VERSION	#define/ uint8	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.