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

Document Version	3.1.0
Document Status	Final
Part of Release	3.1
Revision	5

Document Change History

Date	Version	Changed by	Change Description
20.09.2010	3.1.0	AUTOSAR Administration	• Update sequence chart 9-6 • Extension of FrIf05063 (behavior in case of return value E_NOT_OK for API TriggerTransmit()
23.06.2008	3.0.2	AUTOSAR Administration	Legal disclaimer revised
22.01.2008	3.0.1	AUTOSAR Administration	Correction of Figure 5.1
18.12.2007	3.0.0	AUTOSAR Administration	• Simplification of the FlexRay Interface State Machine due to the introduction of the new AUTOSAR SWS FlexRay State Manager • Cluster-based APIs were removed due to the introduced AUTOSAR SWS FlexRay State Manager • The FlexRay Interface does not initialize any other modules any more due to the introduction of the "flat initialization" for AUTOSAR release 3.0Document meta information extended • Small layout adaptations made
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

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

1 Introduction and Functional Overview

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).

FrIf05095:
Likewise, in order to access the FlexRay Transceiver(s), the FlexRay Interface shall use one or multiple FlexRay Transceiver Driver module(s), which abstract the specific features and interfaces of the respective FlexRay Transceiver(s)

FrIf05096:
Therefore, the FlexRay Interface executable code (however, not the configuration used during runtime) shall be completely independent of the FlexRay Communication Controller(s) and the FlexRay Transceiver(s).

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

FrIf05097:
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

FrIf05098:
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.

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)
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

[14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[15] AUTOSAR Basic Software Module Description Template,
AUTOSAR_BSW_Module_Description.pdf

3.2 Related Standards and Norms

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

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

4 Constraints and Assumptions

4.1 Limitations

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

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

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

FrIf05008:
In order for the AUTOSAR FlexRay BSW (FrIf and FlexRay Driver) modules to be able to control a FlexRay CC, this CC must allow for configuring its transmit/receive buffers to support the Cycle Counter Filter Criterion / (Support of Slot/Cycle Mulitplexing)

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
- $B < 2^R$

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

FrIf05074:

5.1 AUTOSAR Operating System

FrIf05099:
There is one dedicated FlexRay Job List Execution Function for each FlexRay Cluster.

FrIf05100:
The FlexRay Interface executes the Flexray Job List Execution Function.

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

5.2 AUTOSAR BSW Scheduler

FrIf05101:
There is one dedicated FlexRay Interface Main Function for each FlexRay Cluster

FrIf05102:
Each of these FlexRay Interface Main Functions must be called cyclically from a task body provided by the BSW Scheduler. The calling period must be configurable.

5.3 All Upper Layer AUTOSAR BSW Modules

FrIf05050**:**
The calling of the FlexRay Job List Execution Function synchronously to the FlexRay Global Time shall ensure that both the indication (to an upper layer BSW module) of received data and the request (to an upper layer BSW module) for data to be sent occur synchronously to the FlexRay Global Time.

FrIf05049**:**
If the respective upper layer BSW module does not operate synchronously to the FlexRay Global Time, these occurrences are asynchronous to the code execution of this BSW module. Therefore, this BSW module shall allow access to its PDU buffers at all times and it also has to ensure data consistency in its buffers.

5.4 AUTOSAR PDU-Router

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

5.5 AUTOSAR FlexRay Network Management

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

5.6 AUTOSAR FlexRay Transport Protocol

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

5.7 AUTOSAR FlexRay Driver

The 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.

FrIf05051:
The FlexRay Driver shall be the only BSW Modules which has to run necessarily synchronous to the FlexRay Interface.

5.8 AUTOSAR FlexRay Transceiver Driver

The 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).

5.9 AUTOSAR Development Error Tracer

FrIf05065:
In order to be able to report development errors, the FrIf has to have access to the error hook of the Development Error Tracer.

5.10 AUTOSAR Diagnostic Event Manager

FrIf05066:
In order to be able to report production errors, the FrIf has to have access to the Diagnostic Event Manager.

5.11 File Structure

5.11.1 Code File Structure

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

FrIf05075:
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

5.11.2 Header File Structure

FrIf05076:

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_Types.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	

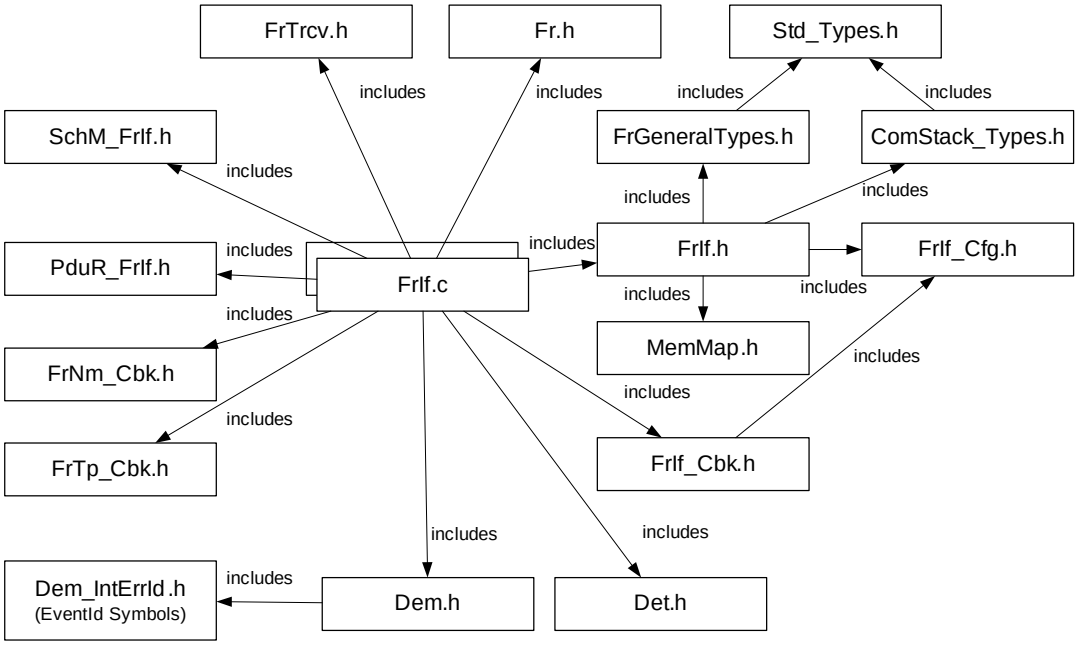


Figure 5-1: FlexRay Interface Header File Structure

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.

FrIf05081:
All files related to the FrIf module shall follow the naming convention *FrIf[_<description>].<extension>*

FrIf05091:
The FrIf module shall include the *Dem.h* file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols, which are provided by XML to the DEM configuration tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in *Dem_IntErrId.h*.

FrIf05140:
The implementation of the FrIf module shall provide the header file *FrIf.h*, which is the main module interface file. It shall contain all types and function prototypes required by the FrIf module's environment.

FrIf05141:
The implementation of the FrIf module shall provide the header file *FrIf_Cfg.h* that shall contain the pre-compile-time configuration parameters.

6 Requirements Traceability

6.1 General Requirements on Basic Software Modules

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

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

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

6.2 Requirements on FlexRay

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

BSW05069	Get FlexRay Cluster Mode	N/A (moved to FrSm)
BSW05153	Get FlexRay Controller Mode	N/A (moved to FrSm)
BSW05022	Get FlexRay CC POC Status	FrIf05014
BSW05023	Get FlexRay CC Sync State	FrIf05012
BSW05035	MTS Sending	N/A (no use without BG)
BSW05038	Get MTS Reception Status	N/A (no use without BG)
BSW05039	Set FlexRay Transceiver Operation Mode	FrIf05034
BSW05162	Set Cluster-wide FlexRay Transceiver Operation Mode	N/A (moved to FrSm)
BSW05157	Get FlexRay Transceiver Operation Mode	FrIf05035
BSW05174	Interrupt Handling	
BSW05170	Receive PDU	FrIf05062
BSW05027	Transmit PDU	FrIf05063
BSW05016	Abortion of a FlexRay CC Communication	FrIf05007
BSW05113	Abortion of a FlexRay Cluster Communication	N/A (moved to FrSm)
BSW05063	Halt of a FlexRay CC Communication	FrIf05006
BSW05102	Halt of a FlexRay Cluster Communication	N/A (moved to FrSm)
BSW05102	Provide Error Information	FrIf05065, FrIf05066
BSW05006	Abstraction of FlexRay-Specific Features	
BSW05009	Local Memory Space Usage	
BSW05056	Configuration of the FlexRay Interface at System Configuration Time	FrIf05067, FrIf05054

6.3 Specification Items

The following Items shall be seen as implementation hints only!

Functional Specification

Usage of abstract index	FrIf05103
Passing of translated index	FrIf05104
Abstraction of FlexRay Transceivers	FrIf05105
Usage of Controller and Channel Index	FrIf05106
Usage of zero-based index	FrIf05107
Usage of FR Cluster Index	FrIf05108
Configuration Data	FrIf05109
Usage of PDU index	FrIf05110
Support one of both or both FlexRay Channels	FrIf05111
Support of at least four FlexRay Clusters	FrIf05112
Support of at least one absolute timer per FlexRay CCs	FrIf05113
Support of at least one relative timer per FlexRay CC	FrIf05114

FlexRay Interface State Machine

One State Machine per Cluster	FrIf05115
FrIf_State offline during initialization	FrIf05116
FrIf_State offline during initialization	FrIf05117

FlexRay Interface Main Function

One Main Function for each FlexRay Cluster	FrIf05119
Main Function tasks	FrIf05120

Data Communication via FlexRay

Packaging of multible PDUs in one FR Frame	FrIf05121
Frame construction plan (layout)	FrIf05122
Frame construction plan (config)	FrIf05123
Transmission rule	FrIf05124
Update Information per PDU	FrIf05125
Location of Update Information	FrIf05126
Configuration of Update Information	FrIf05127
Indication in case of no update information	FrIf05128
Transmission with Immediate Buffer Access	FrIf05129
Ensure synchronous buffer access	FrIf05130
Sortation of Communication Job	FrIf05131
Communication Job properties	FrIf05132
Communication Job execution start time	FrIf05133
Actions specified by Communication Operation	FrIf05134
Communication Operation properties	FrIf05135
Job List Execution Function nameing	FrIf05136
Job List synchronously to global time	FrIf05137
Job List Execution Function actions	FrIf05138

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 Drivers modules. The FlexRay Transceivers are not shown in this figure; however, the structure that applies to the FlexRay Drivers and the FlexRay CCs analogously applies to the FlexRay Transceiver Drivers and the FlexRay Transceivers.

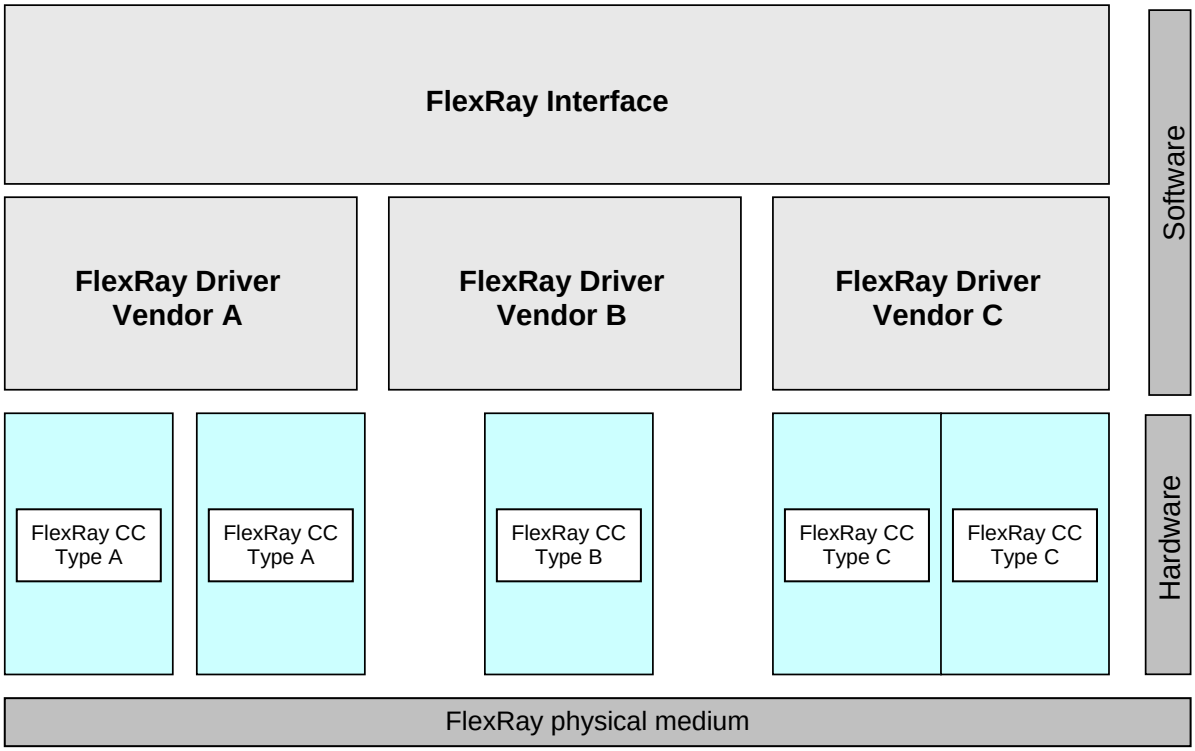


Figure 7-1: Basic Structure of the FlexRay BSW Stack

7.2 Indexing Scheme

7.2.1 Principle

Most of the FrIf'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.

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

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 exemplarily described in

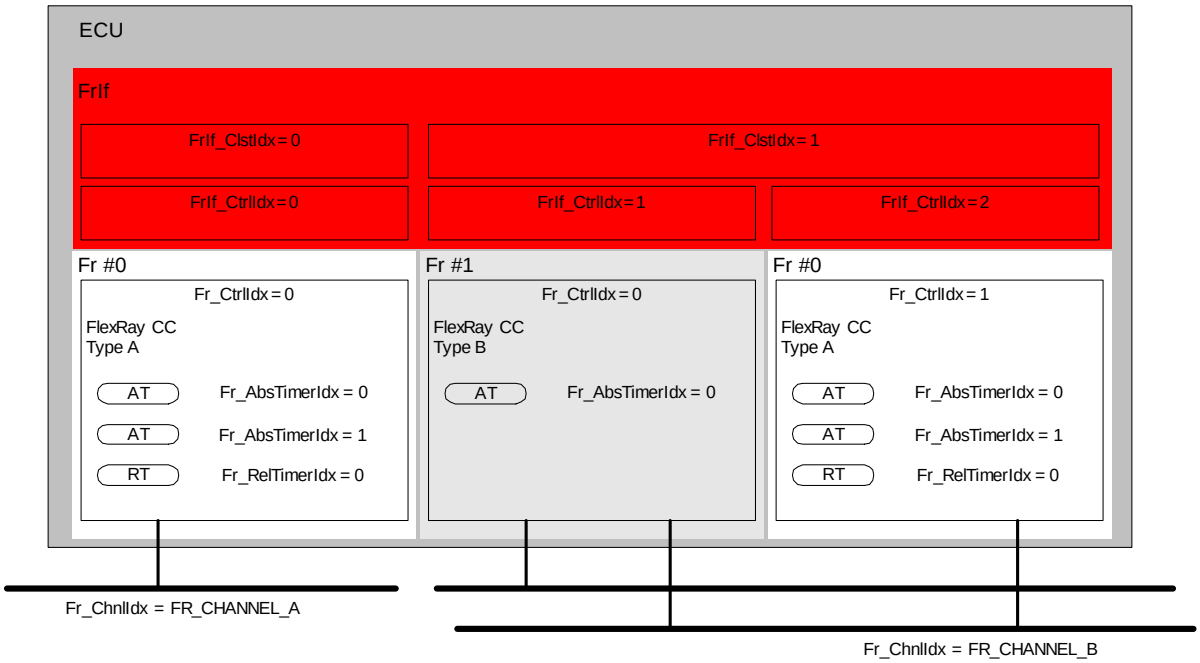
Figure 7-2 and Figure 7-3.

FrIf05052:
The FrIf achieves the abstraction (of the CCs and Drivers) by providing to the upper layer BSW modules an abstract, unique, zero-based consecutive index[3] for each sort of resource, independent of their type, location, and access method.

FrIf05103:
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.

FrIf05104:
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.



[2] FlexRay Drivers, FlexRay Communication Controllers, FlexRay Transceiver Drivers, and FlexRay Transceivers
[3] Like ControllerIndex, ClusterIndex, ChannelIndex, etc.
[4] Since this table contains function pointers to the lower layer BSW modules' API services, it obviously has to be linked against the linked and located code of the lower layer BSW modules.

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

FrIf05060:
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, which maps to a tuple of FlexRay Driver API Service function pointer and CC index Fr_CtrlIdx.

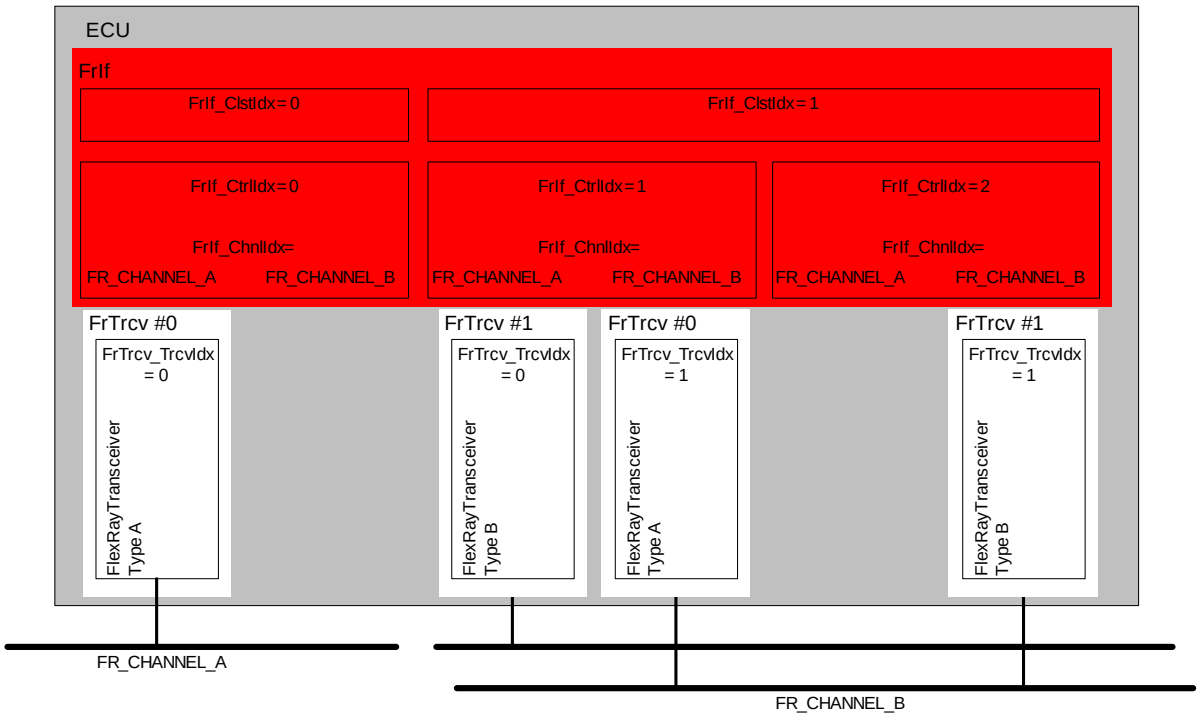


Figure 7-3: Flexray Transceiver Indexing Scheme of the FlexRay Interface

FrIf05105:
In order to abstract for upper layer BSW modules the various FlexRay Transceivers, which the FrIf accesses 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.

FrIf05106:
Therefore, the FrIf abstracts the various FlexRay Transceivers by **a combination** of the two indices FrIf_CtrlIdx (Controller Index) and FrIf_ChnlIdx (Channel Index) and maps this to a tuple of FlexRay Transceiver Driver API Service function pointer and FlexRay Transceiver index FrTrcv_TrcvIdx. (Transceiver Index)

FrIf05107:
Besides hardware and software resources, the FrIf also numbers the logical structure elements presented by FlexRay with an abstract, unique, zero-based consecutive index.

FrIf05108:

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

FrIf05109:
The static configuration data of the FrIf contains a data structure that specifies which FlexRay CC and which FlexRay Transceivers are connected to which Clusters, or in other words, that maps each value of FrIf_ClstIdx to (one, or in general) a set of values for FrIf_CtrlIdx and tuples of (FrIfCtrlIIdx, FrIf_ChnlIdx).

FrIf05110:
The FrIf numbers all PDUs to be transmitted with an abstract, unique, zero-based consecutive index FrIf_TxPduId.

Note:
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

FrIf05057:
The FrIf can be configured to support at least four (possibly different) **FlexRay Drivers** to access the FlexRay Communication Controllers.

FrIf05053:
The FrIf can be configured to support at least four (possibly different) **FlexRay CCs**.

FrIf05111:
The FrIf can be configured to support one of both or both **FlexRay Channels** as specified in [16].

FrIf05112:
The FrIf can be configured to support at least four **FlexRay Clusters**.

FrIf05113:
The FrIf can be configured to support at least one **absolute timer** per FlexRay CCs.

FrIf05114:
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 to control the communication operations of the FlexRay system,

FrIf05115:
the FrIf implements a simple state machine (one per FlexRay cluster), called **FlexRay Interface State Machine**

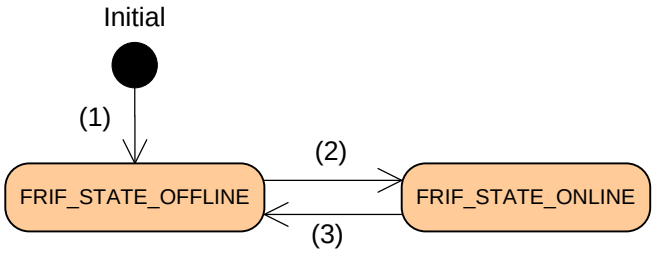


Figure 7-4: FlexRay Interface State Machine

Figure 7-4 shows the states and transitions that are visible to the user of the FrIf. The two different states (FrIf_State) represent the communication capabilities of the FrIf.

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

FrIf05116:
During initialization of the FrIf by executing FrIf_Init() (transistion (1)) all CCs shall be offline (CC is brought to the POC-State 'Ready')

FrIf05117:
and the FrIf_State for each cluster is initialized with state 'FRIF_STATE_OFFLINE'.

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

The following table describes the transistion names that can be passed to FrIf_SetState().

Transistion Name	***Transitions*** (***see*** Figure 7-4)	***Description***
`FRIF_GOTO_ONLINE`	(2)	Transition resulting in FrIf_State FRIF_STATE_ONLINE

Transistion Name	*Transitions (see* Figure 7-4*)*	*Description*
FRIF_GOTO_OFFLINE	(3)	Transition resulting in FrIf_State FRIF_STATE_OFFLINE

7.3.1 FlexRay Interface Main Function

FrIf05118:
The FlexRay Interface Main Function needs to be called cyclically from a task body provided by the BSW Scheduler with a calling period depending on the FlexRay Cycle length and configurable at system configuration time.

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

FrIf05119:
there shall be one dedicated FlexRay Interface Main Function for each FlexRay Cluster that is controlled by the FlexRay Interface.

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

- 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.

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

This is done by the following steps:

FrIf05120:
The FlexRay Interface Main Function performs the following tasks at each invocation:

- Get the global time (FrIf_GetGlobalTime())

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

7.4 Implementation Requirements

FrIf05068:
Link-time and post-build-time configuration data shall be implemented as read-only data structures. Link-time configuration data shall be immediately referenced by the implementation, the start-address of post-build-time configuration data shall be passed during module initialization

FrIf05069:
The FrIf module shall support pre-compile time, link-time and post-build-time configuration.

FrIf05078:
The FrIf module shall implement the API functions specified by the Fr SWS as real C code functions and shall not implement the API functions as macros.

The rationale is to allow object code module integration.

FrIf05079:
The description of the configuration and initialization data itself is not part of this specification but very implementation specific. The generated configuration data should be “human-readable”.

FrIf05080:
The FrIf module’s implementation shall conform to the HIS subset of the MISRA C Standard.

FrIf05085:
In case development error detection is enabled for the FrIf module: the FrIf module shall check API parameters for validity and report detected errors to the DET.

FrIf05087:
The FrIf module source code file(s) shall include *SchM_FrIf.h* if data consistency mechanisms of the BSW scheduler are required as described in [13].

FrIf05088:
The FrIf module header file shall include *MemMap.h* and apply the memory mapping abstraction mechanisms as specified by [15].

FrIf05090:
The header file *FrIf.h* shall contain a software and specification version number.

FrIf05092:
The FrIf shall support dynamic payload length for LPdus whose associated parameter FrIfAllowDynamicLSduLength is set to true.

FrIf05143:
None of the FrIf module’s header files shall define global variables.

7.5 Configuration description

FrIf05089:
The FrIf module shall provide an XML file that contains the data which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.

7.6 Data Communication via FlexRay

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

FrIf05067:
Each datum that should be transmitted or received has to be scheduled at system configuration time. This even holds true for data that - from the application's point of view - are considered *event-driven*.

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

FrIf05054:
However, the resources (e.g. a buffer in the FlexRay Communication Controller or FlexRay Driver) needed for this data transmission (or reception, respectively) have to be defined at system configuration time specifically for this data transmission (or reception, respectively).

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

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

In accordance with basic AUTOSAR rules,

FrIf05055:
the API services that the FrIf provides to upper layer BSW modules for data transmission and data reception are PDU-based.

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

FrIf05121:
the FrIf shall be capable of packing multiple PDUs into one FlexRay Frame.

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

FrIf05122:

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

FrIf05123:
The Frame Construction Plan shall be stored in the static configuration of the FrIf (configuration parameter FrIfFrameStructure).

FrIf05124:
In cases where multiple PDUs are packed into a single FlexRay Frame, this FlexRay Frame has to be transmitted if at least one of the contained PDUs has been updated.

Note:
As a result, those PDUs in this FlexRay Frame that have not been updated by the upper layer BSW module will also be transmitted.

FrIf05056:
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 the upper layer BSW module (by a call of FrIf_Transmit()) on the transmitter side, additional update information, so called **PDU Update-Bits** within the FlexRay Frame, shall be transmitted to the receiving FrIf.

FrIf05125:
For each PDU, a dedicated PDU Update-Bit in the FlexRay Frame can be configured (see configuration parameter FrIfPduUpdateBitOffset).

FrIf05126:
This PDU Update-Bit can be located at an arbitrary bit position in the Frame Construction Plan that is not occupied by any PDU.

FrIf05127:
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]

FrIf05128:
If no update bit is configured for a specific PDU, this PDU is always assumed to be valid and its receiption is always indicated to the upper layer BSW module on the receiver side.

FrIf05129:
If Transmission with Immediate Buffer Access is used, only one PDU is allowed per FlexRay Frame (L-SDU).

Note:
Therefore, PDU Update-Bits can be omitted for Transmission with Immediate Buffer Access.

7.6.1.1 AlwaysTransmit

Note:
According to [16], a FlexRay CC might **only** support the so-called "continuous transmission mode" where a message is transmitted continuously until the host explicitly invalidates the transmit buffer. If such a FlexRay CC is being used for transmission, and the receiving 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 [4], the FlexRay Driver's API service Fr_TransmitTxLPdu() is always called for this L-PDU, independent from any PDUs in this L-PDU having been updated by an upper layer BSW module. This enables resetting the PDU Update-Bits in the FlexRay CC's transmit buffer, even if none of the PDUs in the FlexRay Frame have actually been updated by an upper layer BSW module, and thus ensures the correct interpretation of the received Frame contents by the receiving FrIf.

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.

7.6.2 Realization of the Time-Driven FlexRay Schedule

According to [16], a FlexRay CC is **not** required to provide mechanisms in hardware to ensure asynchronous access to its transmit and receive buffers e.g. by providing shadow buffers that may be accessed asynchronously by the AUTOSAR FlexRay software modules.

FrIf05130:
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[5] and thus avoid concurrent access to the buffers by the hardware and the software.

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

[5] In FlexRay Global Time

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

7.6.2.1 FlexRay Job List

FrIf05131:
A FlexRay Job List is a list of Communication Jobs sorted according to their respective execution start time.

FrIf05132:
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 configurable Communication operation index [Configuration Parameter FrIfCommunicationOperationIdx]. The sorting order defines the order of execution of the Communication Operations within a FlexRay Communication Job.

FrIf05133:
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.

FrIf05134:
The Communication Operations specify the actions to process within the Communication Job.

FrIf05135:
Each Communication Operation contains the following properties:

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

FrIf05059:

7.6.2.2 FlexRay Job List Execution Function

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

FrIf05136:
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.

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

FrIf05137:
The FlexRay Job List Execution Function executes the Cluster's FlexRay Job List Jobs synchronously to the Cluster's global time (i.e. at well-defined points in time).

FrIf05138:
Upon invocation, the FlexRay Job List Execution Function performs the following steps:

1. Retrieve the FlexRay Global Time from the FlexRay CC providing the Cluster's absolute timer interrupt.
2. If the FlexRay Global Time cannot be retrieved or the global time delay compared to the jobs start time is larger than a maximum delay [Configuration Parameter 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 a flag (JobListAsyncFlag) indicating that the execution of the FlexRay Job List of this Cluster is asynchronous
   - Call Dem_ReportErrorStatus(FRIF_E_JLE_SYNC, DEM_EVENT_STATUS_FAILED)
   - Disable absolute Timer Interrupt
   - Terminate the execution of this FlexRay Job.

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

Note:

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

FrIf05093:
In case the parameter 'FrIfAllowDynamicLSduLength' is set to true for the associated frame triggering, the actual used L-PDU length shall be passed to the driver (Fr_TransmitTxLPdu()) taking into account the following for each PDU:
- the position of the PDU within the L-PDU
- the actual PDU length passed via FrIf_Transmit()

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

FrIf05094:
In case the parameter 'FrIfAllowDynamicLSduLength' is set to true for the associated frame triggering for reception, PDUs in non received areas (PDU offset > actual L-PDU length) shall not be indicated to upper layer(s).

7.6.3 Communication Operations

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

FrIf05058:

7.6.3.1 TransmitWithDecoupledBufferAccess

FrIf05063:

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

For Communication Operation DECOUPLED_TRANSMISSION the following steps are performed:

1. Iterate over all PDUs contained in the FrIfFrameStructure of the associated frame triggering of this Comunication Operation and
2. Check whether TrigTxCounter is > 0 or FrIfNoneMode == true for the PDU. If not, clear the update-bit for this PDU [Configuration Parameter FrIfPduUpdateBitOffset] and proceed with the next PDU.
3. 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.
4. Decrement TrigTxCounter only if TrigTxCounter > 0.
5. Remember that a transmission for this PDU is pending if a transmission confirmation is needed for this PDU [Configuration Parameter FrIfConfirm] (increment TxConfCounter[7]).
6. Set the update-bit if configured for this PDU [Configuration Parameter FrIfPduUpdateBitOffset] only in case the API <UL>_TriggerTransmit() returned E_OK for the corresponding PDU.
7. 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] or FrIfNoneMode == true, 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.

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

   d. Fr_LSduLength is set to the L-SDU length [Configuration Parameter FrIfLSduLength]
8. In case the Driver's API Fr_TransmitTxLPdu() returned E_NOT_OK (indicating that the transmission failed) **changes** on TrigTxCounter and TxConfCounter must be rolled back (see 4. and 5.) for each PDU contained in the FlexRay L-SDU.

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

7.6.3.2 ProvideTxConfirmation

FrIf05064:

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

7.6.3.3 ReceiveAndStore

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

For Communication Operation RECEIVE_AND_STORE the following steps are performed:

1. Call the FlexRay Driver's API function Fr_ReceiveRxLPdu():
   a. Fr_CtrlIdx is derived according to the indexing scheme 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 a 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.

7.6.3.4 ProvideRxIndication

FrIf05062:

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

For Communication Operation RX_INDICATION the following steps are performed:

1. Iterate over all PDU-related static buffers of PDUs contained in the 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.

7.6.3.5 ReceiveAndIndicate

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

For Communication Operation RECEIVE_AND_INDICATE the following steps are performed:

1. Call the FlexRay Driver's API function Fr_ReceiveRxLPdu():
    a. Fr_CtrlIdx is derived according to the indexing scheme 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.

7.6.3.6 PREPARE_LPDU

FrIf05061:

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 function 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 hardware 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.

7.6.4 Transmission with Immediate Buffer Access

The FlexRay Job List Execution Function does not initiate transmission with immediate buffer access. Instead, the actions described here are carried out in the context of the 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.

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 function 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.

7.7 Error Classification

FrIf05139:

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.

FrIf05142:
The error values and EventIds are named in capital letters according to the scheme `FRIF_E_<NAME>`, where NAME describes the error/EventId and may consist of several words separated by underscores.

Development error values are of type uint8.

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 FrIf_TxPdu Index	Development	`FRIF_E_INV_TXPDUID`	`0x06`
FrIf not initialized	Development	`FRIF_E_NOT_INITIALIZED`	`0x08`
Invalid Controller State	Development	`FRIF_E_INV_FRIF_CC_STATE`	`0x09`
Job List Execution lost synchronization to the FlexRay Global Time	Production	`FRIF_E_JLE_SYNC`	`Assigned by DEM`

Table 7-1: Definition of Error Codes

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

7.8 Error Detection

FrIf05084:

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

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

If the FRIF_DEV_ERROR_DETECT switch is set to ON, all 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().

7.9 Error Notification

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

Production errors shall be reported to the DEM.

8 API Service Specification

FrIf05083:
All AP functions or global variables, whether they are specified or not shall follow the naming scheme `FrIf_<name>`, where the first letter of each word in <name> is written uppercase and the remainder of the word lowercase.

8.1 Imported types

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

FrIf05001:

Header file	***Imported Type***
Dem_Types.h	Dem_EventIdType
PrimitiveTypes.h	PduInfoType
FrTrcv_Types.h	FrTrcv_TrcvModeType
	FrTrcv_TrcvWUReasonType
Fr_Types.h	Fr_POCStatusType
	Fr_ChannelType
	Fr_RateCorrectionType
	Fr_OffsetCorrectionType
	Fr_SyncStateType
ComStack_Types.h	PduIdType
	BusTrcvErrorType
Std_Types.h	Std_ReturnType
	Std_VersionInfoType

8.2 Type Definitions

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

FrIf05082:
All types whether they are specified or implementation dependant shall follow the naming scheme `FrIf_<name>Type`, where the first letter of each word in <name> is written uppercase and the remainder of the word is written lowercase.

8.2.1 FrIf_ConfigType

Name:	`FrIf_ConfigType`	
Type:	`Structure`	
Range:	`Implementation specific`	--
Description:	This type contains the implementation-specific post build time configuration structure. Only pointers of this type are allowed.	

8.2.2 FrIf_StateType

Name:	FrIf_StateType	
Type:	Enumeration	
Range:	FRIF_STATE_OFFLINE	The FlexRay CC is not ready for communication, the FlexRay cluster is not synchronized.
	FRIF_STATE_ONLINE	The FlexRay CC is ready for communication, the FlexRay cluster is synchronized.
Description:	Variables of this type are used to represent the FrIf_State of a FlexRay CC.	

8.2.3 FrIf_StateTransitionType

Name:	FrIf_StateTransitionType	
Type:	Enumeration	
Range:	FRIF_GOTO_OFFLINE	Literal for requesting transition into FRIF_STATE_OFFLINE
	FRIF_GOTO_ONLINE	Literal for requesting transition into FRIF_STATE_ONLINE state.
Description:	Variables of this type are used to represent the FrIf_State of a FlexRay CC.	

8.3 Function Definitions

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

8.3.1 FrIf_GetVersionInfo

FrIf05002:

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 (inout):	None	
Parameters (out):	FrIf_VersionInfoPtr	Pointer to a memory location where the FlexRay Interface version information shall be stored.
Return value:	None	
Description:	Returns the version information of this module.	

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

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

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

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

Configuration:
If pre-compile-time configuration parameter ‘FRIF_VERSION_INFO_API’ is ‘ON’ this API function is included in the compilation process.
If pre-compile-time configuration parameter ‘FRIF_VERSION_INFO_API’ is ‘OFF’ this API function is excluded from the compilation process.

8.3.2 FrIf_Init

FrIf05003:

Service name:	FrIf_Init

Syntax:	`void FrIf_Init(`
	`  const FrIf_ConfigType* FrIf_ConfigPtr`
	`)`

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

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

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

This initialization is carried out by the following actions:

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

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

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

8.3.3 FrIf_ControllerInit

FrIf05004:

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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Initialized a FlexRay CC.	

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

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

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

Setting parameters Fr_LowLevelConfSetIdx and Fr_BufConfSetIdx to 0.

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].

8.3.4 FrIf_StartCommunication

FrIf05005:

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.

Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_StartCommunication().	

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

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

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

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

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

8.3.5 FrIf_HaltCommunication

FrIf05006:

Service name:	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_HaltCommunication().	

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

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

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

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

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

8.3.6 FrIf_AbortCommunication

FrIf05007:

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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_AbortCommunication().	

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

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

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

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

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

8.3.7 FrIf_GetState

Service name:	FrIf_GetState	
Syntax:	`Std_ReturnType FrIf_GetState ( uint8 FrIf_ClstIdx, FrIf_StateType *FrIf_StatePtr )`	
Service ID[hex]:	0x35	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrIf_ClstIdx	Index of the cluster addressed.
Parameters (inout):	None	
Parameters (out):	FrIf_StatePtr	Pointer to a memory location where the retrieved FrIf_State will be stored.
Return value:	Std_ReturnType	E_OK: Function was successfully executed. State transition request was accepted. E_NOT_OK: Function execution failed due to detected errors. State transition request was not accepted.
Description:	Get current FrIf state.	

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

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

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

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

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

8.3.8 FrIf_SetState

Service name:	FrIf_SetState
Syntax:	`Std_ReturnType FrIf_SetState( FrIf_StateTransitionType FrIf_StateTransition, uint8 FrIf_ClstIdx )`
Service ID[hex]:	0x35
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

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

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

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

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

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

8.3.9 FrIf_SetWakeupChannel

FrIf05010:

Service name:	FrIf_SetWakeupChannel	

Syntax:	`Std_ReturnType FrIf_SetWakeupChannel(`
	`  uint8 FrIf_CtrlIdx,`
	`  Fr_ChannelType FrIf_ChnlIdx`
	`)`	

Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_SetWakeupChannel().	

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

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

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

3. Setting parameters
   - Fr_ChnlIdx to FrIf_ChnlIdx

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

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

8.3.10 FrIf_SendWUP

FrIf05011:

Service name:	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.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_SendWUP().	

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

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

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

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

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

8.3.11 FrIf_GetSyncState

FrIf05012:

Service name:	FrIf_GetSyncState	
Syntax:	`Std_ReturnType FrIf_GetSyncState( uint8 FrIf_CtrlIdx, Fr_SyncStateType* FrIf_SyncStatePtr )`	
Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout):	None	
Parameters (out):	FrIf_SyncStatePtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_GetSyncState()	

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

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

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

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

6. Setting parameters
   - Fr_SyncStatePtr to FrIf_SyncStatePtr

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

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

8.3.12 FrIf_SetExtSync

FrIf05013:

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.
	FrIf_Offset	Offset correction that shall be applied
	FrIf_Rate	Rate correction that shall be applied
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_SetExtSync().	

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

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

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

8. Setting parameters
   - Fr_Rate to FrIf_Rate
   - Fr_Offset to FrIf_Offset
- Calling Fr_SetExtSync() of the determined FlexRay Driver with the parameters determined as described above.

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

8.3.13 FrIf_GetPOCStatus

FrIf05014:

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.
Parameters (inout):	None	
Parameters (out):	FrIf_POCStatusPtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_GetPOCStatus().	

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

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

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

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

9. Setting parameters
   - Fr_POCStatusPtr to FrIf_POCStatusPtr
10. Calling Fr_GetPOCStatus() of the determined FlexRay Driver with the parameters determined as described above.

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

8.3.14 FrIf_GetGlobalTime

FrIf05015:

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.
Parameters (inout):	None	
Parameters (out):	FrIf_CyclePtr	Pointer to a memory location where output value will be stored.
	FrIf_MacroTickPtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_GetGlobalTime().	

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

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

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

11. Setting parameters

- Fr_CylcePtr to FrIf_CyclePtr
- Fr_MacroTickPtr to FrIf_MacroTickPtr

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

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

8.3.15 FrIf_AllowColdstart

FrIf05017:

Service name:	FrIf_AllowColdstart

Syntax:	`Std_ReturnType FrIf_AllowColdstart(`
	`  uint8 FrIf_CtrlIdx`
	`)`

Service ID[hex]:	0x35
Sync/Async:	Asynchronous

Reentrancy:	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_AllowColdstart().	

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

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

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

13. Calling Fr__AllowColdstart() of the determined FlexRay Driver

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

8.3.16 FrIf_GetMacroticksDuration

FrIf0501x:

Service name:	FrIf_GetMacroticksDuration	

Syntax:	`uint16 FrIf_GetMacroticksDuration(`
	`  uint8 FrIf_CtrlIdx`
	`)`	

Service ID[hex]:	0x1b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	uint16	Number of nanoseconds of one Macrotick
Description:	Retrieves the duration of one Macrotick in nanoseconds	

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

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

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

8.3.17 FrIf_Transmit

FrIf05033:

Service name:	FrIf_Transmit	
Syntax:	`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.
	FrIf_PduInfoPtr	Pointer to a structure with FlexRay PDU related data.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: No error has occurred during the execution of this API service. E_NOT_OK: An error occurred during execution of this API service: • FlexRay Driver reported an error in case of immediate transmission • An error has been detected in development mode
Description:	Requests the sending of a PDU.	

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

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

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

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

In case of decoupled transmission the PDU with index FrIf_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].

8.3.18 FrIf_SetTransceiverMode

FrIf05034:

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.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
	FrIf_TrcvMode	Transceiver mode to be set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode().	

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

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

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

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

1. Translating (based on static FrIf configuration) the tuple (FlexRay CC index FrIf_CtrlIdx | FlexRay Channel index FrIf_ChnlIdx) into a tuple (FlexRay Transceiver Driver | Driver-specific Transceiver index FrTrcv_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].

8.3.19 FrIf_GetTransceiverMode

FrIf05035:

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.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	FrIf_TrcvModePtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverMode().	

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

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

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

function FrTrcv_GetTransceiverMode() by:

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

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

8.3.20 FrIf_GetTransceiverWUReason

FrIf05036:

Service name:	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.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	FrIf_TrcvWUReasonPtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverWUReason()	

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

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

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

1. Translating (based on static 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].

8.3.21 FrIf_EnableTransceiverWakeup

FrIf05037:

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.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverWakeup().	

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

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

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

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

Calling FrTrcv_EnableTransceiverWakeup() of the determined FlexRay Driver with

the parameters determined as described above.

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

8.3.22 FrIf_DisableTransceiverWakeup

FrIf05038:

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.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_DisableTransceiverWakeup().	

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

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

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

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

Calling FrTrcv_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].

8.3.23 FrIf_ClearTransceiverWakeup

FrIf05039:

Service name:	FrIf_ClearTransceiverWakeup	
Syntax:	`Std_ReturnType FrIf_ClearTransceiverWakeup( uint8 FrIf_CtrlIdx, Fr_ChannelType FrIf_ChnlIdx )`	
Service ID[hex]:	0x30	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_NO_ERROR. E_NOT_OK: The call of the FlexRay Transceiver Driver's API service has returned BUSTRCV_E_ERROR.
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverWakeup().	

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

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

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

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

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

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

8.3.24 FrIf_GetCycleLength

Service name:	FrIf_GetCycleLength	

Syntax:	`uint32 FrIf_GetCycleLength(`
	`  uint8 FrIf_CtrlIdx`
	`)`	

Service ID[hex]:	0x3a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	uint32	Time in unit of nanoseconds
Description:	This API returns the configured time of the configuration parameter "GdCycle" in nanoseconds for the FlexRay controller with index FrIf_CtrlIdx.	

FrIf05237: If parameter FrIf_CtrlIdx of FrIf_GetCycleLength has an invalid value and if development error detection is enabled (i.e. FRIF_DEV_ERROR_DETECT equals ON), the function FrIf_GetCycleLength shall report development error code FRIF_E_INV_CTRL_IDX to the Det_ReportError service of the DET module.

FrIf05238: Caveats of FrIf_GetCycleLength: The FlexRay Interface module has to be initialized with a call of FrIf_Init() before this API service may be called, see FrIf05002.

8.4 Optional Function Definitions

8.4.1 FrIf_SetAbsoluteTimer

FrIf05021:

Service name:	FrIf_SetAbsoluteTimer	
Syntax:	`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.
	FrIf_AbsTimerIdx	Index of the absolute timer to address.
	FrIf_Cycle	FlexRay Cycle number to be set.
	FrIf_Offset	Number of Macroticks to be set.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_SetAbsoluteTimer().	

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

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

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

2. Setting parameters
   - Fr_AbsTimerIdx to FrIf_AbsTimerIdx
   - Fr_Cycle to FrIf_Cycle
   - Fr_Offset to FrIf_Offset
3. Calling Fr_SetAbsoluteTimer() of the determined FlexRay Driver with the parameters determined as described above.

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

8.4.2 FrIf_SetRelativeTimer

FrIf05022:

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.
	FrIf_RelTimerIdx	Index of the relative timer to address.
	FrIf_Offset	Number of Macroticks the relative timer shall be set to.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_SetRelativeTimer().	

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

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

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

Setting parameters

- Fr_RelTimerIdx to FrIf_RelTimerIdx
- Fr_Offset to FrIf_Offset

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

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

8.4.3 FrIf_CancelAbsoluteTimer

FrIf05023:

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

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

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

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

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

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

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

8.4.4 FrIf_CancelRelativeTimer

FrIf05024:

Service name:	FrIf_CancelRelativeTimer	

Syntax:	`Std_ReturnType FrIf_CancelRelativeTimer(`
	`  uint8 FrIf_CtrlIdx,`
	`  uint8 FrIf_RelTimerIdx`
	`)`	

Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_RelTimerIdx	Index of the relative timer to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has

		returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_CancelAbsoluteTimer().	

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

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

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

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

8.4.5 FrIf_EnableAbsoluteTimerIRQ

FrIf05025:

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

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

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

1. Translating (based on static 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].

8.4.6 FrIf_EnableRelativeTimerIRQ

FrIf05026:

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.
	FrIf_RelTimerIdx	Index of the relative timer to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_EnableRelativeTimerIRQ() .	

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

This API service of the FlexRay Interface wraps the FlexRay Driver API function Fr_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].

8.4.7 FrIf_GetAbsoluteTimerIRQStatus

FrIf05027:

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

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

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

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

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

8.4.8 FrIf_GetRelativeTimerIRQStatus

FrIf05028:

Service name:	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.
	FrIf_RelTimerIdx	Index of the relative timer to address.
Parameters (inout):	None	
Parameters (out):	FrIf_IRQStatusPtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_GetRelativeTimerIRQStatus().	

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

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

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

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

8.4.9 FrIf_AckAbsoluteTimerIRQ

FrIf05029:

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

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

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

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

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

8.4.10 FrIf_AckRelativeTimerIRQ

FrIf05030:

Service name:	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.
	FrIf_RelTimerIdx	Index of the relative timer to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_AckRelativeTimerIRQ().	

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

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

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

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

8.4.11 FrIf_DisableAbsoluteTimerIRQ

FrIf05031:

Service name:	FrIf_DisableAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType FrIf_DisableAbsoluteTimerIRQ( uint8 FrIf_CtrlIdx, uint8 FrIf_AbsTimerIdx )`	
Service ID[hex]:	0x26	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_AbsTimerIdx	Index of the absolute timer to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_DisableAbsoluteTimerIRQ().	

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

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

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

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

8.4.12 FrIf_DisableRelativeTimerIRQ

FrIf05032:

Service name:	FrIf_DisableRelativeTimerIRQ
Syntax:	`Std_ReturnType FrIf_DisableRelativeTimerIRQ( uint8 FrIf_CtrlIdx, uint8 FrIf_RelTimerIdx )`
Service ID[hex]:	0x27

Sync/Async:	Synchronous	
Reentrancy:	non reentrant for the same FlexRay CC, reentrant for different FlexRay CCs	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
	FrIf_RelTimerIdx	Index of the relative timer to address.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.
Description:	Wraps the FlexRay Driver API function Fr_DisableRelativeTimerIRQ().	

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

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

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

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

8.4.13 FrIf_GetNmVector

FrIf05016:

Service name:	FrIf_GetNmVector	

Syntax:	`Std_ReturnType FrIf_GetNmVector(`
	`  uint8 FrIf_CtrlIdx,`
	`  uint8* FrIf_CyclePtr,`
	`  uint8* FrIf_NmVectorPtr`
	`)`	

Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	non reentrant for identical values of FrIf_CtrlIdx, reentrant for different values of FrIf_CtrlIdx	
Parameters (in):	FrIf_CtrlIdx	Index of the FlexRay CC to address.
Parameters (inout):	None	
Parameters (out):	FrIf_CyclePtr	Pointer to a memory location where output value will be stored.
	FrIf_NmVectorPtr	Pointer to a memory location where output value will be stored.
Return value:	Std_ReturnType	E_OK: The call of the FlexRay Driver's API service has returned E_OK. E_NOT_OK: The call of the FlexRay Driver's API service has returned E_NOT_OK, or an error has been detected in development mode.

Description:	Derives the FlexRay NM Vector.

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

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

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

14. Setting parameters

- Fr_CylcePtr to FrIf_CyclePtr
- Fr_NmVectorPtr to FrIf_NmVectorPtr

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

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

8.5 Interrupt Service Routines

8.5.1 FrIf_JobListExec_<ClstIdx>

FrIf05040:

Service name:	FrIf_JobListExec_<ClstIdx>

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

Service ID[hex]:	0x32
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Processes the FlexRay Job List of the FlexRay Cluster with index ClstIdx.

This API service of the FlexRay Interface processes the FlexRay Job List of the FlexRay Cluster with index ClstIdx.
For a detailed description of this API service, please refer to chapter 7.6.2.2.

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

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

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

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

8.6 Call-back Notifications

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

8.6.1 FrIf_Cbk_WakeupByTransceiver

FrIf05041:

Service name:	FrIf_Cbk_WakeupByTransceiver	

Syntax:	`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.
	FrIf_ChnlIdx	Index of the FlexRay Channel to address in scope of the FlexRay controller FrIf_CtrlIdx.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Wraps the FlexRay Transceiver Driver API function FrTrcv_Cbk_WakeupByTransceiver().	

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

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

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

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

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

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

8.7 Scheduled Functions

8.7.1 FrIf_MainFunction_<ClstIdx>

FrIf05042:

Service name:	FrIf_MainFunction_<ClstIdx>

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

Service ID[hex]:	0x33
Timing:	VARIABLE_CYCLIC
Description:	This function will be called cyclically by a task body provided by the BSW Scheduler.

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

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

Please refere to chapter 7.3 for a detailed description.

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

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

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

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

8.8 Expected Interfaces

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

8.8.1 Mandatory Interfaces

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

FrIf05043:

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

FrTrcv_GetTransceiverMode	This function returns the actual state of the transceiver.
Dem_ReportErrorStatus	Reports errors to the DEM.

8.8.2 Optional Interfaces

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

FrIf05044:

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

8.8.3 Configurable Interfaces

This chapter lists all interfaces where the target API service of any upper layer, which require one or more of these mentioned interfaces to be called has to be set up by static configuration of the FlexRay Interface.
These call-out services are specified and implemented in the upper layer BSW modules, which use the FlexRay Interface according to [2]. The specific call-out notification is specified in the corresponding AUTOSAR SWS document (see chapter 3).

In addition to upper layer AUTOSAR BSW modules, the 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.

8.8.3.1 <UL_RxIndication>

FrIf05045:

Service name:	<UL_RxIndication>	
Syntax:	void <UL_RxIndication>(PduIdType FrIf_RxPduId, const PduInfoType* PduInfoPtr)	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrIf_RxPduId, Non reentrant for the same FrIf_RxPduId	
Parameters (in):	FrIf_RxPduId	PDU-ID of FlexRay PDU that has been received

	PduInfoPtr	Contains the length (SduLength) of the received I-PDU and a pointer to a buffer (SduDataPtr) containing the I-PDU.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of a received I-PDU from a lower layer communication module.	

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

8.8.3.2 <UL_TxConfirmation>

FrIf05046:

Service name:	<UL_TxConfirmation>	
Syntax:	void <UL_TxConfirmation>(PduIdType FrIf_TxPduId)	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrIf_TxPduId, Non reentrant for the same FrIf_TxPduId	
Parameters (in):	FrIf_TxPduId	PDU-ID of FlexRay PDU whose transmission is being confirmed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This API service of an upper layer BSW module <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.

8.8.3.3 <UL_TriggerTransmit>

FrIf05047:

Service name:	<UL_TriggerTransmit>

Syntax:	Std_ReturnType <UL_TriggerTransmit>(PduIdType FrIf_TxPduId, PduInfoType* PduInfoPtr)	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FrIf_TxPduId, Non reentrant for the same FrIf_TxPduId	
Parameters (in):	FrIf_TxPduId	PDU-ID of FlexRay PDU that shall be copied to the FrIf
Parameters (inout):	PduInfoPtr	Contains a pointer to a buffer (SduDataPtr) to where the SDU shall be copied to. On return, the service will indicate the length of the copied SDU data in SduLength.
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: SDU has been copied and SduLength indicates the number of copied bytes. E_NOT_OK: No SDU has been copied. PduInfoPtr must not be used since it may contain a NULL pointer or point to invalid data.
Description:	The lower layer communication module requests the buffer of the SDU for transmission from the upper layer module.	

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

9 Sequence Diagrams

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

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

9.1 Data Transmission

9.1.1 TransmitWithImmediateBufferAccess

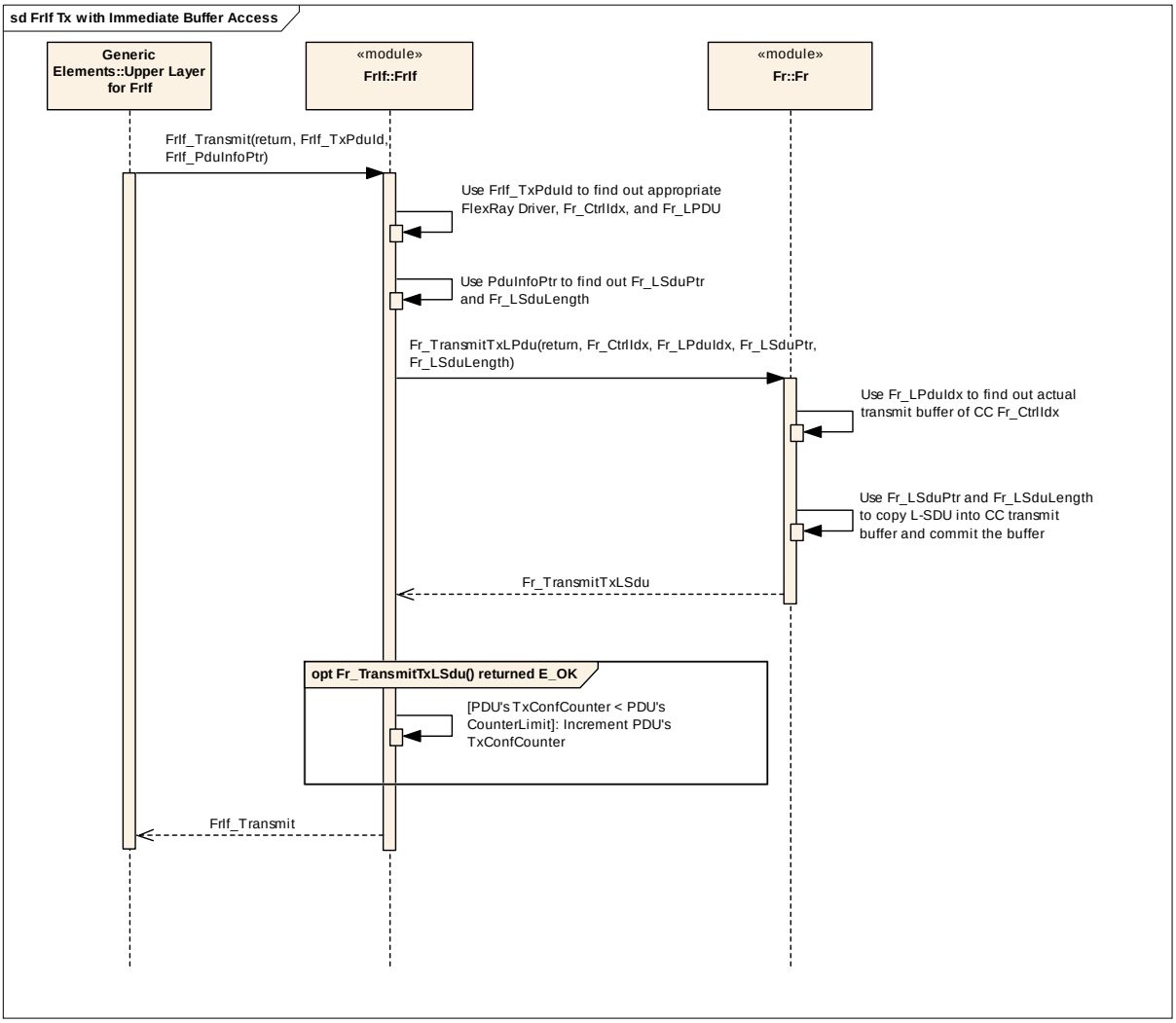


Figure 9-1: TransmitWithImmediateBufferAccess

9.1.2 TransmitWithDecoupledBufferAccess

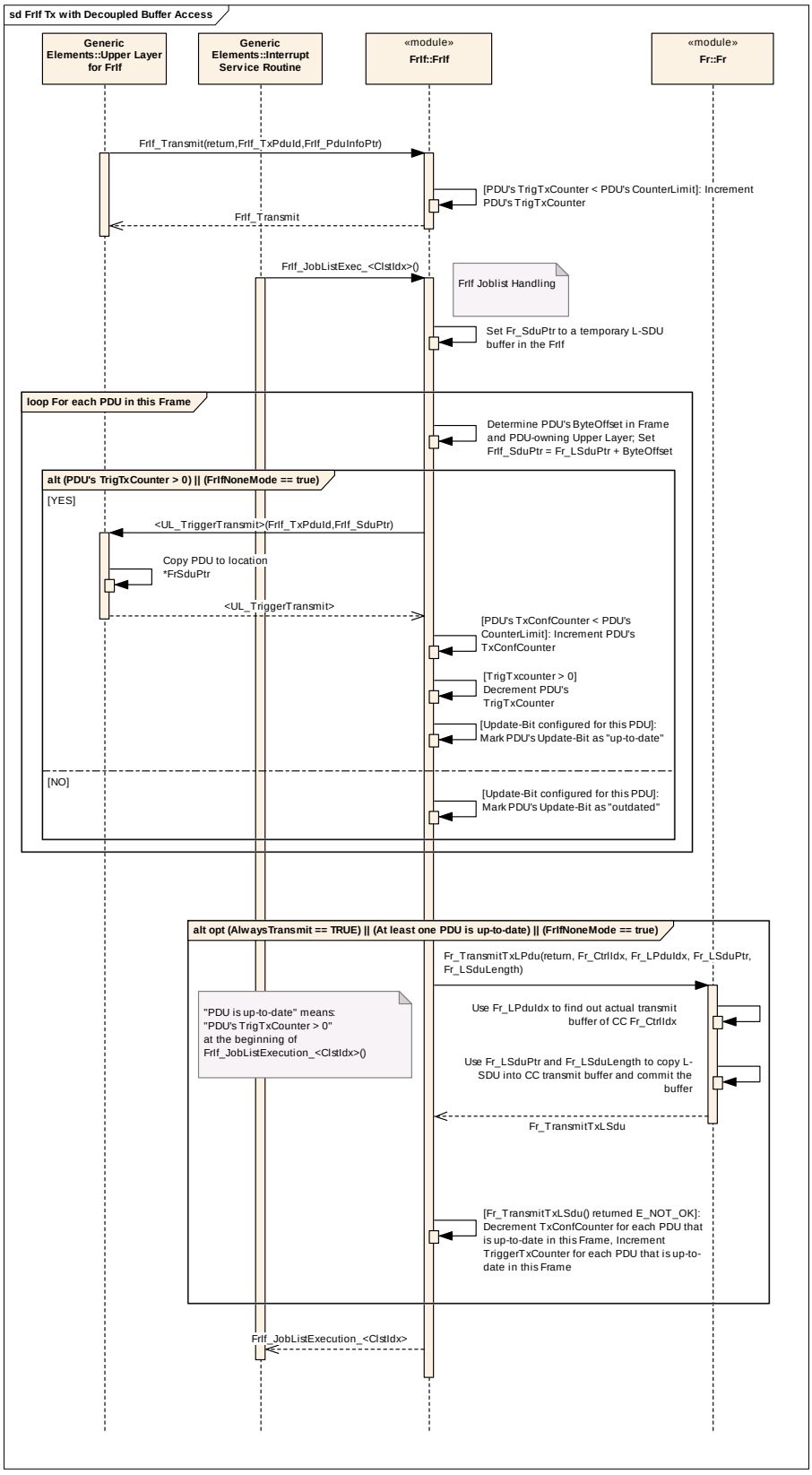


Figure 9-2: TransmitWithDecoupledBufferAccess

9.1.3 ProvideTxConfirmation

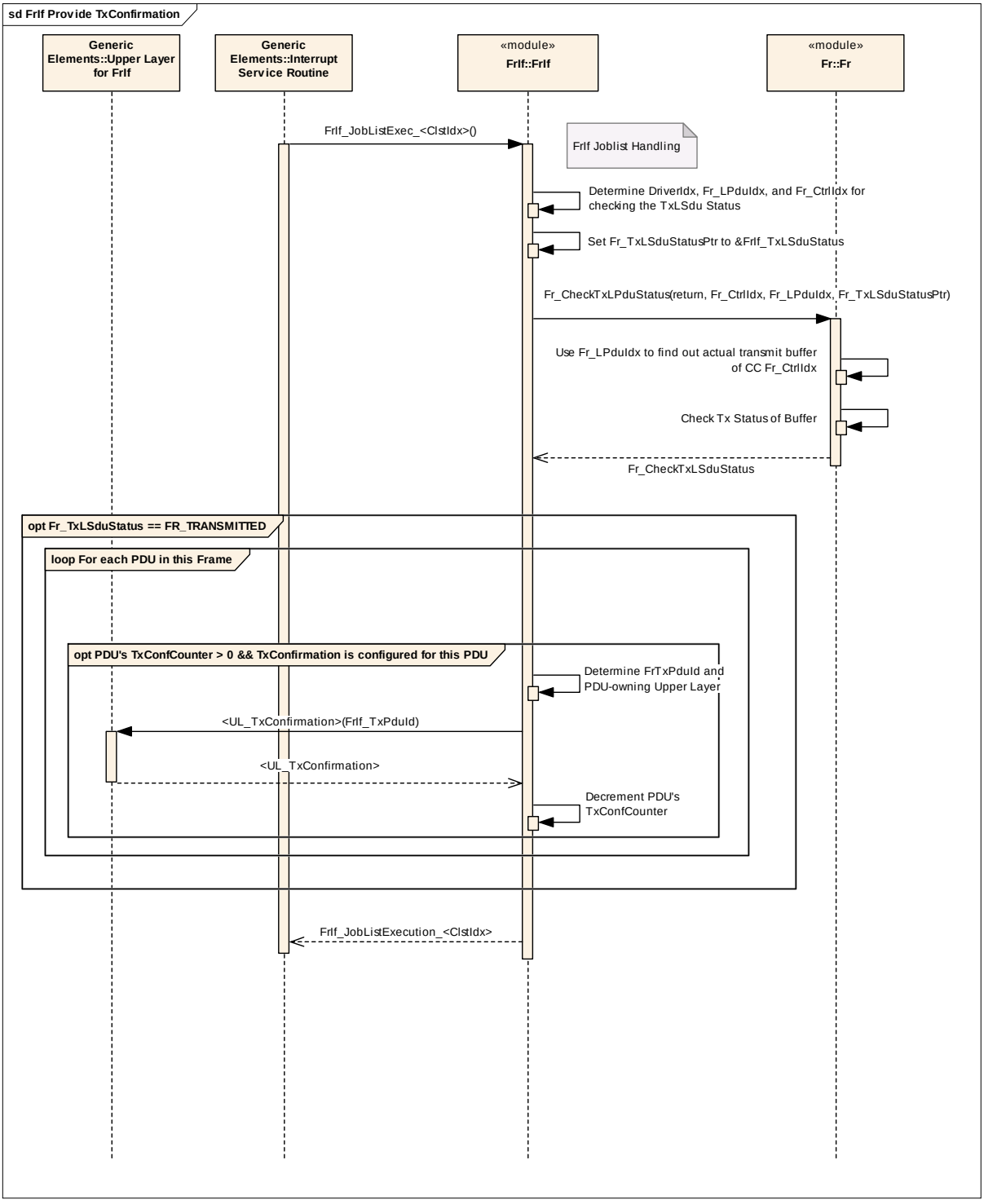


Figure 9-3: ProvideTxConfirmation

9.2 Data Reception

9.2.1 ReceiveAndIndicate

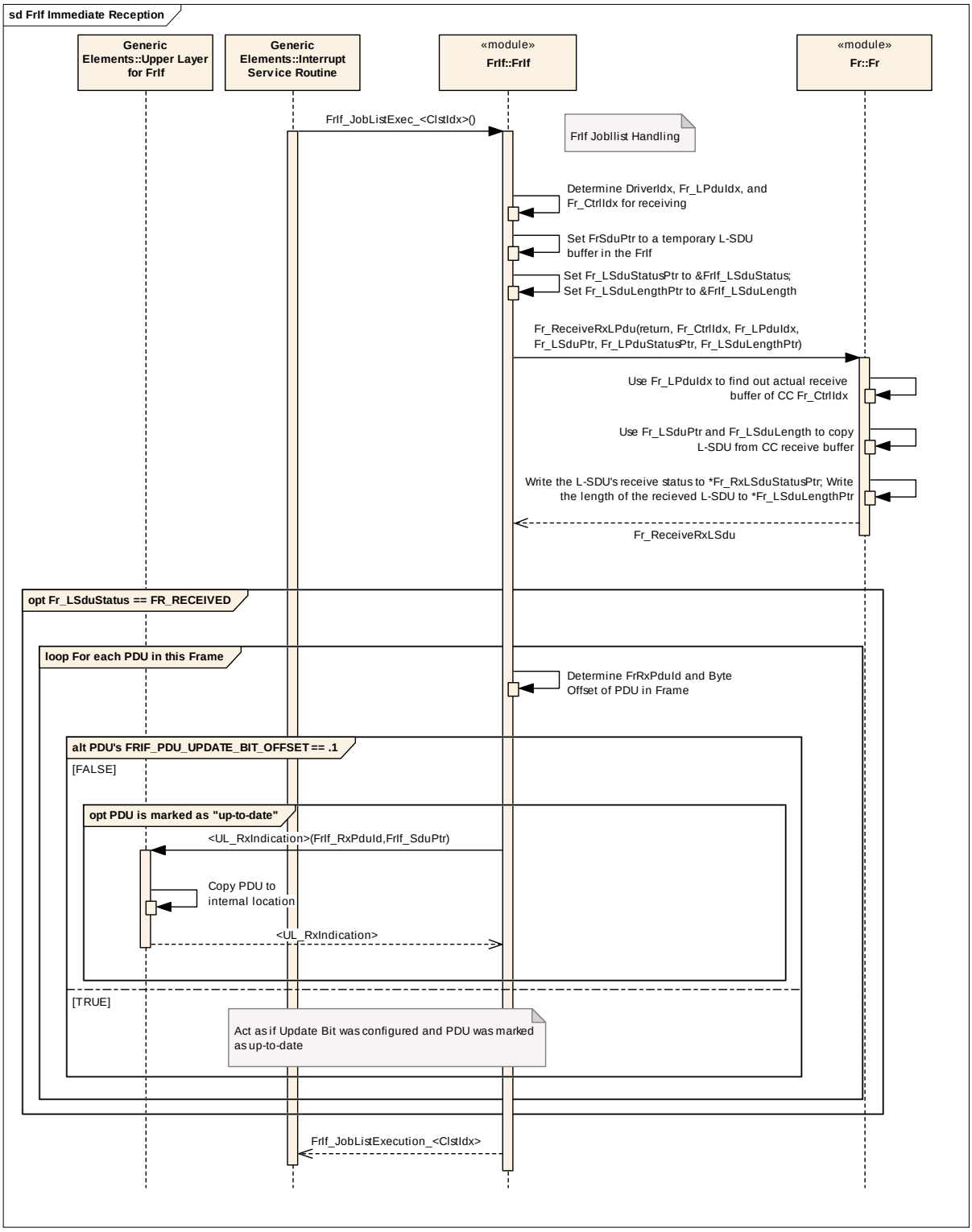


Figure 9-4: ReceiveAndIndicate

9.2.2 ReceiveAndStore

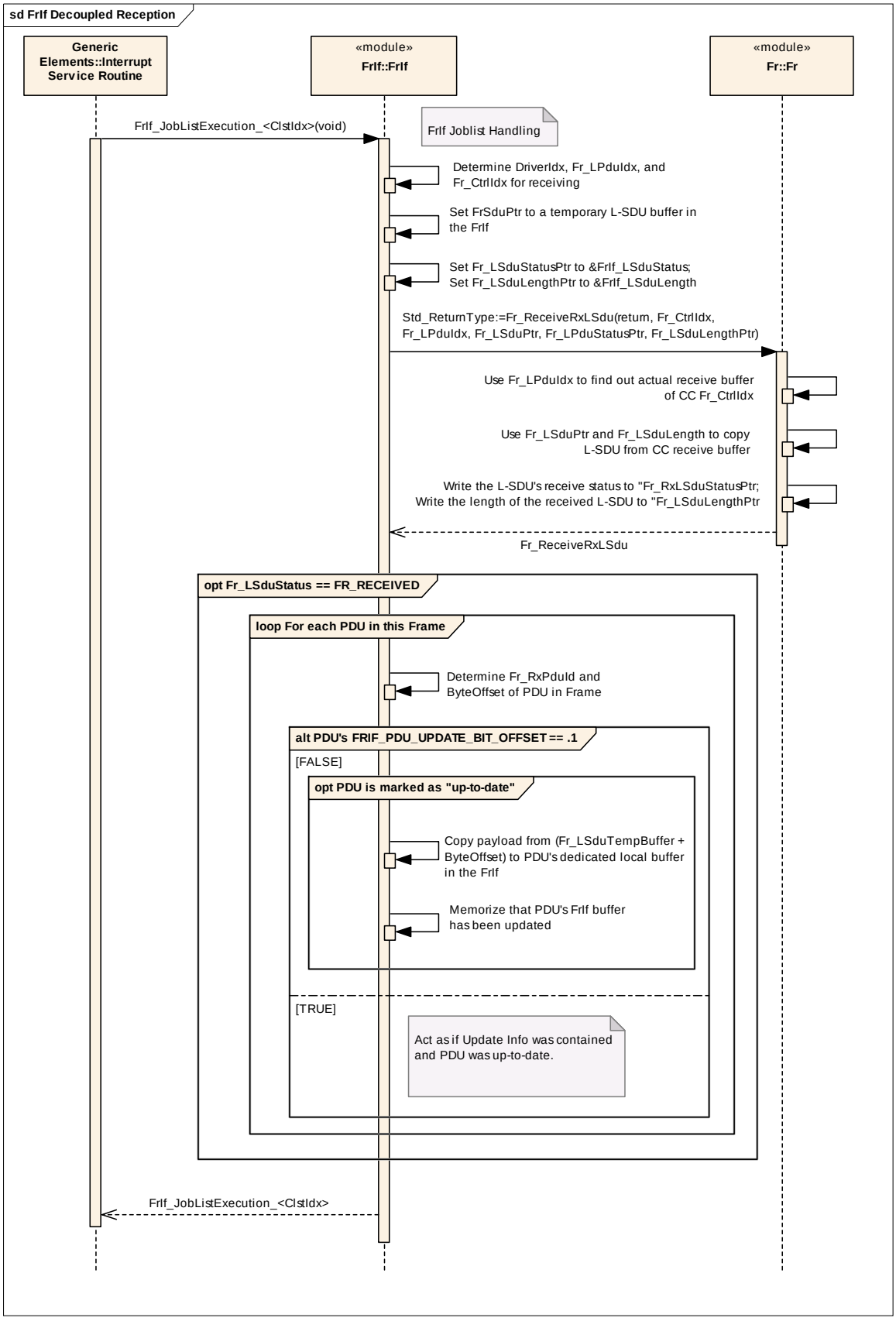


Figure 9-5: ReceiveAndStore

9.2.3 ProvideRxIndication

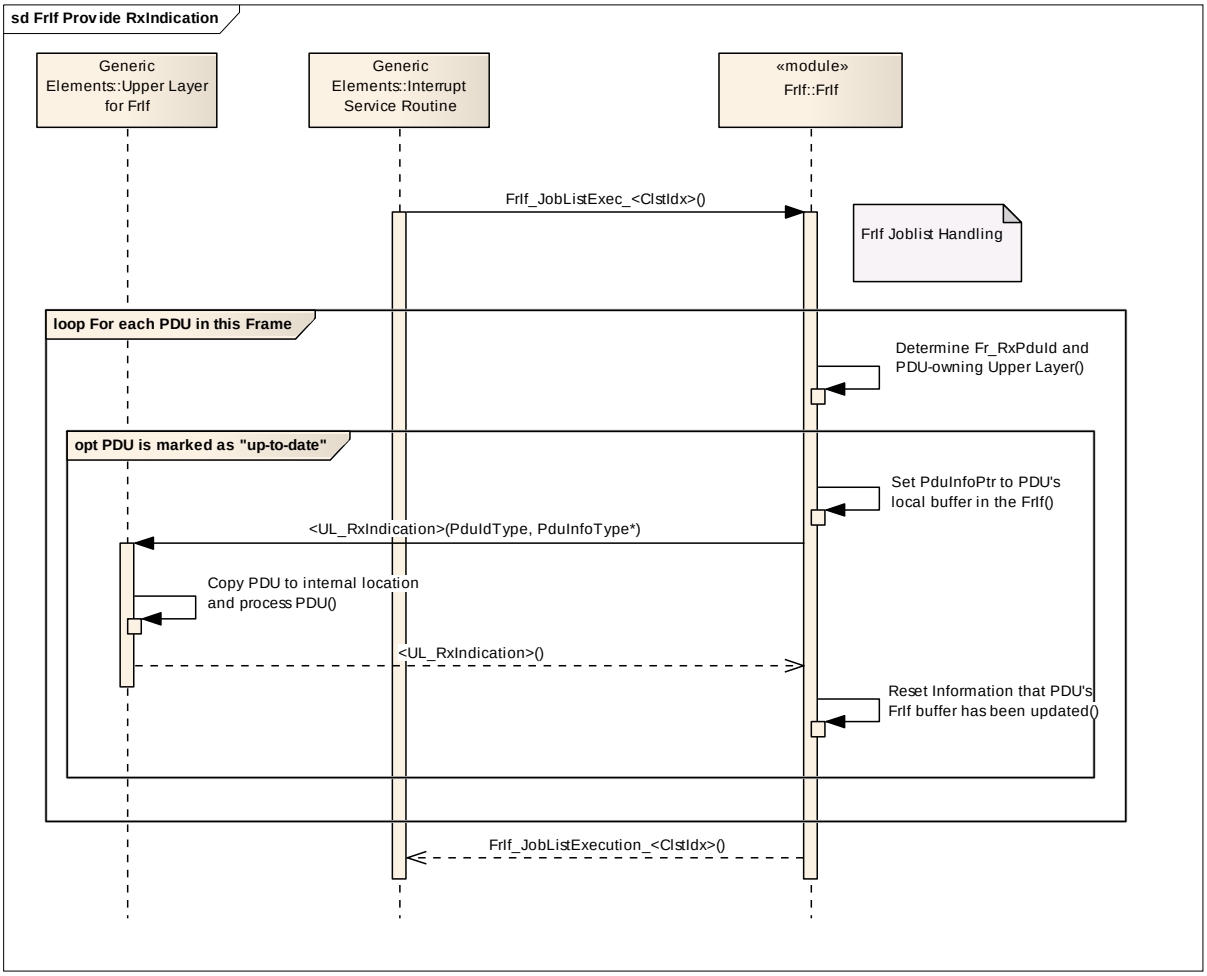


Figure 9-6: ProvideRxIndication

10 Configuration Specification

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

10.1 How to Read this Chapter

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

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

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

10.1.1 Configuration and Configuration Parameters

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

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

FrIf05144:
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

FrIf05077:

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

FrIf05070:

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.

FrIf05071:
The configuration tool must check the consistency of the configuration at configuration time.

FrIf05072:
Configuration rules and constraints for plausibility checks shall be performed during configuration time, wherever possible.

FrIf05073:
These dependencies between FlexRay Interface and FlexRay Driver configuration must be provided at configuration time by the configuration tools.

10.2.1 Variants

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

Use case: Object code delivery, selectable configuration

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

Use case: Execution time optimizations

10.2.2 FrIf

Module Name	*FrIf*
Module Description	Configuration of the FrIf (FlexRay Interface) module.

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

10.2.3 FrIfGeneral

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

SWS Item	**FrIf06080 :**		
Name	FrIfDevErrorDetect {FRIF_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification on or off true: Development Error Detection and Notification on false: Development Error Detection and Notification off		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

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

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

SWS Item	**FrIf06082 :**		
Name	FrIfNumCtrlSupported {FRIF_CTRL_IDX_MAX_SUPPORTED}		
Description	Maximum number of FlexRay CCs that the FlexRay Interface supports		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrIf06083 :**
Name	FrIfVersionInfoApi {FRIF_VERSION_INFO_API}

Description	Enables/disables the existence of the FrIf_GetVersionInfo() API service true: FrIf_GetVersionInfo() API service exists false: FrIf_GetVersionInfo() API service does not exist		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 FrIfCluster

SWS Item	**FrIf05366 :**
Container Name	FrIfCluster{FRIF_CLUSTER}
Description	This container specifies a FrIf Cluster and all related data which is required to enable communication of the Cluster. A Cluster may consist of more than one Controller.
Configuration Parameters	

SWS Item	**FrIf06002 :**		
Name	FrIfClstIdx {FRIF_CLST_IDX}		
Description	This parameter provides a zero-based consecutive index of the FlexRay Clusters. Upper layer BSW modules and the FrIf itself use this index to identify a FlexRay Cluster.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06003 :**		
Name	FrIfMainFunctionPeriod {FRIF_MAINFUNCTION_PERIOD}		
Description	The execution cycle of the FrIf_MainFunction_<cluster>() in seconds. The FrIf does not require this information but the BSW scheduler, which invokes the cluster main functions, needs it in order to plan its tasks.		
Multiplicity	1		
Type	FloatParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06004 :**
Name	FrIfMaxIsrDelay {FRIF_MAX_ISR_DELAY}
Description	The maximum delay in macroticks the FrIf_JoblistExec_<cluster>() function is processed after the absolute timer interrupt was triggered.
Multiplicity	1
Type	IntegerParamDef
Default value	--

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

SWS Item	**FrIf06005 :**		
Name	GAssumedPreciscion {G_ASSUMED_PRECISION}		
Description	Assumed precision of the application network		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06006 :**		
Name	GChannels {G_CHANNELS}		
Description	The channels that are used by the cluster. Implementation Type: Fr_ChannelType		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FR_CHANNEL_A	Cluster uses channel A	
	FR_CHANNEL_AB	Cluster uses channel A and B Implementation Type: Fr_ChannelType	
	FR_CHANNEL_B	Cluster uses channel B	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06007 :**		
Name	GClusterDriftDamping {G_CLUSTER_DRIFT_DAMPING}		
Description	The cluster drift damping factor, based on the longest microtick gdMaxMicrotick used in the cluster. Used to compute the local cluster drift damping factor pClusterDriftDamping [Micrroticks].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 5		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06008 :**		
Name	GColdStartAttempts {G_COLD_START_ATTEMPTS}		
Description	Maximum number of times a node in the cluster is permitted to attempt to start the cluster by initiating schedule synchronization		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 31		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

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

SWS Item	**FrIf06009 :**		
Name	GListenNoise {G_LISTEN_NOISE}		
Description	Upper limit for the start up listen timeout and wake up listen timeout in the presence of noise. It is used as a multiplier of the cluster parameter pdListenTimeout.		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 16		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06010 :**		
Name	GMacroPerCycle {G_MACRO_PER_CYCLE}		
Description	Number of macroticks in a communication cycle.		
Multiplicity	1		
Type	IntegerParamDef		
Range	10 .. 16000		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	**FrIf06011 :**		
Name	GMaxWithoutClockCorrectFatal {G_MAX_WITHOUT_CLOCK_CORRECTION_FATAL}		
Description	Threshold used for testing the vClockCorrectionFailed counter. Defines the number of consecutive even/odd Cycle pairs with missing clock correction terms that will cause the protocol to transition from the POC:normal active or POC:normal passive state into the POC:halt state. [Even/odd cycle pairs].		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 15		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06012 :**		
Name	GMaxWithoutClockCorrectPassive {G_MAX_WITHOUT_CLOCK_CORRECTION_PASSIVE}		
Description	Threshold used for testing the vClockCorrectionFailed counter. Defines the number of consecutive even/odd Cycle pairs with missing clock correction terms that will cause the protocol to transition from the POC:normal active state to the POC:normal passive state. [Even/Odd cycle pairs]		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 15		

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

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

SWS Item	**FrIf06014 :**		
Name	GNumberOfMinislots {G_NUMBER_OF_MINISLOTS}		
Description	Number of minislots in the dynamic segment		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 7986		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

SWS Item	**FrIf06016 :**		
Name	GOffsetCorrectionMax {G_OFFSET_CORRECTION_MAX}		
Description	describes the maximum value which the offset correction should assume in seconds.		
Multiplicity	1		
Type	FloatParamDef		
Range	5.0E-7 .. 3.811E-4		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06017 :		
Name	GOffsetCorrectionStart {G_OFFSET_CORRECTION_START}		
Description	Start of the offset correction phase within the NIT, expressed as the number of macroticks from the start of cycle.		
Multiplicity	1		
Type	IntegerParamDef		
Range	9 .. 15999		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

SWS Item	FrIf06019 :		
Name	GSyncNodeMax {G_SYNC_NODE_MAX}		
Description	Maximum number of nodes that may send frames with the sync frame indicator bit set to one.		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 15		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	FrIf06020 :		
Name	GdActionPointOffset {GD_ACTION_POINT_OFFSET}		
Description	Number of Macroticks the action point is offset from the beginning of a Static Slots or symbol window.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 63		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	FrIf06021 :
Name	GdBit {GD_BIT}
Description	Nominal bit time in seconds
Multiplicity	1
Type	FloatParamDef

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

SWS Item	**FrIf06022 :**		
Name	GdBitMax {GD_BIT_MAX}		
Description	Maximum bit time taking into account the allowable clock deviation of each node (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06023 :**		
Name	GdBitMin {GD_BIT_MIN}		
Description	Minimum bit time taking into account the allowable clock deviation of each node (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06024 :**		
Name	GdCasRxLowMax {GD_CAS_RX_LOW_MAX}		
Description	Upper limit of the CAS acceptance window [gdBit]		
Multiplicity	1		
Type	IntegerParamDef		
Range	67 .. 99		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module/Fr		

SWS Item	**FrIf06025 :**		
Name	GdCycle {GD_CYCLE}		
Description	Length of the cycle, expressed in s		
Multiplicity	1		
Type	FloatParamDef		
Range	1.0E-5 .. 0.016		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06026 :**
Name	GdDynamicSlotIdlePhase {GD_DYNAMIC_SLOT_IDLE_PHASE}

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

SWS Item	**FrIf06027 :**		
Name	GdMacrotick {GD_MACROTICK}		
Description	Duration of the cluster wide nominal macrotick, expressed in s		
Multiplicity	1		
Type	FloatParamDef		
Range	1.0E-6 .. 6.0E-6		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06028 :**		
Name	GdMaxInitializationError {GD_MAX_INITIALIZATION_ERROR}		
Description	Maximum error that a node may have following integration in seconds.		
Multiplicity	1		
Type	FloatParamDef		
Range	0.0 .. 65535.0		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06029 :**		
Name	GdMaxMicrotick {GD_MAX_MICROTICK}		
Description	Maximum Microtick length of all Microticks configured within a Cluster.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	T100NS		
	T12_5NS		
	T200NS		
	T25NS		
	T50NS		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06030 :**
Name	GdMaxPropagationDelay {GD_MAX_PROPAGATION_DELAY}
Description	Maximum propagation delay of a Cluster (in seconds).
Multiplicity	1
Type	FloatParamDef

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

SWS Item	**FrIf06031 :**		
Name	GdMinPropagationDelay {GD_MIN_PROPAGATION_DELAY}		
Description	Minimum propagation delay of a Cluster (in seconds).		
Multiplicity	1		
Type	FloatParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

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

SWS Item	**FrIf06034 :**		
Name	GdNit {GD_NIT}		
Description	Duration of the Network Idle Time [Macroticks]		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 767		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06035 :**

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

SWS Item	**FrIf06036 :**		
Name	GdStaticSlot {GD_STATIC_SLOT}		
Description	Duration of a Static Slot [Macroticks].		
Multiplicity	1		
Type	IntegerParamDef		
Range	4 .. 659		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06037 :**		
Name	GdSymbolWindow {GD_SYMBOL_WINDOW}		
Description	Duration of the symbol window [Macroticks].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 139		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06038 :**		
Name	GdTssTransmitter {GD_TSS_TRANSMITTER}		
Description	Number of bits in the Transmission Start Sequence [gdBits].		
Multiplicity	1		
Type	IntegerParamDef		
Range	3 .. 15		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06039 :**
Name	GdWakeupSymbolRxIdle {GD_WAKEUP_SYMBOL_RX_IDLE}
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.) [gdBit].
Multiplicity	1

Type	IntegerParamDef		
Range	14 .. 59		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06040 :**		
Name	GdWakeupSymbolRxLow {GD_WAKEUP_SYMBOL_RX_LOW}		
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.) [gdBits].		
Multiplicity	1		
Type	IntegerParamDef		
Range	10 .. 55		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06041 :**		
Name	GdWakeupSymbolRxWindow {GD_WAKEUP_SYMBOL_RX_WINDOW}		
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.) [gdBit].		
Multiplicity	1		
Type	IntegerParamDef		
Range	76 .. 301		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06042 :**		
Name	GdWakeupSymbolTxIdle {GD_WAKEUP_SYMBOL_TX_IDLE}		
Description	Number of bits used by the node to transmit the 'idle' part of a wakeup symbol. The duration is equal to cdWakeupSymbolTxIdle [gdBit].		
Multiplicity	1		
Type	IntegerParamDef		
Range	45 .. 180		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06043 :**
Name	GdWakeupSymbolTxLow {GD_WAKEUP_SYMBOL_TX_LOW}

Description	Number of bits used by the node to transmit the LOW part of a wakeup symbol. The duration is equal to cdWakeupSymbolTxLow [gdBit].		
Multiplicity	1		
Type	IntegerParamDef		
Range	15 .. 60		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

10.2.5 FrIfController

SWS Item	**FrIf05363 :**
Container Name	FrIfController
Description	This container contains the configuration of FlexRay CC.
Configuration Parameters	

SWS Item	**FrIf06045 :**		
Name	FrIfCtrlIdx {FRIF_CTRL_IDX}		
Description	This parameter provides a zero-based consecutive index of the FlexRay Communication Controllers. Upper layer BSW modules and the FrIf itself use this index to identify a FlexRay CC.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**FrIf06044 :**		
Name	FrIfFrCtrlRef {FRIF_FR_CTRL}		
Description	Reference to a Controller, which is handled by a specific Driver. This reference is unique for the ECU.		
Multiplicity	1		
Type	Reference to FrController		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfAbsTimer	1..*	This container contains the configuration of an absolute timer of a FlexRay CC.
FrIfFrameTriggering	1..*	A Frame triggering contains the communication parameters of the FlexRay Frame as well as a reference to the Frame Construction Plan.

FrIfLPdu	1..*	LPdu is an abstraction of all FlexRay Frames (L-PDUs) belonging to the same Frame Triggering FrIfFrameTriggering.
FrIfRelTimer	0..*	This container contains the configuration of a relative timer of a FlexRay CC.
FrIfTransceiver	0..2	Up to two FlexRay Transceivers may connect a Controller to a Cluster. This container realizes a Controller-Transceiver assignment.

10.2.6 FrIfAbsTimer

SWS Item	**frIf05361 :**
Container Name	FrIfAbsTimer{FRIF_ABS_TIMER}
Description	This container contains the configuration of an absolute timer of a FlexRay CC.
Configuration Parameters	

SWS Item	**FrIf06047 :**		
Name	FrIfAbsTimerIdx {FRIF_ABS_TIMER_IDX}		
Description	This parameter provides a zero-based consecutive index of the absolute timers. Upper layer BSW modules use this index to identify an absolute timer.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**FrIf06046 :**		
Name	FrIfFrAbsTimerRef {FRIF_FR_ABS_TIMER}		
Description	Reference to an absolute timer, which is handled by a specific FlexRay Driver. This reference is unique for the ECU, therefore, an explicit reference to the FlexRay Driver is not necessary.		
Multiplicity	1		
Type	Reference to FrAbsoluteTimer		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.7 FrIfRelTimer

SWS Item	**FrIf05362 :**
Container Name	FrIfRelTimer{FRIF_REL_TIMER_IDX}
Description	This container contains the configuration of a relative timer of a FlexRay CC.
Configuration Parameters	

SWS Item	**FrIf06060 :**
Name	FrIfRelTimerIdx {FRIF_REL_TIMER_IDX}
Description	This parameter provides a zero-based consecutive index of the relative

	timers. Upper layer BSW modules use this index to identify a relative timer.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**FrIf06059 :**		
Name	FrIfFrRelTimerRef {FRIF_FR_REL_TIMER}		
Description	Reference to a relative timer, which is handled by a specific FlexRay Driver. This reference is unique for the ECU, therefore, an explicit reference to the FlexRay Driver is not necessary.		
Multiplicity	1		
Type	Reference to FrRelativeTimer		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.8 FrIfFrameTriggering

SWS Item	**FRIF_FRAME_TRIGGERING :**
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	

SWS Item	**FrIf06049 :**		
Name	FrIfAllowDynamicLSduLength {FRIF_ALLOW_DYNAMIC_LSDU_LENGTH}		
Description	Allows L-PDU length reduction ('FrIfLSduLength' defines max. length) and indicates that the related CC buffer has to be reconfigured for the actual length and Header-CRC before transmission of the L-PDU.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06050 :**		
Name	FrIfAlwaysTransmit {FRIF_ALWAYS_TRANSMIT}		
Description	Defines wether the driver's API function Fr_TransmitTxLPdu() shall always be called for this L-PDU.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: Module

SWS Item	**FrIf06051 :**		
Name	FrIfBaseCycle {FRIF_BASE_CYCLE}		
Description	This parameter contains the FlexRay Base Cycle used to transmit this FlexRay Frame.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 63		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

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

SWS Item	**FrIf06054 :**		
Name	FrIfLSduLength {FRIF_LSDU_LENGTH}		
Description	The payload length of the Frame is given here. This parameter is required for validation if configured PDUs and update information fits into the Frame at configuration time [bytes].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 254		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

	dependency: The parameter depends on the low level parameters of the FlexRay CC.

SWS Item	**FrIf06050 :**		
Name	FrIfNoneMode {FRIF_NONE_MODE}		
Description	Using the "None-Mode" which means that there is no API FrIf_Transmit call of the upper layer for this PDU.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06055 :**		
Name	FrIfPayloadPreamble {FRIF_PAYLOADPREAMBLE_ID}		
Description	Switching the Payload Preamble bit.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

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

No Included Containers

10.2.9 FrIfJobList

SWS Item	**FrIf05367 :**
Container Name	FrIfJobList{FRIF_JOBLIST}

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

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

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

10.2.10 FrIfJob

SWS Item	**FrIf05368 :**
Container Name	FrIfJob{FRIF_JOB}
Description	A job may contain more than one operation that are executed at a specific point in time.
Configuration Parameters	

SWS Item	**FrIf06064 :**		
Name	FrIfCycle {FRIF_CYCLE}		
Description	The FlexRay Cycle in which the communication operation will execute this job		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 63		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06065 :**		
Name	FrIfMacrotick {FRIF_MACROTICK}		
Description	Macrotick offset in the Cycle [Macrotick]		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfCommunicationOperation	1..*	A separate operation which is part of a FlexRay Job and

		defines what type of action is executed.

10.2.11 FrIfCommunicationOperation

SWS Item	**FrIf05369 :**
Container Name	FrIfCommunicationOperation{FRIF_COMMUNICATION_OPERATION}
Description	A separate operation which is part of a FlexRay Job and defines what type of action is executed.
Configuration Parameters	

SWS Item	**FrIf06067 :**		
Name	FrIfCommunicationAction {FRIF_COMMUNICATION_ACTION}		
Description	The action to be performed in the FlexRay Operation		
Multiplicity	1		
Type	EnumerationParamDef		
Range	DECOUPLED_TRANSMISSION	Decoupled transmission	
	PREPARE_LPDU	Prepare message buffer of CC	
	RECEIVE_AND_INDICATE	Immediate reception	
	RECEIVE_AND_STORE	Decoupled reception	
	RX_INDICATION	Reception indication	
	TX_CONFIRMATION	Transmission confirmation	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06068 :**		
Name	FrIfCommunicationOperationIdx {FRIF_COMMUNICATION_OPERATION_IDX}		
Description	For each FlexRay Communication Job, this index spans a range of zero-based consecutive values and thus defines the order of the FlexRay Communication Operation in the respective FlexRay Communication Job.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06066 :**		
Name	FrIfLPduIdxRef		
Description	Reference to a L-PDu index		
Multiplicity	1		
Type	Reference to FrIfLPdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.12 FrIfFrameStructure

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

SWS Item	**FrIf06113 :**		
Name	FrIfByteOrder {FRIF_BYTE_ORDER}		
Description	This parameter defines the ByteOrder of all Pdus that are mapped into the Frame. The absolute position of a Pdu in the Frame is determined by the definition of the ByteOrder parameter: If BIG_ENDIAN is specified, the FrIfPduOffset indicates the position of the most significant bit in the Frame. If LITTLE_ENDIAN is specified, the FrIfPduOffset indicates the position of the least significant bit in the Frame.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	BIG_ENDIAN		
	LITTLE_ENDIAN		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

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

10.2.13 FrIfPdusInFrame

SWS Item	**FrIf05371 :**
Container Name	FrIfPdusInFrame{FRIF_PDUS_IN_FRAME}
Description	This container holds all the information about a PDU in a FlexRay Frame.
Configuration Parameters	

SWS Item	**FrIf06070 :**		
Name	FrIfPduOffset {FRIF_PDU_OFFSET}		
Description	The value specifies the offset of the PDU within the Frame [bytes].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 253		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: This parameter depends on the number of PDUs contained in the Frame, PDU length, and Update-Bits of other PDUs in the Frame. In addition, if the Frame will is sent in static segment, this parameter depends on GPayloadLengthStatic.		

SWS Item	**FrIf06071 :**
Name	FrIfPduUpdateBitOffset {FRIF_PDU_UPDATE_BIT_OFFSET}
Description	This value specifies where the PDU's Update-Bit is stored in the Frame (bit

	location of PDU's Update-Bit in the FlexRay Frame).		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	0 .. 2031		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: This parameter depends on the number of PDUs contained in the Frame, PDU length, and Update-Bits of other PDUs in the Frame. In addition, if the Frame will is sent in static segment, this parameter depends on GPayloadLengthStatic.		

SWS Item	**FrIf06069 :**		
Name	FrFrIfPduRef {FRIF_PDU}		
Description	This is the reference to the local definition of a PDU.		
Multiplicity	1		
Type	Reference to FrIfPdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.14 FrIfLPdu

SWS Item	**FrIf05364 :**
Container Name	FrIfLPdu
Description	LPdu is an abstraction of all FlexRay Frames (L-PDUs) belonging to the same Frame Triggering FrIfFrameTriggering.
Configuration Parameters	

SWS Item	**FrIf06058 :**		
Name	FrIfLPduIdx {FRIF_LPDU_IDX}		
Description	This parameter identifies the L-PDU in the interaction between FlexRay Interface and FlexRay Driver.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06057 :**		
Name	FrIfVBTriggeringRef {FRIF_FRAME_TRIGGERING}		
Description	Reference to the assigned Frame triggering.		
Multiplicity	1		
Type	Reference to FrIfFrameTriggering		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.15 FrIfPdu

SWS Item	**FrIf05372 :**
Container Name	FrIfPdu{FRIF_PDU}
Description	Contains PDU information. A PDU may be either a transmission PDU or a reception PDU.
Configuration Parameters	

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

10.2.16 FrIfTxPdu

SWS Item	**FrIf05374 :**
Container Name	FrIfTxPdu{FRIF_TX_PDU}
Description	This container specifies transmission PDUs.
Configuration Parameters	

SWS Item	**FrIf06075 :**		
Name	FrIfConfirm {FRIF_CONFIRM}		
Description	Defines whether the transmission of a PDU should be checked and confirmed to the PDU owning BSW module.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06076 :**		
Name	FrIfCounterLimit {FRIF_COUNTER_LIMIT}		
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.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06077 :**		
Name	FrIfImmediate		
Description	Defines whether the the PDU is transmitted immediate or decoupled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrIf06078 :**		
Name	FrIfTxPduId {FRIF_TX_PDU_ID}		
Description	The global PDU identifier, which has to be used by the upper layer BSW module. The identifier has to be zero based and consecutive.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: FrIf and PduR/FrNm/FrTp		

SWS Item	**FrIf06074 :**		
Name	FrIfPduRef {FRIF_PDU_REF}		
Description	Reference to the external PDU definition.		
Multiplicity	1		
Type	Reference to Pdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.17 FrIfRxPdu

SWS Item	**FrIf05373 :**
Container Name	FrIfRxPdu{FRIF_RX_PDU}
Description	Receive PDU
Configuration Parameters	

SWS Item	**FrIf06073 :**		
Name	FrIfPduRef {FRIF_PDU_REF}		
Description	Reference to the external PDU definition.		
Multiplicity	1		
Type	Reference to Pdu		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.18 FrIfPduDirection

SWS Item	**FrIf06072 :**
Choice Container Name	FrIfPduDirection
Description	A PDU is either transmit or receive

Container Choices		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfRxPdu	0..1	Receive PDU

FrIfTxPdu	0..1	This container specifies transmission PDUs.

10.2.19 FrIfTransceiver

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

SWS Item	**FrIf06062 :**		
Name	FrIfClusterChannel {FRIF_TRCV_CLUSTER_CHANNEL}		
Description	This parameter identifies to which one of the two Channels "A" or "B" of the Cluster the Transceiver is connected.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRIF_CHANNEL_A	Channel A	
	FRIF_CHANNEL_B	Channel B	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

SWS Item	**FrIf06061 :**		
Name	FrIfFrTrcvChannelRef {FRIF_TRCV_CHANNEL}		
Description	Reference to a Transceiver Driver Channel. This reference is unique for the ECU.		
Multiplicity	1		
Type	Reference to FrTrcvNode		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU		

No Included Containers

10.2.20 FrIfConfig

SWS Item	**FrIf06001 :**
Container Name	FrIfConfig [Multi Config Container]
Description	Configuration of the FlexRay Interface. This container is a MultipleConfigurationContainer, i.e. this container and its sub-containers exist once per configuration set.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrIfCluster	1..*	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.
FrIfFrameStructure	0..*	The Frame structure specifies a Construction Plan how a Frame is assembled with PDUs and their respective Update-Bits.
FrIfPdu	1..*	Contains PDU information. A PDU may be either a

		transmission PDU or a reception PDU.

10.3 Published Information

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

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

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

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

11 Changes during SWS Improvements by Technical Office

11.1 Deleted SWS Items

None

11.2 Replaced SWS Items

None

11.3 Changed SWS Items

None

11.4 Added SWS Items

SWS Item	*Rationale*
FrIf05001	UML model linking of imported types
FrIf05002	UML model linking of FrIf_GetVersionInfo
FrIf05003	UML model linking of FrIf_Init
FrIf05004	UML model linking of FrIf_ControllerInit
FrIf05005	UML model linking of FrIf_StartCommunication
FrIf05006	UML model linking of FrIf_HaltCommunication
FrIf05007	UML model linking of FrIf_AbortCommunication
FrIf05008	UML model linking of FrIf_RequestControllerStateTransition
FrIf05009	UML model linking of FrIf_GetControllerState
FrIf05010	UML model linking of FrIf_SetWakeupChannel
FrIf05011	UML model linking of FrIf_SendWUP
FrIf05012	UML model linking of FrIf_GetSyncState
FrIf05013	UML model linking of FrIf_SetExtSync
FrIf05014	UML model linking of FrIf_GetPOCStatus
FrIf05015	UML model linking of FrIf_GetGlobalTime
FrIf05016	UML model linking of FrIf_GetNmVector
FrIf05017	UML model linking of FrIf_AllowColdstart
FrIf05018	UML model linking of FrIf_GetMacroticksPerCycle
FrIf05019	UML model linking of FrIf_ConvertNanosecToMacroticks
FrIf05020	UML model linking of FrIf_ConvertMacroticksToNanosec
FrIf05021	UML model linking of FrIf_SetAbsoluteTimer
FrIf05022	UML model linking of FrIf_SetRelativeTimer
FrIf05023	UML model linking of FrIf_CancelRelativeTimer
FrIf05024	UML model linking of FrIf_CancelRelativeTimer
FrIf05025	UML model linking of FrIf_EnableAbsoluteTimerIRQ
FrIf05026	UML model linking of FrIf_EnableRelativeTimerIRQ
FrIf05027	UML model linking of FrIf_GetAbsoluteTimerIRQStatus
FrIf05028	UML model linking of FrIf_GetRelativeTimerIRQStatus
FrIf05029	UML model linking of FrIf_AckAbsoluteTimerIRQ
FrIf05030	UML model linking of FrIf_AckRelativeTimerIRQ
FrIf05031	UML model linking of FrIf_DisableAbsoluteTimerIRQ
FrIf05032	UML model linking of FrIf_DisableRelativeTimerIRQ
FrIf05033	UML model linking of FrIf_Transmit
FrIf05034	UML model linking of FrIf_SetTransceiverMode
FrIf05035	UML model linking of FrIf_GetTransceiverMode

FrIf05036	UML model linking of FrIf_GetTransceiverWUReason
FrIf05037	UML model linking of FrIf_EnableTransceiverWakeup
FrIf05038	UML model linking of FrIf_DisableTransceiverWakeup
FrIf05039	UML model linking of FrIf_ClearTransceiverWakeup
FrIf05040	UML model linking of FrIf_JobListExec_<ClstIdx>
FrIf05041	UML model linking of FrIf_Cbk_WakeupByTransceiver
FrIf05042	UML model linking of FrIf_MainFunction_<ClstIdx>
FrIf05043	UML model linking of mandatory interfaces
FrIf05044	UML model linking of optional interfaces
FrIf05045	UML model linking of <UL_RxIndication>
FrIf05046	UML model linking of <UL_TxConfirmation>
FrIf05047	UML model linking of <UL_TriggerTransmit>