Document Title	Specification of FlexRay Driver
Document Owner	AUTOSAR GbR
Document Responsibility	AUTOSAR GbR
Document Identification No	026
Document Classification	Standard

Document Version	2.2.0
Document Status	Draft
Part of Release	2.1
Revision	0019

Document Change History

Date	Version	Changed by	Change Description
17.04.2008	2.2.0	AUTOSAR Administration	• Chapter 8.3.14: Added description for dynamic payload length support • Legal disclaimer revised • Exchange links to SVNII
31.01.2007	2.1.0	AUTOSAR Administration	• Renamed members of Fr_POCStatusType according to FlexRay Protocol Specification 2.1 • Renamed API function Fr_TransmitTxLSdu(), Fr_CheckTxLSduSTatus(), Fr_ReceiveRxLSdu() and related API types. • Added new API function Fr_PrepareLPdu() • Added new API function Fr_StopMTS() • Added reference to BSW Scheduler SWS • Reworked API function DET and DEM reporting • Reworked DET error codes • Updated traceability table • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
28.06.2006	2.0.0	AUTOSAR Administration	Second release
04.08.2005	1.0.0	AUTOSAR Administration	Initial release

Release Notes

Errata and known deficiencies

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

Known and potential problems resulting from known deficiencies

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

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

Changes planned for next release

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

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

Disclaimer

This document of a specification as released by the AUTOSAR Development Partnership is intended **for the purpose of information only**. The commercial exploitation of material contained in this specification requires membership of the AUTOSAR Development Partnership or an agreement with the AUTOSAR Development Partnership. The AUTOSAR Development Partnership will not be liable for any use of this specification. Following the completion of the development of the AUTOSAR specifications commercial exploitation licenses will be made available to end users by way of written License Agreement only.

Advice to users of AUTOSAR Specification Documents:

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

The FlexRay Driver (Fr) abstracts the hardware related implementation details of specific FlexRay Communication Controllers (CC). All mandatory features of CCs according to the FlexRay Specification [11] are encapsulated within the Fr and shall be accessed via this uniform interface only. The API provides abstract functional operations that are mapped to a sequence of hardware accesses depending on the actual implemented Fr. Thus, the FlexRay Interface (FrIf) as the user of the Fr is independent of the underlying FlexRay CC hardware. The Fr is context free. Fr API functions are executed in the context of the FrIf.

A single Fr supports only one type of FlexRay CC hardware, but several CC of the same type. The FlexRay Driver's prefix is uniquely assigned per Fr to allow usage of different FlexRay Drivers, which are separated by namespace. The FrIf can access different FlexRay CCs types using different FlexRay Drivers. The decision which driver to use to access a particular CC is a configuration parameter of the FrIf.

The configuration of the Fr shall be done at system configuration time, with the Fr specific configuration data being generated by a Dedicated Module Generator (DMG), which translates the parameters out of the ECU configuration description to Fr specific configuration data structures.
FR006: If a CC provides further features marked as optional in the FlexRay Specification (e. g. receive FIFO), those features should be supported if transparently applicable to the specified Fr API. The usage of optional features is encapsulated within the Fr. The configuration tool only might have knowledge about the abilities of the CC in order to generate an appropriate and valid configuration.

Figure 1 depicts the basic structure of the FlexRay stack. One FrIf accesses several CCs using one or several FlexRay Drivers.

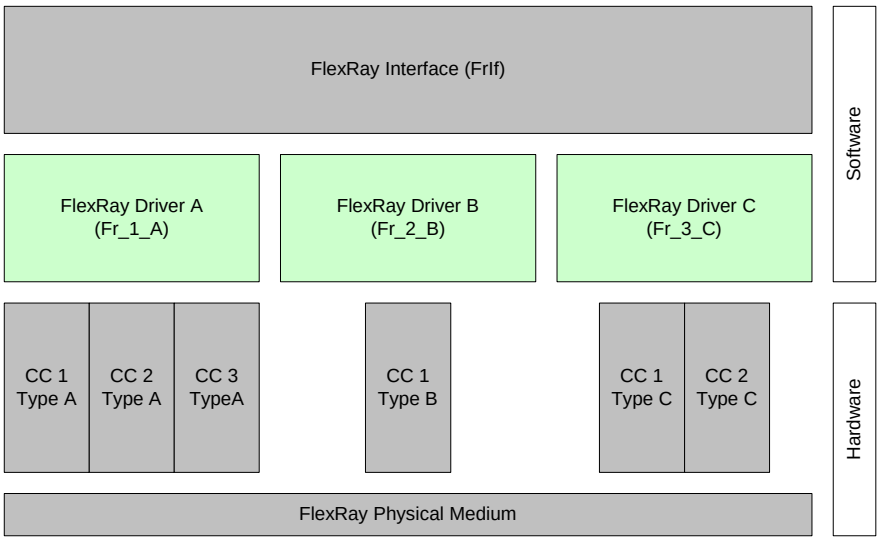

Figure 1: FlexRay stack module overview

2 Acronyms and abbreviations

Abbreviation:	*Description:*
API	Application Programming Interface
DEM	Diagnostic Event Manager
DET	Development Error Tracer
DMG	Dedicated Module Generator
CC	Communication Controller
CHI	Controller Host Interface
Fr	FlexRay Driver
FrIf	FlexRay Interface
POC	Protocol Operation Control (see [11] for details)
POCState	Actual CC internal state of the POC. This state might differ from vPOC!State in certain cases, e.g. after FREEZE command invocation (see [11] for details).
vPOC	Data structure provided from the CC to the host at the CHI, which contains the actual POC status of the CC (see [11] for details).

2.1 Glossary of terms

Term:	*Definition:*
absolute timer	An absolute timer is set to and triggered by an absolute global time of a FlexRay cluster. The FlexRay global time consists of a cycle and a macrotick offset
cluster	A communication system of multiple nodes connected to each other.
macrotick	The macrotick represents the smallest unit of the global synchronized time of a FlexRay cluster.
relative timer	A relative timer is set to and triggered by an offset in macroticks from the current global time of a FlexRay cluster
synchronized	A FlexRay CC is considered synchronized, to the FlexRay cluster connected to, as long as the following condition holds true: `((!vPOC!Freeze) && (vPOC!State == NORMAL_ACTIVE) || (vPOC!State == NORMAL_PASSIVE))`

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SRS_General.pdf

[4] Specification of ECU Configuration,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_ECU_Configuration.pdf

[5] Specification of Communication Stack Types,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_ComStackTypes.pdf

[6] Specification of Platform Types,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_PlatformTypes.pdf

[7] Specification of FlexRay Interface,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_FlexRayInterface.pdf

[8] Specification of FlexRay Transceiver Driver,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_FlexRayTransceiver.pdf

[9] Specification of BSW Scheduler,
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_Scheduler.pdf

[10] Specification of Memory Mapping
https://svn2.autosar.org/repos2/22_Releases
AUTOSAR_SWS_MemoryMapping.pdf

3.2 Related standards and norms

[11] 2005, FlexRay Consortium, FlexRay Communication Systems Protocol Specification, Version 2.1 Revision A.

[12] 2006, Gemeinsames Subset der MISRA C Guidelines, Version 2.0, http://www.automotive-his.de/download/HIS_SubSet_MISRA_C_2.0.pdf

4 Constraints and assumptions

4.1 Limitations

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

4.2 Applicability to car domains

The FlexRay Communication stack can be used wherever high data rates and fault tolerant communication (in conjunction with [11]) is required. Furthermore it enables the synchronized operation of several ECUs within a car.

5 Dependencies to other modules

FR066: his chapter lists the modules interacting with the Fr.

Modules that use Fr:

- The FrIf is the only user of the Fr. It uses the Fr(s) to access possibly different FlexRay Communication Controllers in a uniform and abstracted way.

Modules used by the Fr:

- The Fr shall use the Development Error Tracer (DET) for reporting of development errors.
- The Fr shall use the Diagnostic Event Manager (DEM) for reporting of diagnostic-relevant events and states.
- The Fr shall use the BSW Scheduler mechanisms for data consistency if required.
- The FlexRay Driver might use modules for accessing the CCs. E. g. if a CC is connected via SPI the corresponding Fr has to use the SPI module.

Other Module dependencies:

- **FR061:** On certain systems the CC might share resources with other components (e.g. the MCU), and might depend on their configuration. If those resources are within scope of the other modules (e.g. PLL configuration, memory mapping, etc.) the Fr doesn't take care of configuring those components but requires that their initialization precede the Fr initialization.

5.1 File structure

This section gives an overview about the files and their relations required for a proper Fr implementation. Please note that the file structure is not specified completely but the implementation shall use at least the files and the file structure presented in this chapter.

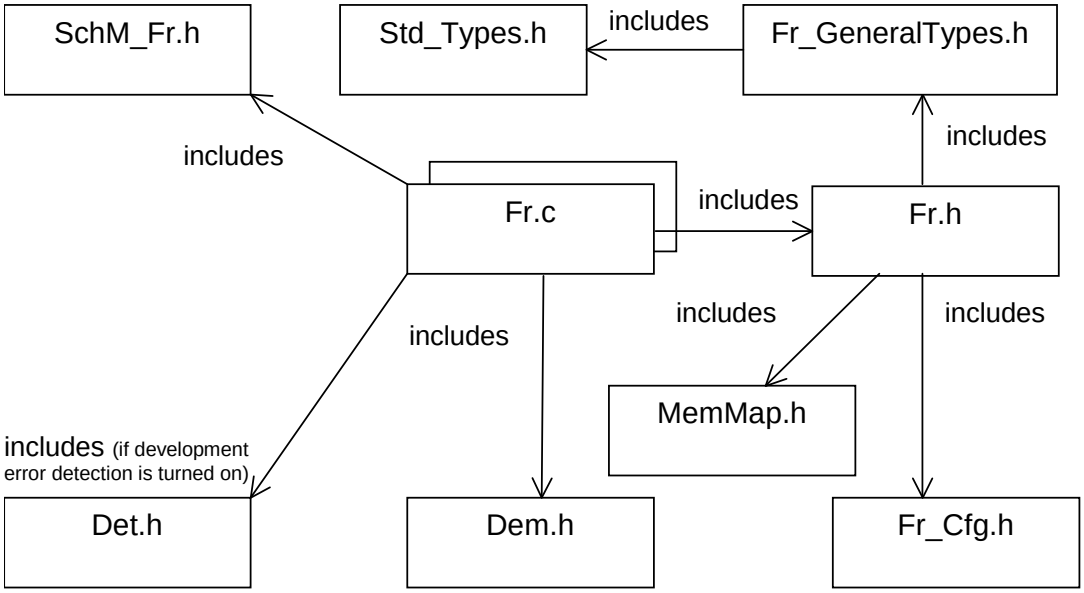

Figure 2 FlexRay Driver header-file structure

FR100: Figure 2 shows the relation of source code files and header files that shall be used for implementation.

FR074: All files related to the Fr module shall follow the naming convention *Fr[_<description>].<extension>* .

5.1.1 Code file structure

FR045: The implementation and configuration of Fr shall contain the following source code files:

- *Fr.c* for the general source code, or *Fr_<purpose>.c* if the implementation is separated into several source code files.
- *Fr_PBcfg.c* for post-build-time configuration parameters.
- *Fr_Lcfg.c* for link-time configuration parameters.

5.1.2 Header file structure

FR079: The implementation and configuration of Fr shall contain the following header files:

- **FR101:** *Fr.h* is the main module interface file that shall contain all types and function prototypes required by Fr users.
- ***FR063:*** *Fr_Cfg.h* shall contain the pre-compile-time configuration parameters.
- *Fr_GeneralTypes.h* shall contain all types and constants that are shared among the AUTOSAR FlexRay modules Fr, FrIf and FrTrcv.
- **FR071:** The 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*.

- The module shall include *Det.h* file. By this inclusion the API's to report development errors are included.
- **FR111:** The module source code files shall include *SchM_Fr.h* if data consistency mechanisms of the BSW scheduler are required as described in [9].
- **FR112:** The module header file shall include *MemMap.h* and apply the memory mapping abstraction mechanisms as specified by [10].

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software, general

Requirement		Satisfied by
BSW00344	Reference to link-time configuration	FR085, FR027
BSW00404	Reference to post build time configuration	FR032, FR027
BSW00405	Reference to multiple configuration sets	FR032
BSW00345	Pre-compile-time configuration	FR027
BSW159	Tool-based configuration	FR055
BSW167	Static configuration checking	FR062
BSW171	Configurability of optional functionality	FR019
BSW170	Data for reconfiguration of AUTOSAR SW-Components	FR061
BSW00380	Separate C-Files for configuration parameters	FR045
BSW00419	Separate C-Files for pre-compile time configuration parameters	FR063
BSW00381	Separate configuration header file for pre-compile time parameters	FR063
BSW00412	Separate H-File for configuration parameters	FR063
BSW00383	List dependencies of configuration files	FR068
BSW00384	List dependencies to other modules	FR066
BSW00387	Specify the configuration class of callback function	not applicable to Fr SWS
BSW00388	Introduce containers	FR067
BSW00389	Containers shall have names	FR082, FR086, FR087, FR088, FR089
BSW00390	Parameter content shall be unique within the module	FR082, FR086, FR087, FR088, FR089
BSW00391	Parameter shall have unique names	FR082, FR086, FR087, FR088, FR089
BSW00392	Parameters shall have a type	FR082, FR086, FR087, FR088, FR089
BSW00393	Parameters shall have a range	FR082, FR086, FR087, FR088, FR089
BSW00394	Specify the scope of the parameters	FR082, FR086, FR087, FR088, FR089
BSW00395	List the required parameters (per parameter)	FR082, FR086, FR087, FR088, FR089
BSW00396	Configuration classes	FR082, FR086, FR087, FR088, FR089
BSW00397	Pre-compile-time parameters	FR082, FR086, FR087, FR088, FR089, FR063, FR019, FR027
BSW00398	Link-time parameters	FR082, FR086, FR087, FR088, FR089, FR045, FR027
BSW00399	Loadable Post-build time parameters	FR082, FR086, FR087, FR088, FR089, FR027, FR045
BSW00400	Selectable Post-build time parameters	FR082, FR086, FR087, FR088, FR089, FR027, FR045, FR032, FR064
BSW00402	Published information	FR069
BSW00375	Notification of wake-up reason	not applicable to Fr SWS
BSW101	Initialization interface	FR032
BSW00416	Sequence of Initialization	not applicable to Fr SWS
BSW00406	Check module initialization	FR032, FR017, FR023, FR090, FR024, FR010, FR014, FR011, FR009,

		FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039
BSW168	Diagnostic Interface of SW components	not applicable to Fr SWS
BSW00407	Function to read out published parameters	FR070, FR019
BSW00423	Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	not applicable to Fr SWS
BSW00424	BSW main processing function task allocation	not applicable to Fr SWS
BSW00425	Trigger conditions for schedulable objects	not applicable to Fr SWS
BSW00426	Exclusive areas in BSW modules	not applicable to Fr SWS
BSW00427	ISR description for BSW modules	not applicable to Fr SWS
BSW00428	Execution order dependencies of main processing functions	not applicable to Fr SWS
BSW00429	Restricted BSW OS functionality access	not applicable to Fr SWS
BSW00431	The BSW Scheduler module implements task bodies	not applicable to Fr SWS
BSW00432	Modules should have separate main processing functions for read/receive and write/transmit data path	not applicable to Fr SWS
BSW00433	Calling of main processing functions	not applicable to Fr SWS
BSW00434	The Schedule Module shall provide an API for exclusive areas	not applicable to Fr SWS
BSW00336	Shutdown interface	FR014
BSW00337	Classification of errors	FR025
BSW00338	Detection and Reporting of development errors	FR032, FR017, FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039, FR070
BSW00369	Do not return development error codes via API	FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039, FR057
BSW00339	Reporting of production relevant error status	FR032, FR017, FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039, FR028
BSW00417	Reporting of Error Events by Non-Basic Software	not applicable to Fr SWS
BSW00323	API parameter checking	FR032, FR017, FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021,

		FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039, FR070, FR029
BSW004	Version check	FR030, FR045, FR079
BSW00409	Header files for production code error IDs	FR071
BSW00385	List possible error notifications	FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039, FR057, FR025
BSW00386	Configuration for detecting an error	not applicable to Fr SWS
BSW161	Microcontroller abstraction	not applicable to Fr SWS
BSW162	ECU layout abstraction	not applicable to Fr SWS
BSW005	No hard coded horizontal interfaces within MCAL	not applicable to Fr SWS
BSW00415	User dependent include files	not applicable to Fr SWS
BSW164	Implementation of interrupt service routines	not applicable to Fr SWS
BSW00325	Runtime of interrupt service routines	not applicable to Fr SWS
BSW00326	Transition from ISRs to OS tasks	not applicable to Fr SWS
BSW00342	Usage of source code and object code	FR097
BSW00343	Specification and configuration of time	FR082, FR086, FR087, FR088, FR089, FR063, FR019, FR027
BSW160	Human-readable configuration data	FR072
BSW007	HIS MISRA C	FR073
BSW00300	Module naming convention	FR074
BSW00413	Accessing instances of BSW modules	FR075
BSW00347	Naming separation of different instances of BSW drivers	FR076
BSW00305	Self-defined data types naming convention	FR077, FR110
BSW00307	Global variables naming convention	FR098
BSW00310	API naming convention	FR032, FR017, FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096, FR034, FR038, FR036, FR040, FR035, FR039, FR070, FR029, FR098
BSW00373	Main processing function naming convention	not applicable to Fr SWS
BSW00327	Error values naming convention	FR025, FR078
BSW00335	Status values naming convention	not applicable to Fr SWS
BSW00350	Development error detection keyword	FR026, FR029, FR019
BSW00408	Configuration parameter naming convention	FR082, FR086, FR087, FR088, FR089
BSW00410	Compiler switches shall have defined values	FR019
BSW00411	Get version info keyword	FR019
BSW00346	Basic set of module files	FR045, FR079
BSW158	Separation of configuration from implementation	FR045, FR079
BSW00314	Separation of interrupt frames and service routines	not applicable to Fr SWS
BSW00370	Separation of callback interface from API	not applicable to Fr SWS
BSW00348	Standard type header	FR099, FR100

BSW00353	Platform specific type header	FR099, FR100
BSW00361	Compiler specific language extension header	FR099, FR100
BSW00301	Limit imported information	FR100
BSW00302	Limit exported information	FR101
BSW00328	Avoid duplication of code	not applicable to Fr SWS
BSW00312	Shared code shall be reentrant	not applicable to Fr SWS
BSW006	Platform independency	not applicable to Fr SWS
BSW00357	Standard API return type	FR023, FR090, FR024, FR010, FR014, FR011, FR009, FR091, FR041, FR021, FR012, FR092, FR093, FR094, FR042, FR033, FR037, FR095, FR096
BSW00377	Module specific API return types	not applicable to Fr SWS
BSW00304	AUTOSAR integer data types	FR099
BSW00355	Do not redefine AUTOSAR integer data types	FR099
BSW00378	AUTOSAR boolean type	FR099
BSW00306	Avoid direct use of compiler and platform specific keywords	not applicable to Fr SWS
BSW00308	Definition of global data	FR102
BSW00309	Global data with read-only constraint	FR085
BSW00371	Do not pass function pointers via API	not applicable to Fr SWS
BSW00358	Return type of init() functions	FR032
BSW00414	Parameter of init function	FR032
BSW00376	Return type and parameters of main processing functions	not applicable to Fr SWS
BSW00359	Return type of callback functions	not applicable to Fr SWS
BSW00360	Parameters of callback functions	not applicable to Fr SWS
BSW00329	Avoidance of generic interfaces	not applicable to Fr SWS
BSW00330	Usage of macros / inline functions instead of functions	not applicable to Fr SWS
BSW00331	Separation of error and status values	not applicable to Fr SWS
BSW009	Module User Documentation	not applicable to Fr SWS
BSW00401	Documentation of multiple instances of configuration parameters	FR082, FR086, FR087, FR088, FR089
BSW172	Compatibility and documentation of scheduling strategy	not applicable to Fr SWS
BSW010	Memory resource documentation	not applicable to Fr SWS
BSW00333	Documentation of callback function context	not applicable to Fr SWS
BSW00374	Module vendor identification	FR069, FR080
BSW00379	Module identification	FR069, FR080
BSW003	Version identification	FR069, FR080
BSW00318	Format of module version numbers	FR069
BSW00321	Enumeration of module version numbers	FR069
BSW00341	Microcontroller compatibility documentation	not applicable to Fr SWS
BSW00334	Provision of XML file	FR080
BSW00435	Header File Structure for the Basic Software Scheduler	FR111
BSW00436	Module Header File Structure for the Memory Mapping	FR112

Document: AUTOSAR requirements on Basic Software, Module FlexRay Driver

Requirement		***Satisfied by***
BSW05000	Support of Synchronous SW Modules	not applicable to Fr SWS
BSW05001	Support of Asynchronous SW Modules	not applicable to Fr SWS
BSW05002	FlexRay Interface and FlexRay Driver as Only Necessarily Synchronous SW Modules	FR104, FR105
BSW05003	Support of Slot/Cycle Multiplexing	FR005, FR092, FR093, FR094
BSW05169	Avoid Timer Interrupts during Start-up	FR017

BSW05055	Avoid Timer Interrupts during Shutdown	FR106
BSW05064	Abstraction of FlexRay CC-specific Implementation	FR001
BSW05065	Number of FlexRay CCs per Driver	FR004
BSW05005	Support of Hardware FIFO Mechanism	FR006
BSW05024	Abstraction from CC Buffer Configuration	FR005, FR092, FR093, FR094
BSW05066	L-SDU-Based API	FR005, FR092, FR093, FR094, FR104
BSW05058	Configuration of FlexRay Driver at System Configuration Time	FR002
BSW05059	Transmit/Receive Buffer Configuration	FR017, FR103
BSW05116	Initialization of FlexRay CC	FR017
BSW05011	Initialize Low-Level Parameters	FR017
BSW05012	Initialize FlexRay CC Transmit/Receive Buffers	FR017
BSW05109	Start-up of a FlexRay CC	FR010
BSW05117	Sending of Wake-Up Pattern	FR009, FR091
BSW05120	Get FlexRay CC POC Status	FR012
BSW05121	Get FlexRay CC Sync State	FR021
BSW05106	Buffer Reconfiguration in Normal Active Mode	FR103, FR107
BSW05107	MTS Sending	FR023, FR090
BSW05111	Get MTS Reception Status	FR024
BSW05125	Interrupt Handling	FR034, FR038, FR036, FR040, FR035, FR039, FR108, FR109
BSW05114	Abortion of FlexRay CC Communication	FR011
BSW05115	Halt of FlexRay CC Communication	FR014
BSW05033	Tick Conversion	not applicable to Fr SWS
BSW05053	Cluster External Clock Synchronization	not applicable to Fr SWS
BSW05156	Controller External Clock Synchronization	FR041
BSW05044	Set Absolute Timer	FR033, FR095
BSW05045	Set Relative Timer	FR034, FR096
BSW05046	Enable Absolute Alarms	FR034
BSW05047	Disable Absolute Alarms	FR035
BSW05048	Acknowledge Absolute Alarms	FR036
BSW05049	Enable Relative Alarms	FR038
BSW05050	Disable Relative Alarms	FR039
BSW05051	Acknowledge Relative Alarms	FR040
BSW05052	Get Cycle Length in Macroticks	not applicable to Fr SWS
BSW05019	Get FlexRay Global Time	FR042
BSW05072	FlexRay Time Services Access if CC is Out of Sync	FR044

7 Functional specification

7.1 General description

FR001: A single Fr module offers a uniform way to use features of FlexRay CCs independently from the CC implementation, thus hiding the actual hardware implementation (registers, message buffers, etc.) from upper layers. A single Fr shall support a single type of FlexRay CC. The Fr maps abstract functional requests to sequences of CC specific hardware accesses.

A very detailed description for all API services can be found in chapter 8.

7.2 Indexing Scheme

Users of the Fr identify Fr resources using an indexing scheme as depicted in Figure 3.

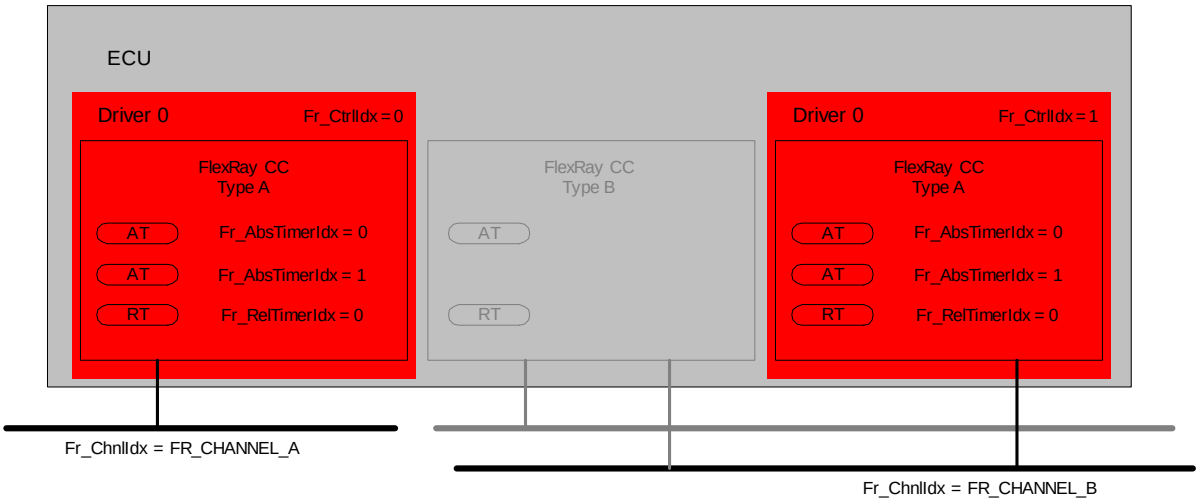

Figure 3 FlexRay Driver indexing scheme

The following Fr resources are available:

- **FR075:** FlexRay Communication Controllers. CCs are identified via controller indices (`Fr_CtrlIdx`). Each driver's CCs are identified by controller indices 0 to (n-1) where n is the number of CCs controlled by the particular Fr.
  - For each FlexRay CC the connected channels are identified by channel indices (`Fr_ChnlIdx`). A dedicated type that holds the enumerations `FR_CHANNEL_A`, `FR_CHANNEL_B` or `FR_CHANNEL_AB` represents the channel index. Channel indices are valid within a tuple <`Fr_CtrlIdx`, `Fr_ChnlIdx`> only.
  - **FR005:** Each FlexRay frame processed by Fr API functions is identified by an LPdu index (`Fr_LPduIdx`). Each LPdu carries the LSdu. Each controller's LPdus are identified by LPdu indices 0 to (n-1) where n is the number of LPdus processed by the corresponding CC. LPdu indices are valid within a tuple

<Fr_CtrlIdx, Fr_LPduIdx> only. An Fr_LPduIdx uniquely identifies the following parameters of a FlexRay frame as a key: {Slot ID, Channel, cycle repetition, base cycle, transmit/receive}.

- Each FlexRay CC contains absolute timers. Absolute FlexRay timers are identified via absolute timer indices (Fr_AbsTimerIdx). Each CC's absolute timers are identified by absolute timer indices 0 to (n-1), where n is the number of absolute timers controlled by the particular CC.
- Each FlexRay CC optionally contains relative timers. Relative FlexRay timers are identified via relative timer indices (Fr_RelTimerIdx). Each CC's relative timers are identified by relative timer indices 0 to (n-1), where n is the number of relative timers controlled by the particular CC.

The FlexRay Driver numbering scheme (Figure 3) assigns indices to these items on a per-driver basis. Note that only the FlexRay CCs handled by one specific Fr (i.e., the FlexRay CCs of type A in the example given) are being assigned indices within the context of this Fr. All other CCs (e.g., the FlexRay CC of type B) are not handled by this Fr and thus no indices have been assigned to these FlexRay CCs within the context of this Fr.

7.3 POC state machine control

Since a FlexRay CC is condition-based, it internally maintains a state machine, the Protocol Operation Control (POC) state machine. The state transitions are both driven by hardware related events as well as commands passed by the Host at the CHI (see [11] for details). The CHI commands driving the POC state machine are incorporated into several FlexRay Driver API functions.

API functions affecting the POC state of a FlexRay CC are:

- Fr_StartCommunication()
- Fr_HaltCommunication()
- Fr_AbortCommunication()
- Fr_SendWUP()
- Fr_ControllerInit()

Figure 4 shows the POC states of the FlexRay CC and the transitions applicable to the FlexRay Driver API functions. Note that

- certain transitions (marked with *)) are performed by a single API function call (Fr_ControllerInit()) invocation.
- certain transitions might be implicitly performed by the FlexRay CC without external command invocation (dotted arrow)

certain transitions specified cannot be performed by the current FlexRay Driver API (not drawn in Figure 4 – compare to [5]).

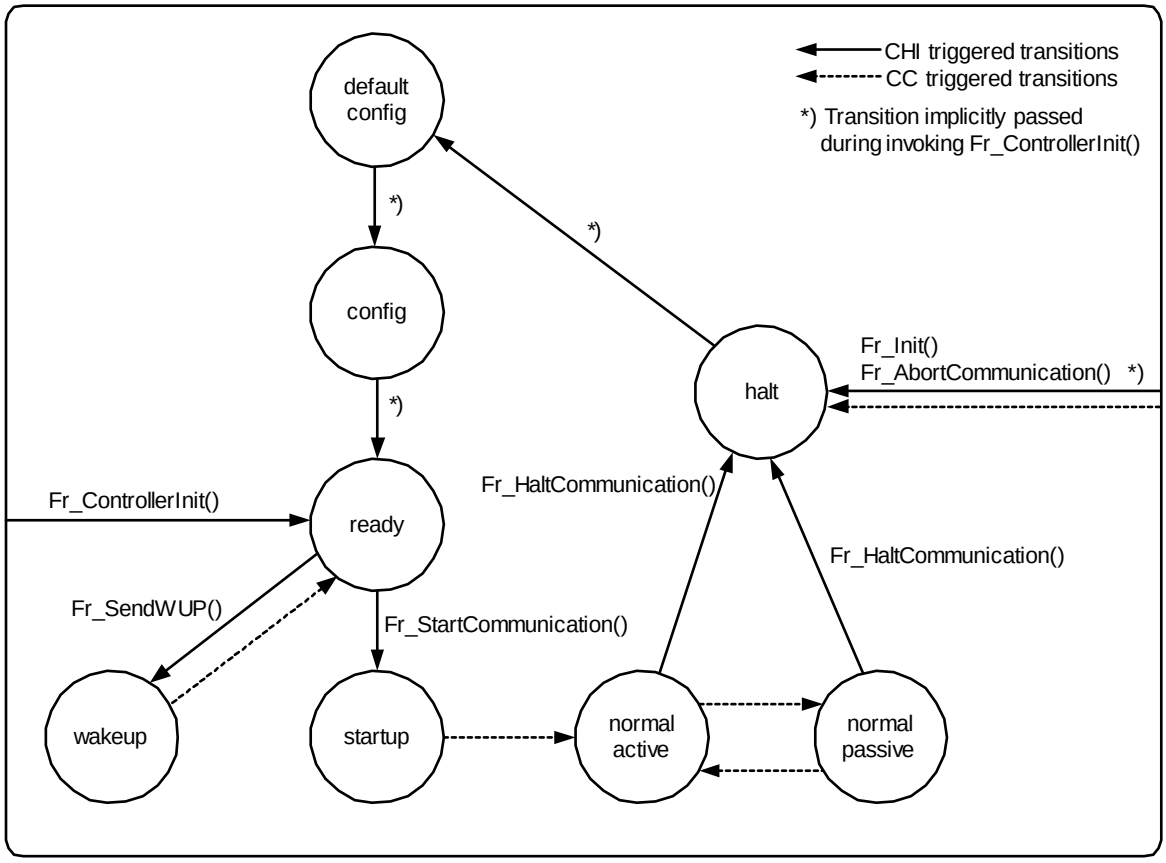

Figure 4 FlexRay Driver POC state machine control

7.4 Implementation Requirements

This chapter lists requirements that shall be fulfilled by Fr implementations.

It shall be ensured that the implementation

- **FR030:** includes a readable software and specification version number in the header-file *Fr.h* (see chapter 10.3).
- **FR031:** performs a consistency check between source- and header-files based on pre-process-checking the version numbers of related source- and header-files.
- **FR029:** checks API parameters are for validity and reports detected errors to the DET in case error detection (DET) is enabled (`FR_DEV_ERROR_DETECT` is `ON`). See chapter 8 for a detailed DET specification for each API function.
- **FR073:** conforms to C programming language in conformance to the HIS subset of the MISRA C Standard (see document [12]).
- **FR076:** replaces all prefixes `Fr` within this specification by a vendor specific prefix `Fr_<Vendor Id>_<Vendor specific name>` for implementation to allow usage of different FlexRay Drivers within one build system. This rule applies to all prefixes in filenames, Fr module specific datatypes, Fr module

specific constants, Fr module specific global variables, API functions and DEM event Ids.

- **FR097:** The API functions specified shall be implemented as real C-code functions and shall not be implemented as macros to allow object code module integration.
- **FR102:** Header-files shall not contain global data definition to avoid multiple data definition caused by multiple header-file inclusion.
- **FR106:** FlexRay timers are stopped in case of loss of synchronization either by software or hardware.

The implementation shall assume that

- **FR104:** LPdu-based services (`Fr_TransmitTxLPdu()`, `Fr_ReceiveRxLPdu()`, `Fr_CheckLPduTxStatus()`, `Fr_PrepareLPdu()`) are called synchronously to the FlexRay global time (at predefined determined points in time) in case of proper system operation.
- **FR105:** all non LPdu-based services might be called at any time independently from the FlexRay global time.

FR019: It shall be possible to enable (switch value `ON`) or disable (switch value `OFF`) the following optional features using pre-compile time switches:

- Development Error Detection and notification of development errors (`FR_DEV_ERROR_DETECT` is either `ON` or `OFF`). If the development error detection is disabled both actions, reporting to DET as well as performing the checks for DET reporting, are not performed meaning that development errors are not detected.
- Version information API (`FR_VERSION_INFO_API` is either `ON` or `OFF`). If the version information API is disabled the following API function is excluded from Fr:
  - `Fr_GetVersionInfo()`
- Support for relative timer (`FR_RELATIVE_TIMER_ENABLE` is either `ON` or `OFF`). If relative timer support is disabled, the following API functions are excluded from the Fr:
  - `Fr_SetRelativeTimer()`
  - `Fr_CancelRelativeTimer()`
  - `Fr_EnableRelativeTimerIRQ()`
  - `Fr_DisableRelativeTimerIRQ()`
  - `Fr_AckRelativeTimerIRQ()`
  - `Fr_GetRelativeTimerIRQStatus()`
- **FR103:** Buffer reconfiguration (`FR_BUFFER_RECONFIG` is either `ON` or `OFF`). If buffer reconfiguration is disabled buffers must not be reconfigured except within API function `Fr_ControllerInit()`. If buffer reconfiguration is disabled, the following API functions are excluded from the Fr:
  - `Fr_PrepareLPdu()`

7.5 Configuration description

FR080: An implementation of the Fr shall provide an XML file that contains the meta 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.

FR055: A driver DMG reads the ECU configuration description of Fr and FrIf. While cluster related configuration parameters are contained in the FrIf configuration description, CC related configuration data is contained in the Fr configuration description. The Fr dependent configuration tool shall read both ECU module descriptions to derive the configuration data for all FlexRay CCs mapped to the Fr.

All frame transmission/reception related configuration is located in the FrIf ECU module description only. The CC configuration data related to frame transmission and reception shall be derived from communication matrix the CC is mapped to within the FrIf. For optimization purposes the Fr DMG should also read the FrIf job list for detecting the points in time certain actions on the Fr will be synchronously invoked by the FrIf (see [7] for the FrIf configuration description). Based on those invocation times the FrIf might decide certain resource alignment optimizations for transmission and reception (buffer assignment).

FR003: If the FrIf job list contains dedicated buffer reconfiguration entries that allow for optimization, the Fr might decide to share one buffer for several FlexRay frames within the static segment.

The Fr DMG shall have knowledge about the capabilities of the CC and the corresponding driver, therefore this tool is called driver dependent. If an Fr DMG is not able to map all required communication operations to the available resources it has to report that conflict.

FR004: The number of FlexRay CCs supported per driver is defined at configuration time. The Fr implementation shall be able to address 4 FlexRay CCs if available in hardware.

FR062: The DMG should ensure the consistency of the generated configuration data.

FR002: All configuration parameters of the Fr are calculated at system configuration time. None of the communication parameters are calculated at runtime.

FR027: The FlexRay driver shall support pre-compile time, link-time and post-build-time configuration.

FR085: 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 (see chapter 8.3.1).

FR072: 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”.

An assignment of those configuration classes to configuration parameters can be found in chapter 10.

A detailed description of all Fr related configuration parameters can be found in chapter 10 of this document. **FR068:** Additionally the configuration description of the FrIf (see chapter 10 of [7]) shall be evaluated for Fr module configuration.

7.6 Error classification

FR025: Errors reported to the DET are of relevance ‘Development’. Diagnostic events that are of relevance ‘Production’ are reported to the DEM. The following errors and diagnostic events shall be detectable by the Fr module:

Description	*Relevance*	*Error / EventId name*	*Value*
parameter timer index exceeds number of available timers	Development	FR_E_INV_TIMER_IDX	0x01
invalid pointer in parameter list	Development	FR_E_INV_POINTER	0x02
parameter offset exceeds bounds	Development	FR_E_INV_OFFSET	0x03
invalid controller index	Development	FR_E_INV_CTRL_IDX	0x04
invalid channel index	Development	FR_E_INV_CHNL_IDX	0x05
parameter cycle exceeds 63	Development	FR_E_INV_CYCLE	0x06
Invalid configuration index	Development	FR_E_INV_CONFIG	0x07
Fr module was not initialized	Development	FR_E_NOT_INITIALIZED	0x08
Fr CC is not in the expected POC state.	Development	FR_E_INV_POCSTATE	0x09
Payload length parameter has an invalid value.	Development	FR_E_INV_LENGTH	0x0A
invalid LPdu index	Development	FR_E_INV_LPDU_IDX	0x0B
Access to FlexRay CC event id	Production	FR_E_ACCESS	Assigned by DEM

FR078: The error values and EventIds are named in capital letters according to the scheme FR_E_<NAME>, where NAME describes the error/EventId and may consist of several words separated by underscores.

7.7 Error notification

FR026: The detection of all development errors shall be configurable (ON/OFF) with the pre-processor switch FR_DEV_ERROR_DETECT. Detected development errors shall be reported to the error hook of the Development Error Tracer (see chapter 8.6) if the pre-processor switch FR_DEV_ERROR_DETECT is ON.

FR057: Development errors shall not be returned by API functions. In case of a development error, API functions shall return E_NOT_OK.

FR028: The status of EventIds shall be reported to the Diagnostic Event Manager (see chapter 8.6).

8 API specification

FR098: All API-functions or global variables, whether they are specified or not shall follow the naming scheme `Fr_<name>`, where the first letter of each word in <name> is written uppercase and the remainder of the word lowercase.

8.1 Imported types

8.1.1 Standard types

FR099: The following types provided by the header file *Std_Types.h* are used within this specification:

- `boolean`
- `uint8`
- `uint16`
- `Std_ReturnType`
- `Std_VersionInfoType`

8.2 Type definitions

FR110: The following type definitions shall be kept in a file named *Fr_GeneralTypes.h* and be protected by a `FR_GENERAL_TYPES` define. If different FlexRay drivers are used, only one instance of this file has to be included in the source tree. The content of *Fr_GeneralTypes.h* consists of types specified within [7], [8] and this document. For implementation all *Fr_GeneralTypes.h* related types in the documents mentioned before shall be considered.

FR077: All types whether they are specified or implementation dependant shall follow the naming scheme `Fr_<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 Fr_ConfigType

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

8.2.2 Fr_SyncStateType

Type:	Enumeration	
Range:	`FR_ASYNC (= 0)`	The local FlexRay CC is asynchronous to the FR global time.
	`FR_SYNC`	The local FlexRay CC is synchronous to the FR global time.

Description:	The values of this enumeration are used to provide information whether or not the local FlexRay CC is synchronous to the FlexRay cluster global time.

8.2.3 Fr_OffsetCorrectionType

Type:	Enumeration	
Range:	`FR_OFFSET_INC (= 0)`	Add the predefined external correction value *vOffsetCorrection* to the CC's offset clock correction process. See chapter "8.6.5 External clock synchronization" in [11].
	`FR_OFFSET_DEC`	Subtract the predefined external correction value *vOffsetCorrection* from the CC's offset clock correction process. See chapter "8.6.5 External clock synchronization" in [11].
	`FR_OFFSET_NOCHANGE`	FR_OFFSET_NOCHANGE – apply no offset correction value.
Description:	These values are used to control the offset correction with service Fr_SetExtSync().	

8.2.4 Fr_RateCorrectionType

Type:	Enumeration	
Range:	`FR_RATE_INC (= 0)`	Add the predefined external correction value *vRateCorrection* to the CC's rate clock correction process. See chapter "8.6.5 External clock synchronization" in [11].
	`FR_RATE_DEC`	Subtract the predefined external correction value *vRateCorrection* from the CC's rate clock correction process. See chapter "8.6.5 External clock synchronization" in [11].
	`FR_RATE_NOCHANGE`	FR_OFFSET_NOCHANGE – apply no rate correction value.
Description:	These values are used to control the rate correction with service Fr_SetExtSync().	

8.2.5 Fr_POCStateType

Type:	Enumeration	
Range:	`FR_POCSTATE_CONFIG (= 0)`	Represents literal CONFIG of formal type definition T_POCState.
	`FR_POCSTATE_DEFAULT_CONFIG`	Represents literal DEFAULT_CONFIG of formal type definition T_POCState.
	`FR_POCSTATE_HALT`	Represents literal HALT of formal type definition T_POCState.
	`FR POCSTATE_NORMAL_ACTIVE`	Represents literal NORMAL_ACTIVE of formal type definition T_POCState.
	`FR_POCSTATE_NORMAL_PASSIVE`	Represents literal NORMAL_PASSIVE of formal type definition T_POCState.
	`FR_POCSTATE_READY`	Represents literal READY of formal type definition T_POCState.
	`FR_POCSTATE_STARTUP`	Represents literal STARTUP of formal type definition T_POCState.
	`FR_POCSTATE_WAKEUP`	Represents literal WAKEUP of formal type definition T_POCState.

Description:	This formal definition refers to the description of type T_POCState in chapter “2.2.1.3 POC status” of [11].

8.2.6 Fr_SlotModeType

Type:	Enumeration	
Range:	`FR_SLOTMODE_SINGLE (= 0)`	Represents literal SINGLE of formal type definition T_SlotMode.
	`FR_SLOTMODE_ALL_PENDING`	Represents literal ALL_PENDING of formal type definition T_SlotMode.
	`FR_SLOTMODE_ALL`	Represents literal ALL of formal type definition T_SlotMode.
Description:	This formal definition refers to the description of type T_SlotMode in chapter “2.2.1.3 POC status” of [11].	

8.2.7 Fr_ErrorModeType

Type:	Enumeration	
Range:	`FR_ERRORMODE_ACTIVE (= 0)`	Represents literal ACTIVE of formal type definition T_ErrorMode.
	`FR_ERRORMODE_PASSIVE`	Represents literal PASSIVE of formal type definition T_ErrorMode.
	`FR_ERRORMODE_COMM_HALT`	Represents literal COMM_HALT of formal type definition T_ErrorMode.
Description:	This formal definition refers to the description of type T_ErrorMode in chapter “2.2.1.3 POC status” of [11].	

8.2.8 Fr_WakeupStatusType

Type:	Enumeration	
Range:	`FR_WAKEUP_UNDEFINED (= 0)`	Represents literal UNDEFINED of formal type definition T_WakeupStatus.
	`FR_WAKEUP_RECEIVED_HEADER`	Represents literal RECEIVED_HEADER of formal type definition T_WakeupStatus.
	`FR_WAKEUP_RECEIVED_WUP`	Represents literal RECEIVED_WUP of formal type definition T_WakeupStatus.
	`FR_WAKEUP_COLLISION_HEADER`	Represents literal COLLISION_HEADER of formal type definition T_WakeupStatus.
	`FR_WAKEUP_COLLISION_WUP`	Represents literal COLLISION_WUP of formal type definition T_WakeupStatus.
	`FR_WAKEUP_COLLISION_UNKNOWN`	Represents literal COLLISION_UNKNOWN of formal type definition T_WakeupStatus.
	`FR_WAKEUP_TRANSMITTED`	Represents literal TRANSMITTED of formal type definition T_WakeupStatus.
Description:	This formal definition refers to the description of type T_WakeupStatus in chapter “2.2.1.3 POC status” of [11].	

8.2.9 Fr_StartupStateType

Type:	Enumeration	
Range:	`FR_STARTUP_UNDEFINED (= 0)`	Represents literal UNDEFINED of formal type definition T_StartupState.

	FR_STARTUP_COLDSTART_LISTEN	Represents literal COLDSTART_LISTEN of formal type definition T_StartupState.
	FR_STARTUP_INTEGRATION_COLDSTART_CHECK	Represents literal INTEGRATION_COLDSTART_CHECK of formal type definition T_StartupState.
	FR_STARTUP_COLDSTART_JOIN	Represents literal COLDSTART_JOIN of formal type definition T_StartupState.
	FR_STARTUP_COLDSTART_COLLISION_RESOLUTION	Represents literal COLDSTART_COLLISION_RESOLUTION of formal type definition T_StartupState.
	FR_STARTUP_COLDSTART_CONSISTENCY_CHECK	Represents literal COLDSTART_CONSISTENCY_CHECK of formal type definition T_StartupState.
	FR_STARTUP_INTEGRATION_LISTEN	Represents literal INTEGRATION_LISTEN of formal type definition T_StartupState.
	FR_STARTUP_INITIALIZE_SCHEDULE	Represents literal INITIALIZE_SCHEDULE of formal type definition T_StartupState.
	FR_STARTUP_INTEGRATION_CONSISTENCY_CHECK	Represents literal INTEGRATION_CONSISTENCY_CHECK of formal type definition T_StartupState.
	FR_STARTUP_COLDSTART_GAP	Represents literal COLDSTART_GAP of formal type definition T_StartupState.
Description:	This formal definition refers to the description of type T_StartupState in chapter “2.2.1.3 POC status” of [11].	

8.2.10 Fr_POCStatusType

Type:	struct	
Elements:	Fr_POCStateType State	
	boolean Freeze	
	boolean CHIHaltRequest	
	boolean ColdstartNoise	
	Fr_SlotModeType SlotMode	
	Fr_ErrorModeType ErrorMode	
	Fr_WakeupStatusType WakeupStatus	
	Fr_StartupState StartupState	
Description:	This formal definition refers to the description of type T_POCStatus in chapter “2.2.1.3 POC status” of [11].	

8.2.11 Fr_TxLPduStatusType

Type:	Enumeration	
Range:	FR_TRANSMITTED (= 0)	LPdu has been transmitted.
	FR_NOT_TRANSMITTED	LPdu has not been transmitted.
Description:	These values are used to determine whether a LPdu has been transmitted or not.	

8.2.12 Fr_RxLPduStatusType

Type:	Enumeration	
Range:	FR_RECEIVED (= 0)	LPdu has been received.
	FR_NOT_RECEIVED	LPdu has not been received.

Description:	These values are used to determine if a LPdu has been received or not.

8.2.13 Fr_MTSStatusType

Type:	Enumeration	
Range:	`FR_MTS_RCV (= 0)`	A valid MTS has been received.
	`FR_MTS_RCV_SYNERR`	A valid MTS has been received and a Syntax Error was detected.
	`FR_MTS_RCV_BVIO`	A valid MTS has been received and a Boundary Violation has been detected.
	`FR_MTS_RCV_SYNERR_BVIO`	A valid MTS has been received and a Syntax Error and a Boundary Violation has been detected.
	`FR_MTS_NOT_RCV`	No valid MTS has been received.
	`FR_MTS_NOT_RCV_SYNERR`	No valid MTS has been received and a Syntax Error was detected.
	`FR_MTS_NOT_RCV_BVIO`	No valid MTS has been received and a Boundary Violation has been detected.
	`FR_MTS_NOT_RCV_SYNERR_BVIO`	No valid MTS has been received and a Syntax Error and a Boundary Violation has been detected.
Description:	These values are derived from chapter “9.3.1.3.5 Symbol window-related data” of [11].	

8.2.14 Fr_ChannelType

Type:	Enumeration	
Range:	`FR_CHANNEL_A (= 0)`	Refers to channel A of a CC.
	`FR_CHANNEL_B`	Refers to channel B of a CC.
	`FR_CHANNEL_AB`	Refers to both channels (A and B) of a CC.
Description:	The values are used to reference channels on a CC.	

8.3 Function definitions

For Fr the following behavior shall be assumed if not specified different for a particular API function:

- API functions have a return type `Std_ReturnType` or `void` (no return code).
- If an API function performs its service successfully it returns `E_OK`, otherwise `E_NOT_OK` in case return type is `Std_ReturnType`.
- Input parameters can be passed immediately as value or by address. If input parameters are passed by address, the API shall use the const qualifier (`const type *`) to guarantee that it doesn’t change the input parameter.
- For output parameters a memory address to store the parameter to is passed as argument.
- If API functions finish successfully (return `E_OK`) it shall be guaranteed that all output parameters were written with valid values.

- If API functions finish erroneous (return `E_NOT_OK`) it shall be guaranteed that no output parameters were written. Output parameters shall keep their original values in this case.
- API functions that have no return type (`void`) shall not have output parameters.

8.3.1 Fr_Init

FR032:

Service name:	Fr_Init	
Syntax:	`void Fr_Init ( const Fr_ConfigType *Fr_ConfigPtr )`	
Service ID:	0x1C	
Sync/Async:	Synchronous	
Re-entrancy:	non re-entrant	
Parameters (in):	`Fr_ConfigPtr`	**FR064:** Address to an Fr dependant configuration structure that contains all information for operating the Fr subsequently.
Parameters (out):	`none`	--
Return value:	`none`	--
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr_ConfigPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return. Monitored Diagnostic Events: • If this API function detects errors while accessing any CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return. CC precondition: None. This API function shall perform the following tasks: 1. Internally store the configuration data address to enable subsequent API calls to access the configuration data. 2. Activate all FlexRay CCs and immediately set them into 'POC:halt' state. 3. If development error detection is enabled (FR_DEV_ERROR_DETECT is ON) the successful initialization shall be remembered internally for other API functions to check for proper module initialization. 4. Return from function. Additionally this API function shall ensure that • no transmission requests are pending. • no reception indications are pending. • no interrupts are pending. • all timers are disabled. • all interrupts are disabled. CC post condition: • All FlexRay CCs controlled by the Fr shall be left in POCState 'POC:halt'.	
Caveats:	--	
Configuration:	--	

8.3.2 Fr_ControllerInit

FR017:

Service name:	Fr_ControllerInit	
Syntax:	`Std_ReturnType Fr_ControllerInit ( uint8 Fr_CtrlIdx, uint8 Fr_LowLevelConfSetIdx, uint8 Fr_BufConfSetIdx )`	
Service ID:	0x00	
Sync/Async:	Synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_LowLevelConfSetIdx`	This parameter is currently not used. Always value 0 shall be passed.
	`Fr_BufConfSetIdx`	This parameter is currently not used. Always value 0 shall be passed.
Parameters (out):	`none`	
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LowLevelConfSetIdx has an invalid value, FR_E_INV_CONF shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_BufConfSetIdx has an invalid value, FR_E_INV_CONF shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Switch CC into ‘POC:config’ (from any other POCState). 2. Configure all FlexRay cluster and node configuration parameters (e.g. cycle length, macrotick duration, ...). 3. Configure all transmit/receive resources (e.g. buffer initialization). 4. Switch CC into ‘POC:ready’ 5. Return E_OK. Additionally this API function shall ensure that • no transmission requests are pending. • no reception indications are pending. • no interrupts are pending. • **FR022:** all timers are disabled.	

	• all interrupts are disabled. CC post condition: • CC Fr_CtrlIdx shall be left in POCState 'POC:ready'.
Caveats:	--
Configuration:	--

8.3.3 Fr_SendMTS

FR023:

Service name:	Fr_SendMTS	
Syntax:	`Std_ReturnType Fr_SendMTS ( uint8 Fr_CtrlIdx, Fr_ChannelType Fr_ChnlIdx )`	
Service ID:	0x01	
Sync/Async:	asynchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_ChnlIdx`	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx (see chapter 7.2). Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_ChnlIdx has an invalid value, FR_E_INV_CHNL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time FR_E_INV_POCSTATE shall be reported to the DET and the API shall return E_NOT_OK. • If the currently configured MTS duration doesn't fit within the currently configured symbol window duration, FR_E_INV_CONFIG shall be reported to the DET and the API shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx is synchronized to FlexRay global time. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Trigger a Media test symbol transmission on FlexRay channel	

	Fr_ChnlIdx. 2. Return E_OK. CC post condition: • No CC POCState change caused by this API invocation.
Caveats:	Since the FlexRay Protocol Specification [11] doesn't specify the MTS transmission interface to the host exactly, different transmit semantics might be implemented in different CCs (e.g. single MTS transmission vs. start/stop MTS transmission semantic). To obtain an abstract behavior Fr_SendMTS() must be called periodically as long as MTS symbols shall be transmitted followed by a single final Fr_StopMTS() API function call.
Configuration:	--

8.3.4 Fr_StopMTS

FR090:

Service name:	Fr_StopMTS	
Syntax:	`Std_ReturnType Fr_StopMTS ( uint8 Fr_CtrlIdx, Fr_ChannelType Fr_ChnlIdx )`	
Service ID:	0x1D	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_ChnlIdx`	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx (see chapter 7.2). Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_ChnlIdx has an invalid value, FR_E_INV_CHNL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Stop the periodic transmission of MTS symbols. 2. Return E_OK. CC post condition:	

	• No CC POCState change caused by this API invocation.
Caveats:	Since the FlexRay Protocol Specification [11] doesn't specify the MTS transmission interface to the host exactly, different transmit semantics might be implemented in different CCs (e.g. single MTS transmission vs. start/stop MTS transmission semantic). To obtain an abstract behavior Fr_SendMTS() must be called periodically as long as MTS symbols shall be transmitted followed by a single final Fr_StopMTS() API function call.
Configuration:	--

8.3.5 Fr_CheckMTS

FR024:

Service name:	Fr_CheckMTS	
Syntax:	`Std_ReturnType Fr_CheckMTS ( uint8 Fr_CtrlIdx, Fr_ChannelType Fr_ChnlIdx, Fr_MTSStatusType *Fr_MTSStatusPtr )`	
Service ID:	0x02	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_ChnlIdx`	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx (see chapter 7.2). Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (out):	`Fr_MTSStatusPtr`	Address the output value is stored to.
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_ChnlIdx has an invalid value, FR_E_INV_CHNL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_MTSStatusPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Read the symbol window status and MTS receive status of the last symbol window and write it to output parameter Fr_MTSStatusPtr. 2. Return E_OK.	

	CC post condition: • No CC POCState change caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.6 Fr_StartCommunication

FR010:

Service name:	Fr_StartCommunication	
Syntax:	`Std_ReturnType Fr_StartCommunication ( uint8 Fr_CtrlIdx )`	
Service ID:	0x03	
Sync/Async:	asynchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
Parameters (out):	`None`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If the CC Fr_CtrlIdx's POCState is not POC:ready, FR_E_INV_POCSTATE shall be reported to the DET and the API shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx is in POCState 'POC:ready'. This API function shall perform the following tasks on FlexRay CC Fr_CtrlIdx: 1. If configuration parameter 'IsLeadingColdstarter' for this CC is set to true, invoke the CC CHI command 'ALLOW_COLDSTART', otherwise don't. 2. Invoke the CC CHI command 'RUN, which initiates the startup procedure within the FlexRay CC. 3. Return E_OK. CC post condition: • This API function call changes the CC POCState to POC:startup which is a transitional state. In case communication startup succeeds the POCState will be changed to 'POC:normal active' or 'POC:normal passive' by the CC.	
Caveats:	It is not guaranteed that the FlexRay CC resides in the 'POC:normal active' or 'POC:normal passive' state after a call to this function.	

Configuration:	--

8.3.7 Fr_HaltCommunication

FR014:

Service name:	Fr_HaltCommunication	
Syntax:	`Std_ReturnType Fr_HaltCommunication ( uint8 Fr_CtrlIdx )`	
Service ID:	0x04	
Sync/Async:	asynchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
Parameters (out):	`None`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time FR_E_INV_POCSTATE shall be reported to the DET and the API shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx is synchronized to the FlerxRay global time. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Invoke the CC CHI command 'HALT, which requests to halt the communication at the end of the current FlexRay communication cycle. 2. Return E_OK. CC post condition: • This API function call requests the halt state which shall be reached by the end of the current FlexRay communication cycle but might not be reached immediately.	
Caveats:	--	
Configuration:	--	

8.3.8 Fr_AbortCommunication

FR011:

Service name:	Fr_AbortCommunication	
Syntax:	`Std_ReturnType Fr_AbortCommunication ( uint8 Fr_CtrlIdx )`	
Service ID:	0x05	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Invoke the CC CHI command 'FREEZE, which immediately aborts communication (if active) and changes to the POC:halt state from any previous POCState. 2. Return E_OK. CC post condition: • The CC shall be left in POCState POC:halt (vPOC!Freeze is set).	
Caveats:	--	
Configuration:	--	

8.3.9 Fr_SendWUP

FR009:

Service name:	Fr_SendWUP	
Syntax:	`Std_ReturnType Fr_SendWUP ( uint8 Fr_CtrlIdx )`	
Service ID:	0x06	
Sync/Async:	asynchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).

Parameters (out):	none	--
Return value:	E_OK	API call finished successfully.
	E_NOT_OK	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If the CC Fr_CtrlIdx's POCState is not POC:ready, FR_E_INV_POCSTATE shall be reported to the DET and the API shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx is in POCState 'POC:ready'. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Invoke the CC CHI command 'WAKEUP, which initiates the wakeup transmission procedure on the configured FlexRay channel. 2. Return E_OK. CC post condition: • This API function call changes the CC POCState to POC:wakeup which is a transitional state. After performing the wakeup procedure the CC will reach POC:ready again.	
Caveats:	Sending a wakeup pattern does not necessarily cause all cluster nodes to be awoken afterwards. This API just invokes the wakeup symbol transmission procedure on a certain FlexRayCC.	
Configuration:	--	

8.3.10 Fr_SetWakeupChannel

FR091:

Service name:	Fr_SetWakeupChannel	
Syntax:	Std_ReturnType Fr_SetWakeupChannel (uint8 Fr_CtrlIdx, Fr_ChannelType Fr_ChnlIdx)	
Service ID:	0x07	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	Fr_ChnlIdx	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx (see chapter 7.2). Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (out):	none	--

Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_ChnlIdx has an invalid value, FR_E_INV_CHNL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If the CC Fr_CtrlIdx's POCState is not POC:ready, FR_E_INV_POCSTATE shall be reported to the DET and the API shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx is in POCState 'POC:ready'. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Change the CC's POCState to POC:config by invoking the CHI command 'CONFIG'. 2. Configure the wakeup channel according to parameter Fr_ChnlIdx. 3. Change the CC's POCState to POC:ready again by invoking the CHI command 'CONFIG_COMPLETE'. 4. Return E_OK. CC post condition: • The CC POCState shall be left in POC:ready.	
Caveats:	--	
Configuration:	--	

8.3.11 Fr_SetExtSync

FR041:

Service name:	Fr_SetExtSync	
Syntax:	`Std_ReturnType Fr_SetExtSync ( uint8 Fr_CtrlIdx, Fr_OffsetCorrectionType Fr_Offset, Fr_RateCorrectionType Fr_Rate )`	
Service ID:	0x08	
Sync/Async:	asynchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_Offset`	Determines the kind of offset correction.
	`Fr_Rate`	Determines the kind of rate correction.

Parameters (out):	none	--
Return value:	E_OK	API call finished successfully.
	E_NOT_OK	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time FR_E_INV_POCSTATE shall be reported to the DET and the API shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx is synchronized to FlexRay global time. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Depending on the parameters Fr_Offset and Fr_Rate, the external clock correction values are added/subtracted/ignored to the internal clock correction procedure of the CC. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation. This function is used to adjust the global time of a FlexRay CC to an external clock source by writing a correction value to a FlexRay CC connected to the cluster.	
Caveats:	The external clock correction value is only applied for one communication cycle and not repetitively!	
Configuration:	--	

8.3.12 Fr_GetSyncState

FR021:

Service name:	Fr_GetSyncState	
Syntax:	Std_ReturnType Fr_GetSyncState (uint8 Fr_CtrlIdx, Fr_SyncStateType *Fr_SyncStatePtr)	
Service ID:	0x09	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
Parameters (out):	Fr_SyncStatePtr	Address the output value is stored to.

Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_SyncStatePtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrlIdx: 1. Evaluate whether the CC is synchronized to the global FlexRay time and write the result to parameter Fr_SyncStatePtr. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.13 Fr_GetPOCStatus

FR012:

Service name:	Fr_GetPOCStatus	
Syntax:	`Std_ReturnType Fr_GetPOCStatus ( uint8 Fr_CtrlIdx, Fr_POCStatusType *Fr_POCStatusPtr )`	
Service ID:	0x0A	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
Parameters (out):	`Fr_POCStatusPtr`	Address the output value is stored to.
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK.	

	• If Fr_SyncStatePtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Query the CC's actual POC status by reading the CHI variable 'vPOC' and write the result to parameter Fr_POCStatusPtr. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.14 Fr_TransmitTxLPdu

FR092:

Service name:	Fr_TransmitTxLPdu	
Syntax:	`Std_ReturnType Fr_TransmitTxLPdu ( uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, const uint8 *Fr_LSduPtr, uint8 Fr_LSduLength )`	
Service ID:	0x0B	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_LPduIdx`	This index is used to uniquely identify a FlexRay frame.
	`Fr_LSduPtr`	This reference points to a buffer where the assembled LSdu to be transmitted within this LPdu is stored at.
	`Fr_LSduLength`	Determines the length of the data (in Bytes) to be transmitted.
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrIIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LPduIdx has an invalid value, FR_E_INV_LPDU_IDX shall be	

	reported to the DET and the API function shall return E_NOT_OK. • If Fr_LSduLength has an invalid value, FR_E_INV_LENGTH shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LSduPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Figure out the physical resource (e.g. a message buffer) mapped to the transmission of the FlexRay frame identified by Fr_LPduIdx. 2. If the LPdu to transmit supports a dynamic payload length (configuration parameter FrIfAllowDynamicLSduLength exists and is TRUE), the CC's transmission resource shall be reconfigured to the length given by the Fr_LSduLength if it is even; otherwise to Fr_LSduLength + 1. 3. Copy Fr_LSduLength bytes from address Fr_LSduPtr into the FlexRay CC's transmission resource mentioned before and activate it for transmission. 4. If the parameter FrIfUnusedBitValue exists, set the remaining bits in the CC's transmission resource to the configured value given by parameter FrIfUnusedBitValue. 5. Return E_OK. The implementation shall ensure that • payload data is transmitted on the FlexRay network in the same byte order as it was passed by Fr_LSduPtr. (first byte = lowest address, last byte = highest address). CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.15 Fr_ReceiveRxLPdu

FR093:

Service name:	Fr_ReceiveRxLPdu	
Syntax:	`Std_ReturnType Fr_ReceiveRxLPdu ( uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, uint8 *Fr_LSduPtr, Fr_RxLPduStatusType *Fr_LPduStatusPtr, uint8 *Fr_LSduLengthPtr )`	
Service ID:	0x0C	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).

	`Fr_LPduIdx`	This index is used to uniquely identify a FlexRay frame.
Parameters (out):	`Fr_LSduPtr`	This reference points to the buffer where the LSdu to be received shall be stored.
	`Fr_RxLPduStatusPtr`	This reference points to the memory location where the status of the LPdu shall be stored.
	`Fr_LSduLengthPtr`	This reference points to the memory location where the length of the LSdu (in bytes) shall be stored. This length represents the number of bytes copied to Fr_LSduPtr.
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LPduIdx has an invalid value, FR_E_INV_LPDU_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LSduPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_RxLPduStatusPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LSduLengthPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Figure out the physical resource (e.g. a message buffer) mapped to the reception of the FlexRay frame identified by Fr_LPduIdx. 2. Figure out whether a new FlexRay frame instance has been received within the receive resource figured out before. 3. If a new FlexRay frame has been received, copy the received payload data to address Fr_LSduPtr, store the number of bytes copied to Fr_LSduLengthPtr and store the status FR_RECEIVED to Fr_RxLPduStatusPtr. 4. If no new frame has been received, dont't copy any payload data to Fr_LSduPtr, write 0 to the parameter Fr_LSduLengthPtr and store the status FR_NOT_RECEIVED to Fr_RxLPduStatusPtr. 5. Return E_OK. The implementation shall ensure that • payload data is copied to Fr_LSduPtr in the same byte order as it was received on the FlexRay bus. (first byte = lowest address, last byte = highest address). • FR_RECEIVED is returned only for valid frames. • FR_RECEIVED is returned only for non-Nullframes. • it returns FR_RECEIVED only once per received frame. • only data of the newly arrived frame is copied. • the number of payload bytes copied to Fr_LSduPtr, and therefore the	

	payload length stored to Fr_LSduLengthPtr are limited by both, the received payload length as well as the configured receive buffer payload length. This enables o the partly reception of large FlexRay frames (e.g. enables local resource optimizations, support for transparent frame extensions). o the reception of short FlexRay frames. (e.g. frames with dynamic payload length). For more information please refer to chapter 9.3.2.2.2.2 of the FlexRay Specification [7]. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.16 Fr_CheckTxLPduStatus

FR094:

Service name:	Fr_CheckTxLPduStatus	
Syntax:	`Std_ReturnType Fr_CheckTxLPduStatus ( uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, Fr_TxLPduStatusType *Fr_TxLPduStatusPtr )`	
Service ID:	0x0D	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_LPduIdx`	This index is used to uniquely identify a FlexRay frame.
Parameters (out):	`Fr_TxLPduStatusPtr`	This reference is used to store the transmit status of the LSdu
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LPduIdx has an invalid value, FR_E_INV_LPDU_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_TxLPduStatusPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None.	

	This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Figure out the physical resource (e.g. a message buffer) mapped to the transmission of the FlexRay frame identified by Fr_LPduIdx. 2. Check whether the transmission resource figured out before is still pending for transmission caused by a previous transmission request (Fr_TransmitTxLPdu()). 3. If a transmission request is pending, store the status FR_NOT_TRANSMITTED to Fr_TxLPduStatusPtr. 4. If no transmission request is pending, store the status FR_TRANSMITTED to Fr_TxLPduStatusPtr. 5. Return E_OK. CC post condition: 1. No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.17 Fr_PrepareLPdu

FR107:

Service name:	Fr_PrepareLPdu	
Syntax:	`Std_ReturnType Fr_PrepareLPdu ( uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx )`	
Service ID:	0x1F	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_LPduIdx`	This index is used to uniquely identify a FlexRay frame.
Parameters (out):	`none`	
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_LPduIdx has an invalid value, FR_E_INV_LPDU_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks on FlexRay CC Fr_CtrIdx: 1. Figure out the physical resource (e.g. a message buffer) mapped to the	

	processing of the FlexRay frame identified by Fr_LPduIdx. 2. Configure the physical resource appropriate for LPduIdx operation if required by Fr configuration. 3. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.18 Fr_GetGlobalTime

FR042:

Service name:	Fr_GetGlobalTime	
Syntax:	`Std_ReturnType Fr_GetGlobalTime ( uint8 Fr_CtrlIdx, uint8 *Fr_CyclePtr, uint16 *Fr_MacroTickPtr )`	
Service ID:	0x10	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
Parameters (out):	`Fr_CyclePtr`	Address where the current FlexRay communication cycle value shall be stored.
	`Fr_MacroTickPtr`	Address where the current macrotick value shall be stored.
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CyclePtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_MacroTickPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • CC Fr_CtrlIdx shall be synchronized to FlexRay global time to obtain valid time information. This API function shall perform the following tasks on FlexRay CC Fr_CtrlIdx: 1. Read the current global FlexRay time and write it to the output	

	parameters Fr_CyclePtr and Fr_MacrotickPtr. 2. Return E_OK. The implementation shall ensure that • **FR044:** the time information is valid and up to date (synchronized CC) – otherwise the ouput parameters shall not be written and E_NOT_OK returned. • the time information is consistent and valid. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.19 Fr_SetAbsoluteTimer

FR033:

Service name:	Fr_SetAbsoluteTimer	
Syntax:	`Std_ReturnType Fr_SetAbsoluteTimer ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx, uint8 Fr_Cycle, uint16 Fr_Offset )`	
Service ID:	0x11	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_AbsTimerIdx`	Index of absolute timer within the context of the FlexRay CC (see chapter 7.2).
	`Fr_Cycle`	Absolute cycle the timer shall elapse in.
	`Fr_Offset`	Offset within cycle Fr_Cycle in units of macrotick the timer shall elapse at.
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_Cycle has an invalid value, FR_E_INV_CYCLE shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_Offset has an invalid value, FR_E_INV_OFFSET shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call	

	Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • The CC Fr_CtrlIdx is related to, shall be synchronized to FlexRay global time for proper timer activation (at the moment of timer activation). This API function shall perform the following tasks: 1. Program the absolute FlexRay timer Fr_AbsTimerIdx according to the parameters Fr_Cycle and Fr_Offset. 2. Return E_OK. The implementation shall ensure that • the timer was programmed successfully, is up and running at the moment of timer programming (synchronized CC) – otherwise E_NOT_OK shall be returned. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.20 Fr_SetRelativeTimer

FR037:

Service name:	Fr_SetRelativeTimer	
Syntax:	`Std_ReturnType Fr_SetRelativeTimer ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx, uint16 Fr_Offset )`	
Service ID:	0x12	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_RelTimerIdx`	Index of relative timer within the context of the FlexRay CC (see chapter 7.2).
	`Fr_MacrotickOffset`	Relative offset from time of service invocation the timer shall elapse at in units of macrotick.
Parameters (out):	`None`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_RelTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_Offset has an invalid value, FR_E_INV_OFFSET should be reported	

	to the DET and the API function should return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: • The CC Fr_CtrlIdx, shall be synchronized to FlexRay global time for proper timer activation (at the moment of timer activation). This API function shall perform the following tasks: 1. Program the absolute FlexRay timer Fr_AbsTimerIdx according to the parameters Fr_Cycle and Fr_Offset. 2. Return E_OK. The implementation shall ensure that • the timer was programmed successfully, is up and running at the moment of timer programming (synchronized CC) – otherwise E_NOT_OK shall be returned. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.21 Fr_CancelAbsoluteTimer

FR095:

Service name:	Fr_CancelAbsoluteTimer	
Syntax:	`Std_ReturnType Fr_CancelAbsoluteTimer ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx )`	
Service ID:	0x13	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_AbsTimerIdx`	Index of absolute timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK.	

	Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Stop the absolute timer Fr_AbsTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.22 Fr_CancelRelativeTimer

FR096:

Service name:	Fr_CancelRelativeTimer	
Syntax:	`Std_ReturnType Fr_CancelRelativeTimer ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx )`	
Service ID:	0x14	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_RelTimerIdx`	Index of relative timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_RelTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Stop the relative timer Fr_RelTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.23 Fr_EnableAbsoluteTimerIRQ

FR034:

Service name:	Fr_EnableAbsoluteTimerIRQ
Syntax:	`Std_ReturnType Fr_EnableAbsoluteTimerIRQ ( uint8 Fr_CtrlIdx,`

	`uint8 Fr_AbsTimerIdx`)	
Service ID:	0x15	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_AbsTimerIdx`	Index of absolute timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Enable the interrupt line related to timer Fr_AbsTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.24 Fr_EnableRelativeTimerIRQ

FR038:

Service name:	Fr_EnableRelativeTimerIRQ	

Syntax:	`Std_ReturnType Fr_EnableRelativeTimerIRQ`
	(
	`uint8 Fr_CtrlIdx,`
	`uint8 Fr_RelTimerIdx`
	)	

Service ID:	0x16	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).

	`Fr_RelTimerIdx`	Index of relative timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`None`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_RelTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Enable the interrupt line related to timer Fr_RelTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.25 Fr_AckAbsoluteTimerIRQ

FR036:

Service name:	Fr_AckAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType Fr_AckAbsoluteTimerIRQ ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx )`	
Service ID:	0x17	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_AbsTimerIdx`	Index of absolute timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`None`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call,	

	FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Reset the interrupt condition of absolute timer Fr_AbsTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	This API call will be called in an interrupt context
Configuration:	--

8.3.26 Fr_AckRelativeTimerIRQ

FR040:

Service name:	Fr_AckRelativeTimerIRQ	
Syntax:	`Std_ReturnType Fr_AckRelativeTimerIRQ ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx )`	
Service ID:	0x18	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_RelTimerIdx`	Index of relative timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`None`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_RelTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call	

	Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Reset the interrupt condition of absolute timer Fr_RelTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	This function will be called in an interrupt context
Configuration:	--

8.3.27 Fr_DisableAbsoluteTimerIRQ

FR035:

Service name:	Fr_DisableAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType Fr_DisableAbsoluteTimerIRQ ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx )`	
Service ID:	0x19	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_AbsTimerIdx`	Index of absolute timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Disable the interrupt line related to absolute timer Fr_AbsTimerIdx. 2. Return E_OK.	

	CC post condition: • No CC POCState change is caused by this API invocation.
Caveats:	--
Configuration:	--

8.3.28 Fr_DisableRelativeTimerIRQ

FR039:

Service name:	Fr_DisableRelativeTimerIRQ	

Syntax:	`Std_ReturnType Fr_DisableRelativeTimerIRQ`
	`(`
	`  uint8 Fr_CtrlIdx,`
	`  uint8 Fr_RelTimerIdx`
	`)`	

Service ID:	0x1A	
Sync/Async:	synchronous	
Reentrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_RelTimerIdx`	Index of relative timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`none`	--
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Disable the interrupt line related to absolute timer Fr_RelTimerIdx. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.29 Fr_GetAbsoluteTimerIRQStatus

FR108:

Service name:	Fr_GetAbsoluteTimerIRQStatus	
Syntax:	`Std_ReturnType Fr_GetAbsoluteTimerIRQStatus ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx, boolean *Fr_IRQStatusPtr )`	
Service ID:	0x20	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_AbsTimerIdx`	Index of absolute timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`Fr_IRQStatusPtr`	Address the output value is stored to.
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_AbsTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_IRQStatusPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Check whether the interrupt of absolute timer Fr_AbsTimerIdx is pending. Write TRUE to output parameter Fr_IRQStatusPtr in case the interrupt is pending, FALSE otherwise. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.30 Fr_GetRelativeTimerIRQStatus

FR109:

Service name:	Fr_GetRelativeTimerIRQStatus	
Syntax:	`Std_ReturnType Fr_GetRelativeTimerIRQStatus ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx, boolean *Fr_IRQStatusPtr )`	
Service ID:	0x20	
Sync/Async:	synchronous	
Re-entrancy:	non re-entrant for the same device	
Parameters (in):	`Fr_CtrlIdx`	Index of FlexRay CC within the context of the FlexRay Driver (see chapter 7.2).
	`Fr_RelTimerIdx`	Index of relative timer within the context of the FlexRay CC (see chapter 7.2).
Parameters (out):	`Fr_IRQStatusPtr`	Address the output value is stored to.
Return value:	`E_OK`	API call finished successfully.
	`E_NOT_OK`	API call aborted due to errors.
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If Fr was not initialized successfully prior to this API function call, FR_E_NOT_INITIALIZED shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_CtrlIdx has an invalid value, FR_E_INV_CTRL_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_RelTimerIdx has an invalid value, FR_E_INV_TIMER_IDX shall be reported to the DET and the API function shall return E_NOT_OK. • If Fr_IRQStatusPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return E_NOT_OK. Monitored Diagnostic Events: • If this API function detects errors while accessing the CC, it should call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK. CC precondition: None. This API function shall perform the following tasks: 1. Check whether the interrupt of relative timer Fr_RelTimerIdx is pending. Write TRUE to output parameter Fr_IRQStatusPtr in case the interrupt is pending, FALSE otherwise. 2. Return E_OK. CC post condition: • No CC POCState change is caused by this API invocation.	
Caveats:	--	
Configuration:	--	

8.3.31 Fr_GetVersionInfo

FR070:

Service name:	Fr_GetVersionInfo
Syntax:	`void Fr_GetVersionInfo {`

	`Std_VersionInfoType *VersioninfoPtr` `}`
Service ID [hex]:	0x1B
Sync/Async:	Synchronous
Reentrancy:	non reentrant
Parameters (in):	`none` --
Parameters (out):	`versioninfo` Pointer to where to store the version information of this module.
Return value:	none
Description:	Development error checking and reporting (executed only if FR_DEV_ERROR_DETECT is ON): • If VersionInfoPtr equals NULL_PTR, FR_E_INV_POINTER shall be reported to the DET and the API function shall return. This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers.
Caveats:	--
Configuration:	--

8.4 Call-back notifications

The FlexRay driver does not call any callbacks.

8.5 Scheduled functions

The FlexRay driver is executed in the context of the FlexRay Interface so it has no function to be scheduled.

8.6 Expected Interfaces

In this chapter, all interfaces required from other modules are listed.

8.6.1 Mandatory Interfaces

This chapter defines all interfaces that are required to fulfill the core functionality of the module.

API function	***Module***	***Description***
`Dem_ReportErrorStatus()`	Diagnostic event manager	--

8.6.2 Optional Interfaces

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

API function	*Module*	*Description*	*Configuration parameter (description see chapter 10)*
`Det_ReportError()`	Development error notification	Development error notification	`FR_DEV_ERROR_DETECT`
`SchM_Enter_Fr()`	BSW Scheduler	Enter an exclusive area	Not applicable
`SchM_Exit_Fr()`	BSW Scheduler	Exit an exclusive area	Not applicable

Further optional interfaces might be accessed in case the Fr uses other modules for accessing the CC hardware.

8.6.3 Configurable interfaces

There are no configurable interfaces related to the FlexRay driver.

9 Sequence diagrams

The usage of the driver is depicted in the Sequence diagrams of the FlexRay Interface.

10 Configuration specification

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

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

Chapter 10.3 specifies published information of the module FlexRay Driver.

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 [4]
  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: before compile time, before link time or after 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.

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

FR067: 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 - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
x	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* 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*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters.

10.2.1 Variants

There are no variants.

10.2.2 Fr

SWS Item	**FR082:**
Container Name	Fr
Description	This container contains the configuration of the Fr (FlexRay driver).
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrGeneral	1	module
FrController	1…*	module/FrIf

10.2.3 FrGeneral

SWS Item	**FR086:**
Container Name	FrGeneral
Description	This container is a subcontainer of Fr and specifies the general configuration parameters of the Fr.
Configuration Parameters	

Name	FrNumCtrlSupported		
Description	Determines the maximum number of communication controllers that the driver supports.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	1..255		
Configuration Class	***Pre-compile***	--	--
	Link time	x	--
	Post Build	--	--
Scope	Module		
Dependency	None		
Multplicity	1		

Name	FrNumCtrl
Description	Number of FlexRay CCs controlled by the FlexRay driver
Type	INTEGER-PARAM-DEF
Unit	--

Range	1..255		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multplicity	1		

Name	FrDevErrorDetect		
Description	Switches the Development Error Detection and Notification ON or OFF. Pre-compile time switch FR_DEV_ERROR_DETECT is derived from this configuration parameter.		
Type	BOOLEAN-PARAM-DEF		
Range	true	development error detection enabled	
	false	development error detection disabled	
Configuration Class	Pre-compile	x	--
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	FrRelativeTimerEnable		
Description	Enables or disables the usage of relative timers. Pre-compile time switch FR_RELATIVE_TIMER_ENABLE is derived from this configuration parameter.		
Type	BOOLEAN-PARAM-DEF		
Range	true	relative timer API enabled	
	false	relative timer API disabled	
Configuration Class	Pre-compile	x	--
	Link time	--	--
	Post Build	--	--
Scope	moduel/FrIf		
Dependency	none		
Multplicity	1		

Name	FrVersionInfoApi		
Description	Enables/disables the existence of the Fr_GetVersionInfo API. Pre-compile time switch FR_VERSION_INFO_API is derived from this configuration parameter.		
Type	BOOLEAN-PARAM-DEF		
Range	true	version info API enabled	
	false	version info API disabled	
Configuration Class	Pre-compile	x	--
	Link time	--	--
	Post Build	--	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	FrModulePrefix		
Description	Specifies the module prefix of the actual Fr implementation (see BSW 00347).		
Type	STRING-PARAM-DEF		
Range	--		
Configuration Class	Pre-compile	x	--
	Link time	--	--

	Post Build	--	--
Scope	FrIf		
Dependency	none		
Multiplicity	1		

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

10.2.4 FrController

SWS Item	**FR087:**
Container Name	FrController
Description	This container is a subcontainer of Fr and specifies the Fr controller specific configuration parameters.
Configuration Parameters	

Name	IsLeadingColdStarter		
Description	Enables/Disables the leading transmission of SUP frames.		
Type	BOOLEAN-PARAM-DEF		
Range	true	enables leading coldstart	
	false	disables leading coldstart	
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	x	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	FrCtrlIdx		
Description	Determines index of CC within Fr.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0…255		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module/FrIf		
Dependency	None		
Multiplicity	1		

Name	FrCtrlClock		
Description	Determines clock connected to the CCs of the driver [Hz].		
Type	INTEGER-PARAM-DEF		
Unit	Hz		
Range	0…80000000		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PAllowHaltDueToClock		
Description	Boolean flag that controls the transition to the POC:halt state due to a clock synchronization errors. If set to true, the CC is allowed to transition to POC:halt. If set to false, the CC will not transition to the POC:halt state but will enter or remain in the POC:normal passive state (self healing would still be possible).		
Type	BOOLEAN-PARAM-DEF		
Range	true	--	
	false	--	
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	x	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	PAllowPassiveToActive		
Description	Number of consecutive even/odd cycle pairs that must have valid clock correction terms before the CC will be allowed to transition from the POC:normal passive state to POC:normal active state. If set to zero, the CC is not allowed to transition from POC:normal passive to POC:normal active.		
Type	BOOLEAN-PARAM-DEF		
Range	true	--	
	false	--	
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	x	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	PClusterDriftDamping		
Description	Local cluster drift damping factor used for rate correction [Microticks]		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…20		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PDecodingCorrection		
Description	Value used by the receiver to calculate the difference between primary time reference point and secondary time reference point [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	14…143		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PDelayCompensationA		
Description	Value used to compensate for reception delays on the indicated channel. This covers assumed propagation delay up to cPropagationDelayMax for microticks in the range of 0.0125 us to 0.05 us. In practice, the minimum of the propagation delays of all sync nodes should be applied. [Microticks]		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…200		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PDelayCompensationB		
Description	Value used to compensate for reception delays on the indicated channel. This covers assumed propagation delay up to cPropagationDelayMax for microticks in the range of 0.0125 us to 0.05 us. In practice, the minimum of the propagation delays of all sync nodes should be applied. [Microticks]		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…200		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PExternOffsetCorrection		
Description	Number of microticks added or subtracted to the NIT to carry out a host-requested external offset correction. [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…7		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PExternRateCorrection		
Description	Number of microticks added or subtracted to the cycle to carry out a host-requested external rate correction [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…7		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PKeySlotId		
Description	ID of the slot used to transmit the startup frame, sync frame, or designated single slot frame.		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	1…1023		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	0…1		

Name	PKeySlotUsedForStartup		
Description	Flag indicating whether the Key Slot is used to transmit a startup frame.		
Type	BOOLEAN-PARAM-DEF		
Range	true	--	
	false	--	
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	x	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	PKeySlotUsedForSync		
Description	Flag indicating whether the Key Slot is used to transmit a sync frame.		
Type	BOOLEAN-PARAM-DEF		
Range	true	--	
	false	--	
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	x	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	PLatestTx		
Description	Number of the last minislot in which a frame transmission can start in the dynamic segment [Minislots].		
Type	INTEGER-PARAM-DEF		
Unit	Minislots		
Range	0…7981		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PMacroInitialOffsetA		
Description	Integer number of macroticks between the static slot boundary and the following macrotick boundary of the secondary time reference point based on the nominal macrotick duration [Macroticks].		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	2…72		

Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PMacroInitialOffsetB		
Description	Integer number of macroticks between the static slot boundary and the following macrotick boundary of the secondary time reference point based on the nominal macrotick duration [Macroticks].		
Type	INTEGER-PARAM-DEF		
Unit	Macroticks		
Range	2…72		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PMicroInitialOffsetA		
Description	Number of microticks between the closest macrotick boundary described by pMacroInitialOffset[Ch] and the secondary time reference point. The parameter depends on pDelayCompensation[Ch] and therefore it has to be set independently for each channel [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…240		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PMicroInitialOffsetB		
Description	Number of microticks between the closest macrotick boundary described by pMacroInitialOffset[Ch] and the secondary time reference point. The parameter depends on pDelayCompensation[Ch] and therefore it has to be set independently for each channel [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…240		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PMicroPerCycle
Description	Nominal number of microticks in the communication cycle of the local node. If nodes have different microtick durations this number will differ from node to node [Microticks].
Type	INTEGER-PARAM-DEF
Unit	Microticks
Range	640…640000

Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PMicroPerMacroNom		
Description	Number of microticks per nominal macrotick that all implementations must support [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	40…240		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	POffsetCorrectionOut		
Description	Magnitude of the maximum permissible offset correction value [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	5…15266		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PPayloadLengthDynMax		
Description	Maximum payload length for dynamic frames [16bit words].		
Type	INTEGER-PARAM-DEF		
Unit	16bit words		
Range	0…127		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PRateCorrectionOut		
Description	Magnitude of the maximum permissible rate correction value [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	2…1923		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PSamplesPerMicrotick		
Description	Number of samples per microtick.		
Type	ENUMERATION-PARAM-DEF		
Range	N1SAMPLES, N2SAMPLES, N4SAMPLES		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PSingleSlotEnabled		
Description	Flag indicating whether or not the node shall enter single slot mode following startup.		
Type	BOOLEAN-PARAM-DEF		
Range	true	--	
	false	--	
Configuration Class	Pre-compile	--	--
	Link time	--	--
	Post Build	x	--
Scope	module		
Dependency	none		
Multplicity	1		

Name	PWakeupChannel		
Description	Channel used by the node to send a wakeup pattern.		
Type	ENUMERATION-PARAM-DEF		
Range	FR_CHANNEL_A, FR_CHANNEL_B		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PWakeupPattern		
Description	Number of repetitions of the wakeup symbol that are combined to form a wakeup pattern when the node enters the POC:wakeup send state.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	2…63		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PChannels		
Description	Channels to which the node is connected		
Type	ENUMERATION-PARAM-DEF		
Range	FR_CHANNEL_A, FR_CHANNEL_AB, FR_CHANNEL_B		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		

Dependency	None		
Multiplicity	1		

Name	PdAcceptedStartupRange		
Description	Expanded range of measured clock deviation allowed for startup frames during integration [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	0…1875		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PdListenTimeout		
Description	Upper limit for the start up listen timeout and wake up listen timeout [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	1284…1283846		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PdMaxDrift		
Description	Maximum drift offset between two nodes that operate with unsynchronized clocks over one communication cycle [Microticks].		
Type	INTEGER-PARAM-DEF		
Unit	Microticks		
Range	2…1923		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Name	PdMicrotick		
Description	Duration of a microtick.		
Type	ENUMERATION-PARAM-DEF		
Range	T12NS, T25NS, T50NS, T100NS, T200NS		
Configuration Class	***Pre-compile***	--	--
	Link time	--	--
	Post Build	x	--
Scope	Module		
Dependency	None		
Multiplicity	1		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrAbsoluteTimer	1…*	--
FrRelativeTimer	0…*	--

10.2.5 FrAbsoluteTimer

SWS Item	**FR088:**
Container Name	FrAbsoluteTimer
Description	This container is a subcontainer of Fr and specifies the absolute timer configuration parameters of the Fr.
Configuration Parameters	

Name	FrAbsTimerIdx		
Description	Contains the index of an absolute timer contained in Fr on a certain FlexRay CC.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0..254		
Configuration Class	***Pre-compile***	--	--
	Link time	x	--
	Post Build	--	--
Scope	Module		
Dependency	None		
Multplicity	1		

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

10.2.6 FrRelativeTimer

SWS Item	**FR089:**
Container Name	FrRelativeTimer
Description	This container is a subcontainer of Fr and specifies the relative timer configuration parameters of the Fr.
Configuration Parameters	

Name	FrRelTimerIdx		
Description	Contains the index of a relative timer contained in Fr on a certain FlexRay CC.		
Type	INTEGER-PARAM-DEF		
Unit	--		
Range	0..254		
Configuration Class	***Pre-compile***	--	--
	Link time	x	--
	Post Build	--	--

Scope	Module
Dependency	None
Multplicity	1

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

10.3 Published parameters

SWS Item	**FR069:**	
Information elements		
Information element name	***Type / Range***	***Information element description***
FR_VENDOR_ID	#define	Vendor ID of the dedicated implementation of this module according to the AUTOSAR vendor list
FR_MODULE_ID	#define	Module ID of this module from Module List
FR_AR_MAJOR_VERSION	#define	Major version number of AUTOSAR specification where the appropriate implementation is based on.
FR_AR_MINOR_VERSION	#define	Minor version number of AUTOSAR specification where the appropriate implementation is based on.
FR_AR_PATCH_VERSION	#define	Patch level version number of AUTOSAR specification where the appropriate implementation is based on.
FR_SW_MAJOR_VERSION	#define	Major version number of the vendor specific implementation of the module. The numbering is vendor specific.
FR_SW_MINOR_VERSION	#define	Minor version number of the vendor specific implementation of the module. The numbering is vendor specific.
FR_SW_PATCH_VERSION	#define	Patch level version number of the vendor specific implementation of the module. The numbering is vendor specific.

11 Release Change History

11.1 Changes from Release 1.0 to Release 2.0

11.1.1 Deleted SWS Items

SWS Item	*Rationale*
--	Type 'Fr_ReturnTypes' removed because errors are no longer returned via API, they are reported to the diagnostic event handler.
--	API function Fr_ProvideTxLSduPtr removed because it breaks hardware abstraction, may cause alignment problems.
--	API function Fr_CommitTxLSdu removed because it breaks hardware abstraction, may cause alignment problems.
--	API function Fr_ProvideRxLSduPtr removed because it breaks hardware abstraction, may cause alignment problems.
--	API function Fr_ReleaseRxLSduPtr removed because it breaks hardware abstraction, may cause alignment problems.
---	FR_USE_HW_ACCESS removed, because no longer necessary.

11.1.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
--	--	--

11.1.3 Changed SWS Items

SWS Item	*Rationale*
--	Changed Fr_LSduInfoType : BufferConfigPtr renamed to BufferSelectorPtr.
--	Changed Fr_AbsTimeInfoType : MacrotickOffset changed to uint16.
--	Changed Fr_RelativeTimerInfoType to MacrotickOffset changed to uint16.
--	Changed Fr_StartupStateType: Changed two enum names.
--	Changed Fr_WakeupStateType: Removed since not needed anymore.

11.1.4 Added SWS Items

SWS Item	*Rationale*
FR017, FR016, FR018	Added Fr_Init(): Initialization function
FR015	Added Fr_TransmitTxLSdu(): hardware abstracted transmit function
FR015	Added Fr_ReceiveTxLSdu(): hardware abstracted receive function
--	Added Fr_RxLSduStatusType(): status of a received LSdu
FR070	Added Fr_GetVersionInfo(): contained in new SWS-template
FR018	Added Fr_ConfigAllBuffers(): used to configure all transmit/receive buffers of a CC
FR049	Added Fr_CancelAbsoluteTimer(): used to cancel an absolute timer so that timer period can be changed
FR049	Added Fr_CancelRelativeTimer(): used to cancel a relative timer so that the timer period can be changed

11.2 Changes from Release 2.0 to Release 2.1

11.2.1 Deleted SWS Items

SWS Item	*Rationale*
FR065	Not an Fr specification element.
FR020	Already contained in FR012.
FR016	Already contained in FR017.
FR018	Already contained in FR017.
FR043	Redundant to FR044.
FR049	Not an Fr specification element.

11.2.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
FR015	FR092, FR093, FR094, FR107	More precise specification elements.

11.2.3 Changed SWS Items

SWS Item	*Rationale*
FR066	Added dependency to module BSW scheduler.
FR068	Added configuration dependency to FrIf.
FR019	Added pre-compile-time switch FR_VERSION_INFO_API.
FR005	Replaced LSdu by LPdu.
FR082	Single SWS item now maps to single specification element. Removed multiple usage of FR082.
FR032	Reworked API-error checks and pre-conditions. Leaves CC in POC-state ‘halt’ instead of POC-state ‘default-config’.
FR017	Reworked API-error checks and pre-conditions.
FR032	Reworked API-error checks and pre-conditions.
FR024	Reworked API-error checks and pre-conditions.
FR010	Reworked API-error checks and pre-conditions.
FR014	Reworked API-error checks and pre-conditions.
FR011	Reworked API-error checks and pre-conditions.
FR009	Reworked API-error checks and pre-conditions.
FR041	Reworked API-error checks and pre-conditions.
FR021	Reworked API-error checks and pre-conditions.
FR012	Reworked API-error checks and pre-conditions.
FR042	Reworked API-error checks and pre-conditions.
FR033	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR037	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR034	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR038	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR036	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR040	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR035	Reworked API-error checks and pre-conditions.

	Changed API parameters – timer index is now controller related.
FR039	Reworked API-error checks and pre-conditions. Changed API parameters – timer index is now controller related.
FR064	Changed API parameter type.

11.2.4 Added SWS Items

SWS Item	*Rationale*
FR086	Added distinct SWS item for already existing specification element.
FR087	Added distinct SWS item for already existing specification element. Change ENUMERATION-PARAM-DEF values to Uppercase.
FR088	Added distinct SWS item for already existing specification element.
FR089	Added distinct SWS item for already existing specification element.
FR090	Added new API function.
FR091	Added distinct SWS item for already existing specification element. Reworked API-error checks and pre-conditions.
FR092	Added distinct SWS item for already existing specification element, which was renamed. Reworked API-error checks and pre-conditions.
FR093	Added distinct SWS item for already existing specification element, which was renamed. Reworked API-error checks and pre-conditions.
FR094	Added distinct SWS item for already existing specification element, which was renamed. Reworked API-error checks and pre-conditions.
FR095	Added distinct SWS item for already existing specification element. Reworked API-error checks and pre-conditions.
FR096	Added distinct SWS item for already existing specification element. Reworked API-error checks and pre-conditions.
FR097	Added to fullfill BSW requirement.
FR098	Added to fullfill BSW requirement.
FR099	Added to fullfill BSW requirement.
FR100	Added distinct SWS item for already existing specification element.
FR101	Added distinct SWS item for already existing specification element.
FR102	Added to fullfill BSW requirement.
FR103	Added due to new specification element.
FR104	Added due to new specification element.
FR105	Added due to new specification element.
FR106	Added distinct SWS item for already existing specification element.
FR107	Added due to new specification element.
FR108	Added due to new specification element.
FR109	Added due to new specification element.
FR110	Added distinct SWS item for already existing specification element.
FR111	Added to fullfill BSW requirement.
FR112	Added to fullfill BSW requirement.