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

Document Version	2.4.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	2.4.0	AUTOSAR Release Management	• Added API function Fr_GetWakeupRxStatus • Editorial changes • Removed chapter(s) on change documentation
19.04.2011	2.3.0	AUTOSAR Administration	• Added API functions for reading FlexRay status information • Added functionality to verify the FlexRay controller configuration • Added basic Rx-FIFO support • Added support for FlexRay 3.0 protocol compliant FlexRay controllers
23.06.2008	2.2.1	AUTOSAR Administration	Legal disclaimer revised
28.11.2007	2.2.0	AUTOSAR Administration	• Added NM-Vector Support • Added dynamic frame length support for dynamic FlexRay segment • Added API service for coldstart control • Document meta information extended • Small layout adaptations made

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
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

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

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 [12] are encapsulated within the Fr module 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 module. Thus, the FlexRay Interface (FrIf) as the user of the Fr module is independent of the underlying FlexRay CC hardware. The Fr module doesn't have a main-function or an ISR. All Fr module API functions are executed in the context of the FrIf only.

A single Fr module 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 module 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 module shall be done at system configuration time, with the Fr module's specific configuration data being generated by a Module Configuration Generator (MCG), which translates the parameters out of the ECU configuration description to Fr module specific configuration data structures.
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 module's API. The usage of optional features is encapsulated within the Fr module. 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.

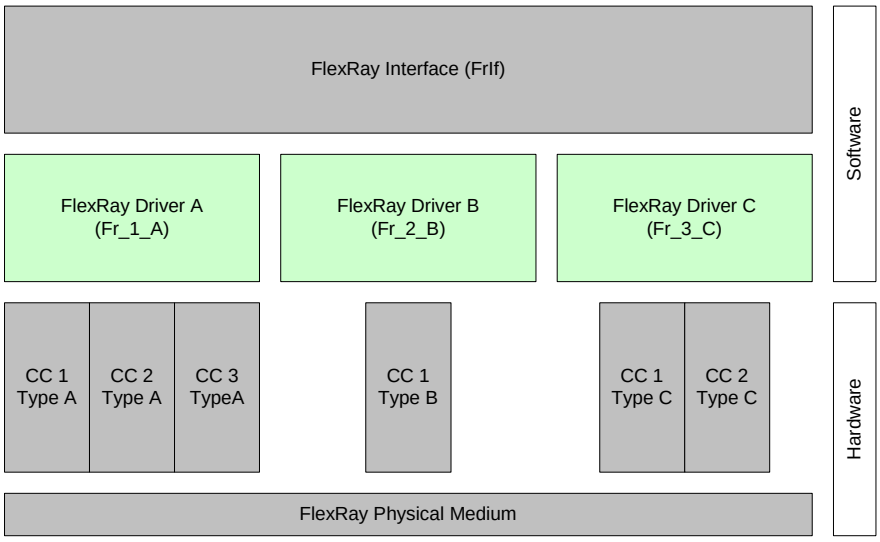

Figure 1: FlexRay stack module overview

2 Acronyms and abbreviations

Abbreviation:	*Description:*
API	Application Programming Interface
DEM	Diagnostic Event Manager
DET	Development Error Tracer
MCG	Module Configuration Generator
CC	Communication Controller
CHI	Controller Host Interface
Fr	FlexRay Driver
FrIf	FlexRay Interface
POC	Protocol Operation Control (see [12] 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 [12] 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 [12] 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:
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture:
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] General Requirements on Basic Software Modules:
AUTOSAR_SRS_General.pdf

[4] Specification of ECU Configuration:
AUTOSAR_ECU_Configuration.pdf

[5] Specification of Communication Stack Types,:
AUTOSAR_SWS_ComStackTypes.pdf

[6] Specification of Platform Types:
AUTOSAR_SWS_PlatformTypes.pdf

[7] Specification of FlexRay Interface:
AUTOSAR_SWS_FlexRayInterface.pdf

[8] Specification of FlexRay Transceiver Driver:
AUTOSAR_SWS_FlexRayTransceiver.pdf

[9] Specification of BSW Scheduler:
AUTOSAR_SWS_Scheduler.pdf

[10] Specification of Memory Mapping:
AUTOSAR_SWS_MemoryMapping.pdf

[11] AUTOSAR Basic Software Module Description Template:
AUTOSAR_BSW_Module_Description.pdf

3.2 Related standards and norms

[12] 2009, FlexRay Consortium, FlexRay Communication Systems Protocol Specification, Version 3.0 (unreleased draft).

[13] 2005, FlexRay Consortium, FlexRay Communication Systems Protocol Specification, Version 2.1 Revision A.

[14] 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 [12]) is required. Furthermore it enables the synchronized operation of several ECUs within a car.

5 Dependencies to other modules

This chapter lists the modules interacting with the Fr module.

Modules that use Fr module:

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

Modules used by the Fr module:

- The Fr module shall use the Development Error Tracer (DET) for reporting of development errors.
- The Fr module shall use the Diagnostic Event Manager (DEM) for reporting of diagnostic-relevant events and states.
- The Fr module shall use the BSW Scheduler mechanisms for data consistency if required.

Other Module dependencies:

- 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 module doesn't take care of configuring those components but requires that their initialization precede the Fr module's initialization.

5.1 File structure

This section gives an overview about the files and their relations required for a proper Fr module 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 section.

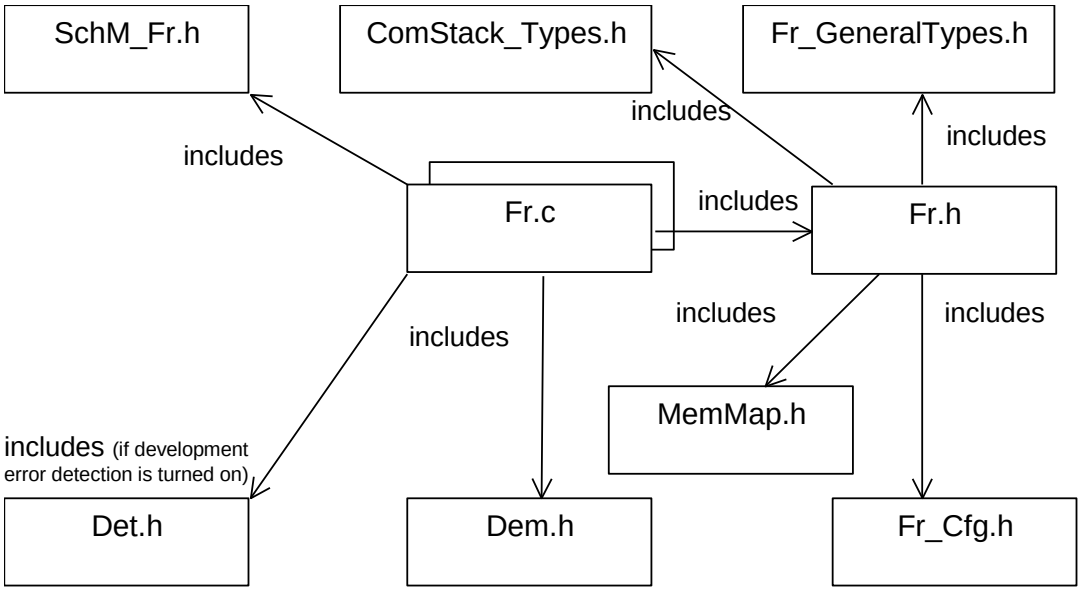

Figure 2 FlexRay Driver header file structure

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

FR115: The implementation of the Fr module 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.

FR116: The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- Fr_Lcfg.c – for link time configurable parameters and
- Fr_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.

5.1.2 Header file structure

FR101: The implementation of the Fr module shall provide the header file *Fr.h*, which is the main module interface file. It shall contain all types and function prototypes required by the Fr module's environment.

FR063*:* The implementation of the Fr module shall provide the header file *Fr_Cfg.h* that shall contain the pre-compile-time configuration parameters.

FR117*: FrGeneralTypes.h* shall contain all types and constants that are shared among the AUTOSAR FlexRay modules Fr, FrIf and FrTrcv. The integrator of the FlexRay modules shall provide this file.

FR071: The Fr 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*.

FR118: The Fr module source code file(s) shall include *Det.h* file. By this inclusion the APIs to report development errors are included.

FR111: The Fr module source code file(s) shall include *SchM_Fr.h* if data consistency mechanisms of the BSW scheduler are required as described in [9].

FR112: The Fr 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	Chapter 7.5
BSW167	Static configuration checking	Chapter 7.5
BSW171	Configurability of optional functionality	Chapter 7.5
BSW170	Data for reconfiguration of AUTOSAR SW-Components	Chapter 5
BSW00380	Separate C-Files for configuration parameters	FR115, FR116
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	Chapter 5
BSW00384	List dependencies to other modules	Chapter 5
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, FR027
BSW00398	Link-time parameters	FR082, FR086, FR087, FR088, FR089, FR115, FR116, FR027
BSW00399	Loadable Post-build time parameters	FR082, FR086, FR087, FR088, FR089, FR027, FR115, FR116
BSW00400	Selectable Post-build time parameters	FR082, FR086, FR087, FR088, FR089, FR027, FR115, FR116, FR032
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,
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, FR115, FR116,
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, 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,
BSW00408	Configuration parameter naming convention	FR082, FR086, FR087, FR088, FR089
BSW00410	Compiler switches shall have defined values	not applicable to Fr SWS
BSW00411	Get version info keyword	FR070, FR340, FR341, FR342
BSW00346	Basic set of module files	FR115, FR116
BSW158	Separation of configuration from implementation	FR115, FR116
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, Figure 2
BSW00353	Platform specific type header	FR099, Figure 2
BSW00361	Compiler specific language extension header	FR099, Figure 2
BSW00301	Limit imported information	Figure 2
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
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	Chapter 1
BSW05024	Abstraction from CC Buffer Configuration	FR005, FR092, FR093, FR094, FR440, FR441, FR442
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, FR123
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	FR123, 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

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.

The Fr module 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.

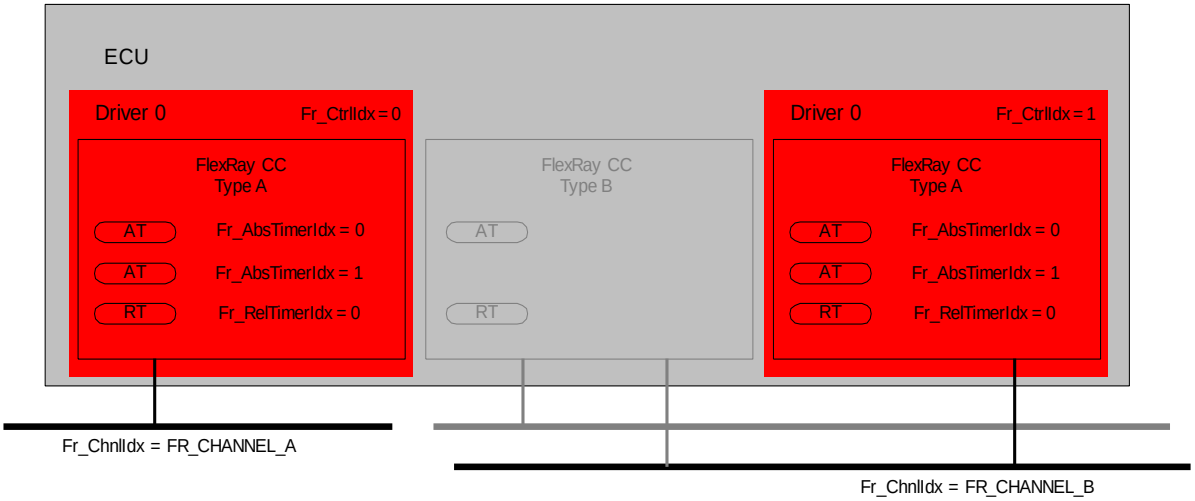

Figure 3 FlexRay Driver indexing scheme

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.

- **FR344:** 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}.

- **FR345:** 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. Absolute timer indices are valid within a tuple <`Fr_CtrlIdx`, `Fr_AbsTimerIdx`> only.

- **FR346:** 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. Relative timer indices are valid within a tuple <`Fr_CtrlIdx`, `Fr_RelTimerIdx`> only.

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 module (i.e., the FlexRay CCs of type A in the example given) are being assigned indices within the context of this Fr module. All other CCs (e.g., the FlexRay CC of type B) are not handled by this Fr module and thus no indices have been assigned to these FlexRay CCs within the context of this Fr module.

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 [12] for details). The CHI commands driving the POC state machine are incorporated into several Fr module API functions.

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

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

FR438: All other API functions than listed above shall not change the POC state of the FlexRay CC.

Figure 4 shows the POC states of the FlexRay CC and the transitions applicable to the Fr module 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 Fr module API (not drawn in Figure 4 – compare to [5]).

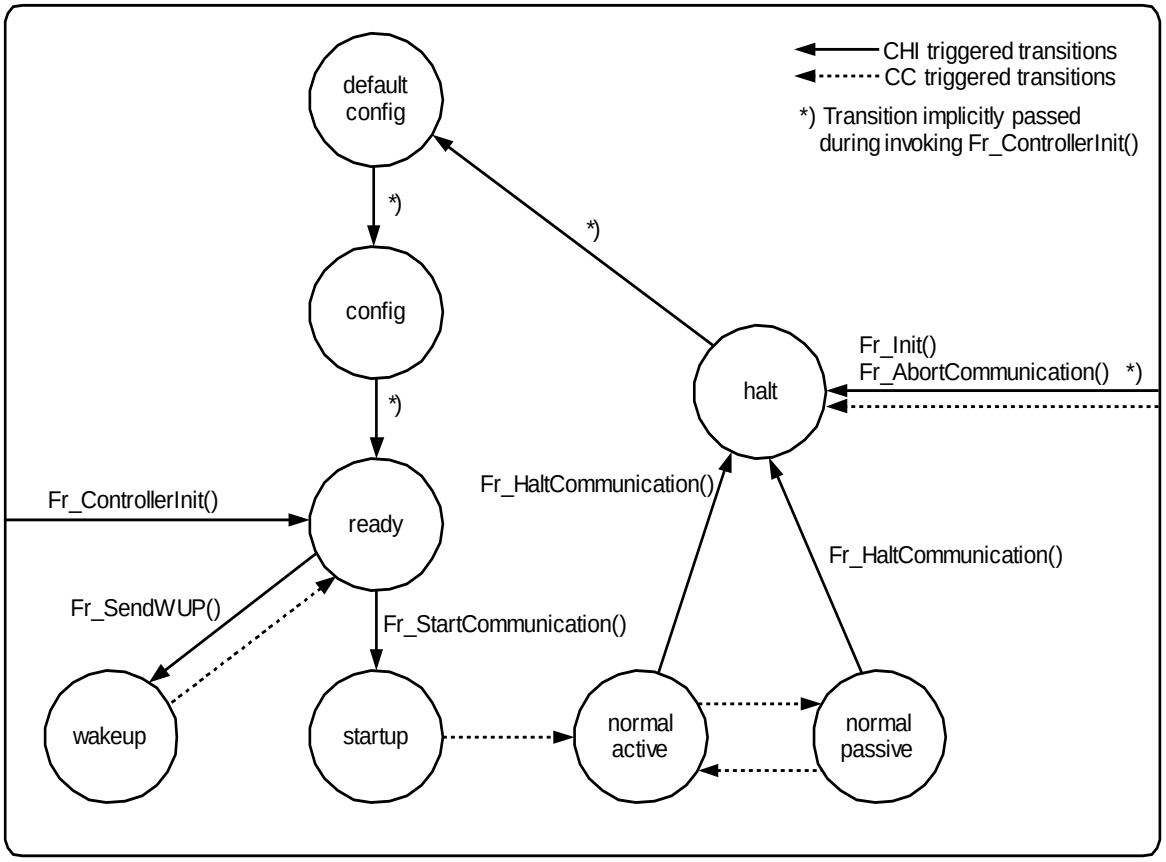

Figure 4 FlexRay Driver POC state machine control

7.4 FIFO support and message ID filtering

To efficiently support reception in certain use-cases, FlexRay controllers might support receive-FIFOs. The receive-FIFOs accept FlexRay frames based on a set of configured filter criterias which match FlexRay specific properties such as frameID, cycle, channel, as well as protocol add-ons like the message ID, in hardware.

FR593: The hardware receive-FIFO shall be used if the FIFO filter-criterias as configured can be applied to the hardware FIFO.

FR594: All LPdus (as configured within FrIf) matching a receive-FIFO's filter-criteria shall be assigned to the respective receive-FIFO.

FR595: No specific buffers shall be assigned to LPdus that are assigned to a receive-FIFO.

FR596: If Fr_ReceiveRxLPdu() is called for an LPdu assigned to the receive FIFO, the service Fr_ReceiveRxPdu() consumes the first valid frame out of the respective FIFO and returns it as received frame. There is no receive-FIFO specific API, thus keeping the upper layers unaffected.

Hint: This restricted implementation of the receive-FIFO covers a very typical use-case (FrTp):

- All received (L)Pdus assigned to the FIFO shall be processed by a single upper layer module.
- The upper layer does not care about the specific assignment of (L)Pdus to FlexRay FrameTriggerings.

FR597: LPdus received via the FIFO shall be returned in the same order as they were received on the FlexRay network.

7.5 Implementation Requirements

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

FR030: The header file *Fr.h* shall include a software and specification version number.

FR031: The Fr module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files.

FR029: In case development error detection is enabled for the Fr module: the Fr module shall check API parameters for validity and report detected errors to the DET.

See chapter 8 for a detailed DET specification for each API function.

FR073: The Fr module's implementation shall conform to the HIS subset of the MISRA C Standard (see document [14]).

FR076: The Fr module's implementer shall replace all prefixes `Fr` within the Fr specification by a vendor specific prefix `Fr_<Vendor Id>_<Vendor specific name>` for implementation to allow the usage of different FlexRay Drivers within one build system. The Fr module's implementer shall apply this rule 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 Fr module shall implement the API functions specified by the Fr SWS as real C-code functions and shall not implement the API functions as macros.

The rationale of FR097 is to allow object code module integration.

FR102: None of the Fr module's header files shall define global variables.

FR106: The Fr module and/or the underlying hardware shall stop FlexRay timers in case of loss of synchronization.

The implementation may assume that

- **FR104:** The Fr module's environment shall call all LPdu-based services (`Fr_TransmitTxLPdu()`, `Fr_ReceiveRxLPdu()`, `Fr_CheckLPduTxStatus()`, `Fr_PrepareLPdu()`) synchronously to the FlexRay global time (at predefined determined points in time) in case of proper system operation.
- **FR105:** The Fr module's environment may call all non LPdu-based services at any time independently from the FlexRay global time.

FR103: If buffer reconfiguration is disabled (configuration parameter FR_BUFFER_RECONFIG is OFF), buffers must not be configured by any other Fr API function than Fr_ControllerInit.

7.6 Configuration description

FR080: The Fr module shall provide an XML file that contains the data, which is required for the SW identification (it shall contain the vendor identification, module ID and software version information), configuration and integration process. This file should describe vendor specific configuration parameters as well as it should contain recommended configuration parameter values.

A driver MCG reads the ECU configuration description of the Fr and the FrIf module(s). While cluster related configuration parameters are contained in the FrIf module's configuration description, CC related configuration data is contained in the Fr module's configuration description. The Fr module's specfic configuration tool shall read both ECU module descriptions to derive the configuration data for all FlexRay CCs mapped to the Fr module.

All frame transmission/reception related configuration is located in the FrIf module description only (within configuration container 'FrIf/FrIfCluster/FrIfFrameTriggering'). 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 MCG 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 module's MCG may decide to share one buffer for several FlexRay frames within the static segment.

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

The number of FlexRay CCs supported is defined at configuration time.

FR004: The Fr module shall be able to address 4 FlexRay CCs if available in hardware.

The MCG should ensure the consistency of the generated configuration data.

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

FR027: The Fr module 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 8.4.1).

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. Additionally the configuration description of the FrIf (see chapter 10 of [7]) shall be evaluated for Fr module configuration.

7.7 Error classification

FR124: Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file `Dem_IntErrId.h` and included via `Dem.h`.

FR125: Development error values are of type uint8.

FR025: 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
Verify FlexRay parameters event id	Production	FR_E_CTRLTESTRESULT	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.

7.8 Error detection

FR026: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch FrDevErrorDetect (see chapter 10) shall activate or deactivate the detection of all development errors.

FR126: The detection of production code errors cannot be switched off.

7.9 Error notification

FR127: Detected development errors shall be reported to the Det_ReportError service of the Development Error Tracer (DET) if the pre-processor switch FrDevErrorDetect is set (see chapter 10).

FR028: The status of EventIds shall be reported to the Diagnostic Event Manager (see chapter 8.7).

8 API specification

FR098: All AP 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

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

FR099:

Module	*Imported Type*
Dem	Dem_EventIdType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Macro definitions

FR657: The macros listed in this chapter shall be defined in Fr_GeneralTypes.h.

8.2.1 Configuration parameter index macros

The following table lists macros which list symbolic names that can be passed into API function Fr_ReadCCConfig as parameter Fr_ConfigParamIdx (see chapter 8.4.37).

Each macro (index) uniquely identifies a configuration parameter which value can be read out of the controller's configuration using Fr_ReadCCConfig.

Macro name	*Value*	*Mapps to configuration parameter*
FR_CIDX_GDCYCLE	0	FrIfGdCycle
FR_CIDX_PMICROPERCYCLE	1	FrPMicroPerCycle
FR_CIDX_PDLISTENTIMEOUT	2	FrPdListenTimeout
FR_CIDX_GMACROPERCYCLE	3	FrIfGMacroPerCycle
FR_CIDX_GDMACROTICK	4	FrIfGdMacrotick
FR_CIDX_GNUMBEROFMINISLOTS	5	FrIfGNumberOfMinislots
FR_CIDX_GNUMBEROFSTATICSLOTS	6	FrIfGNumberOfStaticSlots
FR_CIDX_GDNIT	7	FrIfGdNit
FR_CIDX_GDSTATICSLOT	8	FrIfGdStaticSlot
FR_CIDX_GDWAKEUPRXWINDOW	9	FrIfGdWakeupRxWindow
FR_CIDX_PKEYSLOTID	10	FrPKeySlotId
FR_CIDX_PLATESTTX	11	FrPLatestTx
FR_CIDX_POFFSETCORRECTIONOUT	12	FrPOffsetCorrectionOut
FR_CIDX_POFFSETCORRECTIONSTART	13	FrPOffsetCorrectionStart
FR_CIDX_PRATECORRECTIONOUT	14	FrPRateCorrectionOut
FR_CIDX_PSECONDKEYSLOTID	15	FrPSecondKeySlotId
FR_CIDX_PDACCEPTEDSTARTUPRANGE	16	FrPdAcceptedStartupRange
FR_CIDX_GCOLDSTARTATTEMPTS	17	FrIfGColdStartAttempts

FR_CIDX_GCYCLECOUNTMAX	18	FrIfGCycleCountMax
FR_CIDX_GLISTENNOISE	19	FrIfGListenNoise
FR_CIDX_GMAXWITHOUTCLOCKCORRECTFATAL	20	FrIfGMaxWithoutClockCorrectFatal
FR_CIDX_GMAXWITHOUTCLOCKCORRECTPASSIVE	21	FrIfGMaxWithoutClockCorrectPassive
FR_CIDX_GNETWORKMANAGEMENTVECTORLENGTH	22	FrIfGNetworkManagementVectorLength
FR_CIDX_GPAYLOADLENGTHSTATIC	23	FrIfGPayloadLengthStatic
FR_CIDX_GSYNCFRAMEIDCOUNTMAX	24	FrIfGSyncFrameIDCountMax
FR_CIDX_GDACTIONPOINTOFFSET	25	FrIfGdActionPointOffset
FR_CIDX_GDBIT	26	FrIfGdBit
FR_CIDX_GDCASRXLOWMAX	27	FrIfGdCasRxLowMax
FR_CIDX_GDDYNAMICSLOTIDLEPHASE	28	FrIfGdDynamicSlotIdlePhase
FR_CIDX_GDMINISLOTACTIONPOINTOFFSET	29	FrIfGdMiniSlotActionPointOffset
FR_CIDX_GDMINISLOT	30	FrIfGdMinislot
FR_CIDX_GDSAMPLECLOCKPERIOD	31	FrIfGdSampleClockPeriod
FR_CIDX_GDSYMBOLWINDOW	32	FrIfGdSymbolWindow
FR_CIDX_GDSYMBOLWINDOWACTIONPOINTOFFSET	33	FrIfGdSymbolWindowActionPointOffset
FR_CIDX_GDTSSTRANSMITTER	34	FrIfGdTssTransmitter
FR_CIDX_GDWAKEUPRXIDLE	35	FrIfGdWakeupRxIdle
FR_CIDX_GDWAKEUPRXLOW	36	FrIfGdWakeupRxLow
FR_CIDX_GDWAKEUPTXACTIVE	37	FrIfGdWakeupTxActive
FR_CIDX_GDWAKEUPTXIDLE	38	FrIfGdWakeupTxIdle
FR_CIDX_PALLOWPASSIVETOACTIVE	39	FrPAllowPassiveToActive
FR_CIDX_PCHANNELS	40	FrPChannels
FR_CIDX_PCLUSTERDRIFTDAMPING	41	FrPClusterDriftDamping
FR_CIDX_PDECODINGCORRECTION	42	FrPDecodingCorrection
FR_CIDX_PDELAYCOMPENSATIONA	43	FrPDelayCompensationA
FR_CIDX_PDELAYCOMPENSATIONB	44	FrPDelayCompensationB
FR_CIDX_PMACROINITIALOFFSETA	45	FrPMacroInitialOffsetA
FR_CIDX_PMACROINITIALOFFSETB	46	FrPMacroInitialOffsetB
FR_CIDX_PMICROINITIALOFFSETA	47	FrPMicroInitialOffsetA
FR_CIDX_PMICROINITIALOFFSETB	48	FrPMicroInitialOffsetB
FR_CIDX_PPAYLOADLENGTHDYNMAX	49	FrPPayloadLengthDynMax
FR_CIDX_PSAMPLESPERMICROTICK	50	FrPSamplesPerMicrotick
FR_CIDX_PWAKEUPCHANNEL	51	FrPWakeupChannel
FR_CIDX_PWAKEUPPATTERN	52	FrPWakeupPattern
FR_CIDX_PDMICROTICK	53	FrPdMicrotick
FR_CIDX_PDEXTERNRATECORRECTION	54	FrPExternRateCorrection
FR_CIDX_PDEXTERNOFFSETCORRECTION	55	FrPExternOffsetCorrection
FR_CIDX_GDIGNOREAFTERTX	56	FrIfGdIgnoreAfterTx
FR_CIDX_PALLOWHALTDUETOCLOCK	57	FrPAllowHaltDueToClock
FR_CIDX_PEXTERNALSYNC	58	FrPExternalSync
FR_CIDX_PFALLBACKINTERNAL	59	FrPFallBackInternal
FR_CIDX_PKEYSLOTONLYENABLED	60	FrPKeySlotOnlyEnabled
FR_CIDX_PKEYSLOTUSEDFORSTARTUP	61	FrPKeySlotUsedForStartup
FR_CIDX_PKEYSLOTUSEDFORSYNC	62	FrPKeySlotUsedForSync
FR_CIDX_PNMVECTOREARLYUPDATE	63	FrPNmVectorEarlyUpdate
FR_CIDX_PTWOKEYSLOTMODE	64	FrPTwoKeySlotMode

8.3 Type definitions

FR110: The content of *FrGeneralTypes.h* consists of types specified within [7], [8] and the following type specifications within this document. *FrGeneralTypes.h* shall 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. For implementation all *FrGeneralTypes.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.

8.3.1 Fr_ConfigType

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

8.3.2 Fr_SyncStateType

Name:	`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.	

8.3.3 Fr_OffsetCorrectionType

Name:	`Fr_OffsetCorrectionType`	
Type:	`Enumeration`	
Range:	`FR_OFFSET_INC (=0)`	Add the predefined external correction value vOffsetCorrection to the CC's offset clock correction process.
	`FR_OFFSET_DEC`	Subtract the predefined external correction value vOffsetCorrection from the CC's offset clock correction process.
	`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().	

8.3.4 Fr_RateCorrectionType

Name:	`Fr_RateCorrectionType`	
Type:	`Enumeration`	
Range:	`FR_RATE_INC (=0)`	Add the predefined external correction value vRateCorrection to the CC's rate clock correction process.
	`FR_RATE_DEC`	Subtract the predefined external correction value vRateCorrection from the CC's rate clock correction process.
	`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().	

8.3.5 Fr_POCStateType

Name:	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].	

8.3.6 Fr_SlotModeType

Name:	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].	

8.3.7 Fr_ErrorModeType

Name:	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].	

8.3.8 Fr_WakeupStatusType

Name:	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].	

8.3.9 Fr_StartupStateType

Name:	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].	

Note: Fr_StartupStateType contains the superset of FlexRay 2.1 and FlexRay 3.0 specification. Thus state FR_STARTUP_EXTERNAL_STARTUP canot be reached on FlexRay 2.1 compliant FlexRay controllers.

8.3.10 Fr_POCStatusType

Name:	Fr_POCStatusType
Type:	Structure

Element:	Fr_POCStateType	State	--
	boolean	Freeze	--
	boolean	CHIHaltRequest	--
	boolean	ColdstartNoise	--
	Fr_SlotModeType	SlotMode	--
	Fr_ErrorModeType	ErrorMode	--
	Fr_WakeupStatusType	WakeupStatus	--
	Fr_StartupStateType	StartupState	--
Description:	This formal definition refers to the description of type T_POCStatus in chapter "2.2.1.3 POC status" of [11].		

8.3.11 Fr_TxLPduStatusType

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

8.3.12 Fr_RxLPduStatusType

Name:	Fr_RxLPduStatusType	
Type:	Enumeration	
Range:	FR_RECEIVED (=0)	LSdu has been received
	FR_NOT_RECEIVED	LSdu has not been received
Description:	These values are used to determine if a LPdu has been received or not.	

8.3.13 Fr_MTSStatusType

Name:	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].	

8.3.14 Fr_ChannelType

Name:	`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.	

8.4 Function definitions

During specification of the API functions the following guidelines were applied:

- The API functions of the Fr module shall have the return type `Std_ReturnType` or `void` (no return code).
- If an API function of the Fr module has the return type `Std_ReturnType`: If the function performs its service successfully, it shall return `E_OK` otherwise `E_NOT_OK`.
- If the Fr module's environment is passing input parameters by a reference, the Fr SWS 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 of the Fr module finish successfully (return `E_OK`), they shall have written all output parameters with valid values.
- If API functions of the Fr module finish erroneously (return `E_NOT_OK`), they shall have written no output parameters. Output parameters shall keep their original values in this case.

8.4.1 Fr_Init

FR032:

Service name:	Fr_Init	
Syntax:	`void Fr_Init( const Fr_ConfigType* Fr_ConfigPtr )`	
Service ID[hex]:	0x1c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Fr_ConfigPtr	Address to an Fr dependant configuration structure that contains all information for operating the Fr subsequently.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initalizes the Fr.	

CC precondition for the function Fr_Init: None.

FR137: The function Fr_Init shall internally store the configuration data address to enable subsequent API calls to access the configuration data.

FR347: The function Fr_Init shall ensure that all FlexRay CCs controlled by the Fr module are left in POCState 'POC:halt'.

FR138: The function Fr_Init shall ensure that no transmission requests are pending.

FR139: The function Fr_Init shall ensure that no reception indications are pending.

FR140: The function Fr_Init shall ensure that no interrupts are pending.

FR141: The function Fr_Init shall ensure that all timers are disabled.

FR142: The function Fr_Init shall ensure that all interrupts are disabled.

FR136: If the function Fr_Init detects errors while accessing any CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return.

FR135: If development error detection for the Fr module is enabled: the function Fr_Init shall check the parameter Fr_ConfigPtr for not being a NULL pointer (NULL_PTR). If Fr_ConfigPtr is a NULL pointer, the function Fr_Init shall raise development error FR_E_INV_POINTER and return.

8.4.2 Fr_ControllerInit

FR017:

Service name:	Fr_ControllerInit	

Syntax:	`Std_ReturnType Fr_ControllerInit(`
	`  uint8 Fr_CtrlIdx,`
	`  uint8 Fr_LowLevelConfSetIdx,`
	`  uint8 Fr_BufConfSetIdx`
	`)`	

Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	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 (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Initialzes a FlexRay CC.	

CC precondition for the function Fr_ControllerInit: None

FR148: The function Fr_ControllerInit 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) according to the frame triggering configuration contained in the FrIf.
4. Switch CC into ‘POC:ready’
5. Return E_OK.

CC post condition for the function Fr_ControllerInit: CC Fr_CtrlIdx shall be left in POCState ‘POC:ready’.

FR149: The function Fr_ControllerInit shall ensure that no transmission requests are pending.

FR150: The function Fr_ControllerInit shall ensure that no reception indications are pending.

FR151: The function Fr_ControllerInit shall ensure that no interrupts are pending.

FR152: The function Fr_ControllerInit shall ensure that all timers are disabled.

FR153: The function Fr_ControllerInit shall ensure that all interrupts are disabled.

FR147: If the function Fr_ControllerInit detects errors while accessing any CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR649: If the function Fr_ControllerInit detects errors while testing the CC (CC test), then it calls Dem_ReportErrorStatus (FR_E_CTRLTESTRESULT, DEM_EVENT_STATUS_FAILED) and returns E_NOT_OK (only if configuration parameter FrCCReadbackSupport is true).

FR647: The CC test as described in FR649 shall verify (read back and compare to reference values held in the configuration) that the node and cluster FlexRay parameters were written properly into the FlexRay CC (only if configuration parameter FrCCReadbackSupport is true).

FR598: If the function Fr_ControllerInit passes the CC test, then it calls Dem_ReportErrorStatus (*FrDemCtrlTestResultRef*, DEM_EVENT_STATUS_PASSED) (only if configuration parameter FrCCReadbackSupport is true)..

FR143: If development error detection for the Fr module is enabled: if the function Fr_ControllerInit is called before the Fr was initialized successfully, the function

Fr_ControllerInit shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR144: If development error detection for the Fr module is enabled: the function Fr_ControllerInit shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_ControllerInit shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR145: If development error detection for the Fr module is enabled: the function Fr_ControllerInit shall check the parameter Fr_LowLevelConfSetIdx for being valid. If Fr_LowLevelConfSetIdx is invalid, the function Fr_ControllerInit shall raise the development error FR_E_INV_CONFIG and return E_NOT_OK.

FR146: If development error detection for the Fr module is enabled: the function Fr_ControllerInit shall check the parameter Fr_BufConfSetIdx for being valid. If Fr_BufConfSetIdx is invalid, the function Fr_ControllerInit shall raise the development error FR_E_INV_CONFIG and return E_NOT_OK.

8.4.3 Fr_SendMTS

FR023:

Service name:	Fr_SendMTS	

Syntax:	`Std_ReturnType Fr_SendMTS(`
	`  uint8 Fr_CtrlIdx,`
	`  Fr_ChannelType Fr_ChnlIdx`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_ChnlIdx	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx. Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Triggers a MTS.	

FR159: The Fr module's environment shall only call the function Fr_SendMTS when the CC Fr_CtrlIdx is synchronized to FlexRay global time (=CC precondition for the function Fr_SendMTS)

FR161: The function Fr_SendMTS 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.

Since the FlexRay Protocol Specification [12] 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 the environment must behave according to FR349.

FR349: The Fr module's environment shall call the function Fr_SendMTS periodically as long as MTS symbols shall be transmitted followed by a single final Fr_StopMTS() API function call.

FR160: If the function Fr_SendMTS detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR154: If development error detection for the Fr module is enabled: if the function Fr_SendMTS is called before the Fr module was initialized successfully, the function Fr_SendMTS shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR155: If development error detection for the Fr module is enabled: the function Fr_SendMTS shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_SendMTS shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR156: If development error detection for the Fr module is enabled: the function Fr_SendMTS shall check the parameter Fr_ChnlIdx for being valid. If Fr_ChnlIdx is invalid, the function Fr_SendMTS shall raise the development error FR_E_INV_CHNL_IDX and return E_NOT_OK.

FR157: If development error detection for the Fr module is enabled: the function Fr_SendMTS shall check the CC Fr_CtrlIdx for being synchronized to the FlexRay global time. If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time, the function Fr_SendMTS shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

FR158: If development error detection for the Fr module is enabled: the function Fr_SendMTS shall check if the currently configured MTS duration doest fit within the currently configured symbol window duration. If it does not fit, the function Fr_SendMTS shall shall raise the development error FR_E_INV_CONFIG and return E_NOT_OK.

8.4.4 Fr_StopMTS

FR090:

Service name:	Fr_StopMTS

Syntax:	`Std_ReturnType Fr_StopMTS(`
	`  uint8 Fr_CtrlIdx,`
	`  Fr_ChannelType Fr_ChnlIdx`
	`)`

Service ID[hex]:	0x1d

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_ChnlIdx	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx. Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Stops the periodic transmission of MTS symbols.	

CC precondition for the function Fr_StopMTS: None.

FR166: 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.

Since the FlexRay Protocol Specification [12] 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 the environment must behave according to FR349.

FR165: If the function Fr_StopMTS detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR162: If development error detection for the Fr module is enabled: if the function Fr_StopMTS is called before the Fr module was initialized successfully, the function Fr_StopMTS shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR163: If development error detection for the Fr module is enabled: the function Fr_StopMTS shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_StopMTS shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR164: If development error detection for the Fr module is enabled: the function Fr_StopMTS shall check the parameter Fr_ChnlIdx for being valid. If Fr_ChnlIdx is invalid, the function Fr_StopMTS shall raise the development error FR_E_INV_CHNL_IDX and return E_NOT_OK.

8.4.5 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[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_ChnlIdx	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx. Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (inout):	None	
Parameters (out):	Fr_MTSStatusPtr	Address the output value is stored to.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Checks the MTS.	

CC precondition for the function Fr_CheckMTS: None.

FR172: The function Fr_CheckMTS 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.

FR171: If the function Fr_CheckMTS detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR167: If development error detection for the Fr module is enabled: if the function Fr_CheckMTS is called before the Fr was initialized successfully, the function Fr_CheckMTS shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR168: If development error detection for the Fr module is enabled: the function Fr_CheckMTS shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_CheckMTS shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR169: If development error detection for the Fr module is enabled: the function Fr_CheckMTS shall check the parameter Fr_ChnlIdx for being valid. If Fr_ChnlIdx is invalid, the function Fr_CheckMTS shall raise the development error FR_E_INV_CHNL_IDX and return E_NOT_OK.

FR170: If development error detection for the Fr module is enabled: the function Fr_CheckMTS shall check the parameter Fr_MTSStatusPtr for not being a NULL pointer (NULL_PTR). If Fr_MTSStatusPtr is a NULL pointer, the function

Fr_CheckMTS shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.6 Fr_StartCommunication

FR010:

Service name:	Fr_StartCommunication	
Syntax:	`Std_ReturnType Fr_StartCommunication( uint8 Fr_CtrlIdx )`	
Service ID[hex]:	0x03	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Starts communication.	

FR352: The Fr module's environment shall only call the function Fr_StartCommunucation when CC Fr_CtrlIdx is in POCState 'POC:ready' (=CC precondition for the function Fr_StartCommunication).

FR177: The function Fr_StartCommunication shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Invoke the CC CHI command 'RUN, which initiates the startup procedure within the FlexRay CC.
2. Return E_OK.

The function call of Fr_StartCommunication 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. It is not guaranteed that the FlexRay CC resides in the 'POC:normal active' or 'POC:normal passive' state after a call of the function Fr_StartCommunication.

CC post condition for the function Fr_StartCommunication: The FlexRay CC is either in POCState 'POC:normal-active', 'POC:normal-passive' or 'POC:halt'.

FR176: If the function Fr_StartCommunication detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR173: If development error detection for the Fr module is enabled: if the function Fr_StartCommunication is called before the Fr was initialized successfully, the function Fr_StartCommunication shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR174: If development error detection for the Fr module is enabled: the function Fr_StartCommunication shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_StartCommunication shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR175: If development error detection for the Fr module is enabled: the function Fr_StartCommunication shall check the CC Fr_CtrlIdx's POCState for being POC:ready. If the POCState is not POC:ready, the function Fr_StartCommunication shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.7 Fr_AllowColdstart

FR114:

Service name:	Fr_AllowColdstart	

Syntax:	`Std_ReturnType Fr_AllowColdstart(`
	`  uint8 Fr_CtrlIdx`
	`)`	

Service ID[hex]:	0x23	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Invokes the CC CHI command 'ALLOW_COLDSTART'.	

FR353: The Fr Module's environment shall only call the function Fr_AllowColdstart when the CC Fr_CtrlIdx is in any POCState except 'POC:default config, POC:config or POC:halt' (=CC precondition for the function Fr_AllowColdstart)

FR182: The function Fr_AllowColdstart shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Invoke the CC CHI command 'ALLOW_COLDSTART'.
2. Return E_OK.

FR181: If the function Fr_AllowColdstart detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR178: If development error detection for the Fr module is enabled: if the function Fr_AllowColdstart is called before the Fr was initialized successfully, the function Fr_AllowColdstart shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR179: If development error detection for the Fr module is enabled: the function Fr_AllowColdstart shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_AllowColdstart shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR180: If development error detection for the Fr module is enabled: the function Fr_AllowColdstart shall check the CC Fr_CtrlIdx's POCState. If the POCState is POC:default config, POC:config or POC:halt the function Fr_AllowColdstart shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.8 Fr_HaltCommunication

FR014:

Service name:	Fr_HaltCommunication	
Syntax:	`Std_ReturnType Fr_HaltCommunication( uint8 Fr_CtrlIdx )`	
Service ID[hex]:	0x04	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Invokes the CC CHI command 'HALT'.	

FR355: The Fr module's environment shall only call the function Fr_HaltCommunication when CC Fr_CtrlIdx is synchronized to the FlexRay global time (=CC precondition for the function Fr_HaltCommunication).

FR187: The function Fr_HaltCommunication shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Invoke the CC CHI command 'DEFERRED_HALT[1].
2. Return E_OK.

FR356: The function Fr_HaltCommunication requests the halt state which shall be reached by the end of the current FlexRay communication cycle but might not be reached immediately (=CC post condition for the function Fr_HaltCommunication)

FR186: If the function Fr_HaltCommunication detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR183: If development error detection for the Fr module is enabled: if the function Fr_HaltCommunication is called before the Fr was initialized successfully, the function Fr_HaltCommunication shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR184: If development error detection for the Fr module is enabled: the function Fr_HaltCommunication shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_HaltCommunication shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR185: If development error detection for the Fr module is enabled: the function Fr_HaltCommunication shall check the CC Fr_CtrlIdx for being synchronized to the FlexRay global time. If the CC Fr_CtrlIdx is not synchronized to the FlexRay global

[1] Invoke the command 'HALT' for FlexRay Controllers compliant to [13].

time, the function Fr_HaltCommunication shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.9 Fr_AbortCommunication

FR011:

Service name:	Fr_AbortCommunication	
Syntax:	`Std_ReturnType Fr_AbortCommunication( uint8 Fr_CtrlIdx )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Invokes the CC CHI command 'FREEZE'.	

CC precondition for the function Fr_AbortCommunication: None.

FR191: The function Fr_AbortCommunication 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.

FR357: The function Fr_AbortCommunication shall leave the CC Fr_CtrlIdx in POCState POC:halt (vPOC!Freeze is set) (=CC post condition for the function Fr_AbortCommunication)

FR190: If the function Fr_AbortCommunication detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR188: If development error detection for the Fr module is enabled: if the function Fr_AbortCommunication is called before the Fr was initialized successfully, the function Fr_AbortCommunication shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR189: If development error detection for the Fr module is enabled: the function Fr_AbortCommunication shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_AbortCommunication shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

8.4.10 Fr_SendWUP

FR009:

Service name:	Fr_SendWUP	
Syntax:	`Std_ReturnType Fr_SendWUP( uint8 Fr_CtrlIdx )`	
Service ID[hex]:	0x06	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Invokes the CC CHI command 'WAKEUP'.	

FR358: The Fr module's environment shall only call Fr_SendWUP when CC Fr_CtrlIdx is in POCState 'POC:ready'. (=CC precondition for the function Fr_SendWUP)

FR196: The function Fr_SendWUP 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.

FR359: The function Fr_SendWUP shall change the CC Fr_CtrlIdx POCState to POC:wakeup which is a transitional state. After performing the wakeup procedure the CC will reach POC:ready again (=CC post condition for the function Fr_SendWUP)

Note: Sending a wakeup pattern does not necessarily cause all cluster nodes to be awoken afterwards. The function Fr_SendWUP just invokes the wakeup symbol transmission procedure on a certain FlexRay CC.

FR195: If the function Fr_SendWUP detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR192: If development error detection for the Fr module is enabled: if the function Fr_SendWUP is called before the Fr was initialized successfully, the function Fr_SendWUP shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR193: If development error detection for the Fr module is enabled: the function Fr_SendWUP shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_SendWUP shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR194: If development error detection for the Fr module is enabled: the function Fr_SendWUP shall check the CC Fr_CtrlIdx's POCState for being POC:ready. If the POCState is not POC:ready, the function Fr_SendWUP shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.11 Fr_SetWakeupChannel

FR091:

Service name:	Fr_SetWakeupChannel	
Syntax:	`Std_ReturnType Fr_SetWakeupChannel( uint8 Fr_CtrlIdx, Fr_ChannelType Fr_ChnlIdx )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_ChnlIdx	Index of FlexRay channel within the context of the FlexRay CC Fr_CtrlIdx. Valid values are FR_CHANNEL_A and FR_CHANNEL_B.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Sets a wakeup channel.	

FR360: The Fr module's environment shall only call the function Fr_SetWakupChannel when the CC Fr_CtrlIdx is in POCState 'POC:ready' (=CC precondition for the function Fr_SetWakeupChannel)

FR202: The function Fr_SetWakeupChannel 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.

FR201: If the function Fr_SetWakeupChannel detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR197: If development error detection for the Fr module is enabled: if the function Fr_SetWakeupChannel is called before the Fr was initialized successfully, the function Fr_SetWakeupChannel shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR198: If development error detection for the Fr module is enabled: the function Fr_SetWakeupChannel shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_SetWakeupChannel shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR199: If development error detection for the Fr module is enabled: the function Fr_SetWakeupChannel shall check the parameter Fr_ChnlIdx for being valid. If Fr_ChnlIdx is invalid, the function Fr_SetWakeupChannel shall raise the development error FR_E_INV_CHNL_IDX and return E_NOT_OK.

FR200: If development error detection for the Fr module is enabled: the function Fr_SetWakeupChannel shall check the CC Fr_CtrlIdx's POCState for being POC:ready. If the POCState is not 'POC:ready', the function Fr_SetWakeupChannel shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.12 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[hex]:	0x08	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_Offset	Determines the kind of offset correction.
	Fr_Rate	Determines the kind of rate correction.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Adjusts the global time of a FlexRay CC to an external clock source.	

The function Fr_SetExtSync 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. This external clock correction value is only applied for one communication cycle and not repetitively!

FR362: The Fr module's environment shall only call Fr_SetExtSync when the CC Fr_CtrlIdx is synchronized to FlexRay global time (=CC precondition for the function Fr_SetExtSync)

FR207: The function Fr_SetExtSync shall perform the following tasks on FlexRay CC Fr_CtrlIdx:

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.

FR206: If the function Fr_SetExtSync detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR203: If development error detection for the Fr module is enabled: if the function Fr_SetExtSync is called before the Fr was initialized successfully, the function Fr_SetExtSync shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR204: If development error detection for the Fr module is enabled: the function Fr_SetExtSync shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_SetExtSync shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR205: If development error detection for the Fr module is enabled: the function Fr_SetExtSync shall check the CC Fr_CtrlIdx for being synchronized to the FlexRay global time. If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time, the function Fr_SetExtSync shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.13 Fr_GetSyncState

FR021:

Service name:	Fr_GetSyncState	
Syntax:	`Std_ReturnType Fr_GetSyncState( uint8 Fr_CtrlIdx, Fr_SyncStateType* Fr_SyncStatePtr )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	Fr_SyncStatePtr	Address the output value is stored to.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the sync state.	

CC precondition for the function Fr_GetSyncState: None.

FR212: The function Fr_GetSyncState 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.

FR211: If the function Fr_GetSyncState detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR208: If development error detection for the Fr module is enabled: if the function Fr_GetSyncState is called before the Fr was initialized successfully, the function Fr_GetSyncState shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR209: If development error detection for the Fr module is enabled: the function Fr_GetSyncState shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_GetSyncState shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR210: If development error detection for the Fr module is enabled: the function Fr_GetSyncState shall check the parameter Fr_SyncStatePtr for not being a NULL pointer (NULL_PTR). If Fr_SyncStatePtr is a NULL pointer, the function Fr_GetSyncState shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.14 Fr_GetPOCStatus

FR012:

Service name:	Fr_GetPOCStatus	
Syntax:	`Std_ReturnType Fr_GetPOCStatus( uint8 Fr_CtrlIdx, Fr_POCStatusType* Fr_POCStatusPtr )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	Fr_POCStatusPtr	Address the output value is stored to.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the POC status.	

CC precondition for the function Fr_GetPOCStatus: None.

FR217: The function Fr_GetPOCStatus 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.

FR216: If the function Fr_GetPOCStatus detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR213: If development error detection for the Fr module is enabled: if the function Fr_GetPOCStatus is called before the Fr was initialized successfully, the function Fr_GetPOCStatus shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR214: If development error detection for the Fr module is enabled: the function Fr_GetPOCStatus shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_GetPOCStatus shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR215: If development error detection for the Fr module is enabled: the function Fr_GetPOCStatus shall check the parameter Fr_SyncStatePtr for not being a NULL pointer (NULL_PTR). If Fr_POCStatusPtr is a NULL pointer, the function Fr_GetPOCStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.15 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[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	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 (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Transmits data on the FlexRay network.	

CC precondition for the function Fr_TransmitTxLPdu: None.

FR224: The function Fr_TransmitTxLPdu 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 is true), the transmission resource shall be reconfigured to match payload length Fr_LSduLength passed at the API.
3. Copy Fr_LSduLength bytes from address Fr_LSduPtr into the FlexRay CC's transmission resource and activate it for transmission.
4. Return E_OK.

FR440: If a transmit resource is shared between more than 1 LPdu (using reconfiguration mechanism of Fr_PrepareLPdu), the function Fr_TransmitTxLPdu must ensure that the transmit resource is correctly configured to match the properties of LPduIdx. Return E_NOT_OK and abort the function execution if a wrong configuration is detected.

FR225: The Fr module shall ensure that the payload data is transmitted on the FlexRay network in the same byte order as it was passed in the parameter Fr_LSduPtr of the function Fr_TransmitTxLPdu. (first byte = lowest address, last byte = highest address).

FR223: If the function Fr_TransmitTxLPdu detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR218: If development error detection for the Fr module is enabled: if the function Fr_TransmitTxLPdu is called before the Fr was initialized successfully, the function Fr_TransmitTxLPdu shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR219: If development error detection for the Fr module is enabled: the function Fr_TransmitTxLPdu shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_TransmitTxLPdu shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR220: If development error detection for the Fr module is enabled: the function Fr_TransmitTxLPdu shall check the parameter Fr_LPduIdx for being valid. If Fr_LPduIdx is invalid, the function Fr_TransmitTxLPdu shall raise the development error FR_E_INV_LPDU_IDX and return E_NOT_OK.

FR221: If development error detection for the Fr module is enabled: the function Fr_TransmitTxLPdu shall check the parameter Fr_LSduLength for being valid. If Fr_LSduLength is invalid, the function Fr_TransmitTxLPdu shall raise the development error FR_E_INV_LPDU_IDX and return E_NOT_OK.

FR222: If development error detection for the Fr module is enabled: the function Fr_TransmitTxLPdu shall check the parameter Fr_LSduPtr for not being a NULL pointer (NULL_PTR). If Fr_LSduPtr is a NULL pointer, the function Fr_TransmitTxLPdu shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.16 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[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_LPduIdx	This index is used to uniquely identify a FlexRay frame.
Parameters (inout):	None	
Parameters (out):	Fr_LSduPtr	This reference points to the buffer where the LSdu to be received shall be stored.
	Fr_LPduStatusPtr	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:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Receives data from the FlexRay network.	

CC precondition for the function Fr_ReceiveRxLPdu: None.

FR233: The function Fr_ReceiveRxLPdu 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.

FR441: If a receive resource is shared between more than 1 LPdu (using reconfiguration mechanism of Fr_PrepareLPdu), the function Fr_ReceiveRxLPdu must ensure that the receive resource is correctly configured to match the properties of LPduIdx. Return E_NOT_OK and abort the function execution if a wrong configuration is detected.

FR234: The function Fr_ReceiveRxLPdu shall ensure that the 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).

FR235: The function Fr_ReceiveRxLPdu shall ensure that FR_RECEIVED is returned only for valid frames.

FR236: The function Fr_ReceiveRxLPdu shall ensure that FR_RECEIVED is returned only for non-Nullframes.

FR237: The function Fr_ReceiveRxLPdu shall ensure that the function returns FR_RECEIVED only once per received frame.

FR238: The function Fr_ReceiveRxLPdu shall ensure that only data of the newly arrived frame is copied.

FR239: The function Fr_ReceiveRxLPdu shall ensure that 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.

FR239 enables

- the partly reception of large FlexRay frames (e.g. enables local resource optimizations, support for transparent frame extensions).
- 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].

FR232: If the function Fr_ReceiveRxLPdu detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR226: If development error detection for the Fr module is enabled: if the function Fr_ReceiveRxLPdu is called before the Fr was initialized successfully, the function Fr_ReceiveRxLPdu shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR227: If development error detection for the Fr module is enabled: the function Fr_ReceiveRxLPdu shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_ReceiveRxLPdu shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR228: If development error detection for the Fr module is enabled: the function Fr_ReceiveRxLPdu shall check the parameter Fr_LPduIdx for being valid. If Fr_LPduIdx is invalid, the function Fr_ReceiveRxLPdu shall raise the development error FR_E_INV_LPDU_IDX and return E_NOT_OK.

FR229: If development error detection for the Fr module is enabled: the function Fr_ReceiveRxLPdu shall check the parameter Fr_LSduPtr for not being a NULL

pointer (NULL_PTR). If Fr_LSduPtr is a NULL pointer, the function Fr_ReceiveRxLPdu shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

FR230: If development error detection for the Fr module is enabled: the function Fr_ReceiveRxLPdu shall check the parameter Fr_RxLPduStatusPtr for not being a NULL pointer (NULL_PTR). If Fr_RxLPduStatusPtr is a NULL pointer, the function Fr_ReceiveRxLPdu shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

FR231: If development error detection for the Fr module is enabled: the function Fr_ReceiveRxLPdu shall check the parameter Fr_LSduLengthPtr for not being a NULL pointer (NULL_PTR). If Fr_LSduLengthPtr is a NULL pointer, the function Fr_ReceiveRxLPdu shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.17 Fr_CheckTxLPduStatus

FR094:

Service name:	Fr_CheckTxLPduStatus	
Syntax:	`Std_ReturnType Fr_CheckTxLPduStatus( uint8 Fr_CtrlIdx, uint16 Fr_LPduIdx, Fr_TxLPduStatusType* Fr_TxLSduStatusPtr )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_LPduIdx	This index is used to uniquely identify a FlexRay frame
Parameters (inout):	None	
Parameters (out):	Fr_TxLSduStatusPtr	This reference is used to store the transmit status of the LSdu
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Checks the transmit status of the LSdu.	

CC precondition for the function Fr_CheckTxLPduStatus: None.

FR244: The function Fr_CheckTxLPduStatus shall check, whether a dedicated transmission resource is pending for transmission. Therefore the following steps shall be performed 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 is pending for transmission[2].
3. If the transmission resource is pending for transmission, store the status FR_NOT_TRANSMITTED to Fr_TxLPduStatusPtr.

[2] The returned status does not check whether a transmission has been really performed, but returns whether a transmission resource is empty or not.

4. If the transmission resource is not pending for transmission, store the status FR_TRANSMITTED to Fr_TxLPduStatusPtr.
5. Return E_OK.

FR442: If a transmit resource is shared between more than 1 LPdu (using reconfiguration mechanism of Fr_PrepareLPdu), the function Fr_TransmitTxLPdu must ensure that the transmit resource is correctly configured to match the properties of LPduIdx. Return E_NOT_OK and abort the function execution if a wrong configuration is detected.

FR243: If the function Fr_CheckTxLPduStatus detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR240: If development error detection for the Fr module is enabled: if the function Fr_CheckTxLPduStatus is called before the Fr was initialized successfully, the function Fr_CheckTxLPduStatus shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR241: If development error detection for the Fr module is enabled: the function Fr_CheckTxLPduStatus shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_CheckTxLPduStatus shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR242: If development error detection for the Fr module is enabled: the function Fr_CheckTxLPduStatus shall check the parameter Fr_LPduIdx for being valid. If Fr_LPduIdx is invalid, the function Fr_CheckTxLPduStatus shall raise the development error FR_E_INV_LPDU_IDX and return E_NOT_OK.

FR343: If development error detection for the Fr module is enabled: the function Fr_CheckTxLPduStatus shall check the parameter Fr_TxLPduStatusPtr for not being a NULL pointer (NULL_PTR). If Fr_TxLPduStatusPtr is a NULL pointer, the function Fr_CheckTxLPduStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.18 Fr_PrepareLPdu

FR107:

Service name:	Fr_PrepareLPdu	

Syntax:	`Std_ReturnType Fr_PrepareLPdu(`
	`  uint8 Fr_CtrlIdx,`
	`  uint16 Fr_LPduIdx`
	`)`	

Service ID[hex]:	0x1f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_LPduIdx	This index is used to uniquely identify a FlexRay frame
Parameters	None	

(inout):		
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Prepares a LPdu.	

CC precondition for the function Fr_PrepareLPdu: None.

FR249: The function Fr_PrepareLPdu 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 (a message buffer) appropriate for LPduIdx operation (SlotId, Cycle filter, payload length, header CRC, etc.) if the MCG decided to use the reconfiguration feature.
3. Return E_OK.

FR250: The function Fr_PrepareLPdu shall be pre compile time configurable On/Off by the configuration parameter: FR_RECONFIG_BUFFER.

FR248: If the function Fr_PrepareLPdu detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR245: If development error detection for the Fr module is enabled: if the function Fr_PrepareLPdu is called before the Fr was initialized successfully, the function Fr_PrepareLPdu shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR246: If development error detection for the Fr module is enabled: the function Fr_PrepareLPdu shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_PrepareLPdu shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR247: If development error detection for the Fr module is enabled: the function Fr_PrepareLPdu shall check the parameter Fr_LPduIdx for being valid. If Fr_LPduIdx is invalid, the function Fr_PrepareLPdu shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

8.4.19 Fr_GetGlobalTime

FR042:

Service name:	Fr_GetGlobalTime

Syntax:	`Std_ReturnType Fr_GetGlobalTime(`
	`  uint8 Fr_CtrlIdx,`
	`  uint8* Fr_CyclePtr,`
	`  uint16* Fr_MacroTickPtr`
	`)`

Service ID[hex]:	0x10

Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
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:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the current global FlexRay time.	

FR370: The Fr module's environment shall only call Fr_GetGloalTime if the CC Fr_CtrlIdx is synchronized to FlexRay global time (=CC precondition for the function Fr_GetGlobalTime)

FR256: The function Fr_GetGlobalTime shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Read the current global FlexRay time and write it to the output parameters Fr_CyclePtr and Fr_MacrotickPtr.
2. Return E_OK.

FR257: The function Fr_GetGlobalTime shall ensure that the time information is consistent and valid.

FR044: The function Fr_GetGlobalTime shall ensure that the time information is valid and up to date (synchronized CC) – otherwise the ouput parameters shall not be written and E_NOT_OK returned.

FR255: If the function Fr_GetGlobalTime detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR251: If development error detection for the Fr module is enabled: if the function Fr_GetGlobalTime is called before the Fr was initialized successfully, the function Fr_GetGlobalTime shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR252: If development error detection for the Fr module is enabled: the function Fr_GetGlobalTime shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_GetGlobalTime shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR253: If development error detection for the Fr module is enabled: the function Fr_GetGlobalTime shall check the parameter Fr_CyclePtr for not being a NULL pointer (NULL_PTR). If Fr_CyclePtr is a NULL pointer, the function Fr_GetGlobalTime shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

FR254: If development error detection for the Fr module is enabled: the function Fr_GetGlobalTime shall check the parameter Fr_MacroTickPtr for not being a NULL pointer (NULL_PTR). If Fr_MacroTickPtr is a NULL pointer, the function Fr_GetGlobalTime shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.20 Fr_GetNmVector

FR113:

Service name:	Fr_GetNmVector	
Syntax:	`Std_ReturnType Fr_GetNmVector( uint8 Fr_CtrlIdx, uint8* Fr_NmVectorPtr )`	
Service ID[hex]:	0x22	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	Fr_NmVectorPtr	Address where the NmVector of the last communication cycle shall be stored.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the network management vector of the last communication cycle.	

FR372: The Fr module's environment shall only call the function Fr_GetNmVector when the CC Fr_CtrlIdx is synchronized to FlexRay global time (=precondition for the function Fr_GetNmVector)

FR262: The function Fr_GetNmVector shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Read the network management vector of the last communication cycle and write it to the output parameter Fr_NmVectorPtr. The number of bytes written to the output parameter is constant and known at configuration time (FrIf configuration parameter equivalent to FlexRay Protocol configuration parameter 'gNetworkManagementVectorLength').
2. Return E_OK.

FR263: The function Fr_GetNmVector shall ensure that the FlexRay CC is synchronized to global time – otherwise the ouput parameters shall not be written and E_NOT_OK returned.

FR264: The function Fr_GetNmVector shall ensure that the payload data is copied to Fr_NmVectorPtr in the same byte order as it was received on the FlexRay bus. (first byte = lowest address, last byte = highest address).

FR265: The function Fr_GetNmVector shall ensure that the network management vector data is consistent and valid.

FR266: The function Fr_GetNmVector shall be pre compile time configurable On/Off by the configuration parameter: FR_NMVECTOR_ENABLE.

FR261: If the function Fr_GetNmVector detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR258: If development error detection for the Fr module is enabled: if the function Fr_GetNmVector is called before the Fr was initialized successfully, the function Fr_GetNmVector shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR259: If development error detection for the Fr module is enabled: the function Fr_GetNmVector shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_GetNmVector shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR260: If development error detection for the Fr module is enabled: the function Fr_GetNmVector shall check the parameter Fr_NmVectorPtr for not being a NULL pointer (NULL_PTR). If Fr_NmVectorPtr is a NULL pointer, the function Fr_GetNmVector shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.21 Fr_GetChannelStatus

FR556:

Service name:	Fr_GetChannelStatus	
Syntax:	`Std_ReturnType Fr_GetChannelStatus( uint8 Fr_CtrlIdx, uint16* Fr_ChannelAStatusPtr, uint16* Fr_ChannelBStatusPtr )`	
Service ID[hex]:	0x28	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	Fr_ChannelAStatusPtr	Address where the bitcoded channel A status information shall be stored.
	Fr_ChannelBStatusPtr	Address where the bitcoded channel B status information shall be stored.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the channel status information.	

Note: The Fr module's environment shall only call Fr_GetChannelStatus if the CC Fr_CtrlIdx is synchronous to FlexRay global time.

FR558: The function Fr_GetChannelStatus shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Read the aggregated channel status, NIT status, symbol window status and write it to the output parameter Fr_ChannelAStatusPtr/ Fr_ChannelBStatusPtr. The value of *Fr_ChannelAStatusPtr/*Fr_ChannelBStatusPtr is bitcoded with the following meaning (Bit 0 = LSB, Bit 15 = MSB)[3]:
   - Bit 0: Channel A/B aggregated channel status vSS!ValidFrame
   - Bit 1: Channel A/B aggregated channel status vSS!SyntaxError
   - Bit 2: Channel A/B aggregated channel status vSS!ContentError
   - Bit 3: Channel A/B aggregated channel status additional communication
   - Bit 4: Channel A/B aggregated channel status vSS!Bviolation
   - Bit 5: Channel A/B aggregated channel status vSS!TxConflict
   - Bit 6: Not used (0)
   - Bit 7: Not used (0)
   - Bit 8: Channel A/B symbol window status data vSS!ValidMTS
   - Bit 9: Channel A/B symbol window status data vSS!SyntaxError
   - Bit 10: Channel A/B symbol window status data vSS!Bviolation
   - Bit 11: Channel A/B symbol window status data vSS!TxConflict
   - Bit 12: Channel A/B NIT status data vSS!SyntaxError
   - Bit 13: Channel A/B NIT status data vSS!Bviolation
   - Bit 14: Not used (0)
   - Bit 15: Not used (0)
2. Reset the aggregated channel status information within the FlexRay controller.
3. Return E_OK.

FR559: The function Fr_GetChannelStatus shall ensure that the information is valid and up to date (synchronized CC) – otherwise the output parameters shall not be written and E_NOT_OK returned.

FR560: If the function Fr_GetChannelStatus is able to and detects a hardware error while performing the requested functionality, then it shall call Dem_ReportErrorStatus (*FrDemCtrlTestResultRef*, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR561: If development error detection for the Fr module is enabled, and if the function Fr_GetChannelStatus is called before the successful initialization of Fr, then the function Fr_GetChannelStatus shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR562: If development error detection for the Fr module is enabled, then the function Fr_GetChannelStatus shall check the validity of the parameter Fr_CtrlIdx. If Fr_CtrlIdx is invalid, then the function Fr_GetChannelStatus shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR563: If development error detection for the Fr module is enabled, then the function Fr_GetChannelStatus shall check whether the parameter Fr_ChannelAStatusPtr is a NULL pointer (NULL_PTR). If Fr_ChannelAStatusPtr is a

[3] Bit 5 shall be set to 0 for FlexRay 2.1 compliant controllers, since vSS!TxConflict of channel status is not supported on this hardware.

NULL pointer, then the function Fr_GetChannelStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

FR607: If development error detection for the Fr module is enabled, then the function Fr_GetChannelStatus shall check whether the parameter Fr_ChannelBStatusPtr is a NULL pointer (NULL_PTR). If Fr_ChannelBStatusPtr is a NULL pointer, then the function Fr_GetChannelStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.22 Fr_GetClockCorrection

FR564: [4] [5]

Service name:	Fr_GetClockCorrection	
Syntax:	`Std_ReturnType Fr_GetClockCorrection( uint8 Fr_CtrlIdx, sint16* Fr_RateCorrectionPtr, sint32* Fr_OffsetCorrectionPtr )`	
Service ID[hex]:	0x29	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	Fr_RateCorrectionPtr	Address where the current rate correction value shall be stored.
	Fr_OffsetCorrectionPtr	Address where the current offset correction value shall be stored.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the current clock correction values. See variables vInterimRateCorrection and vInterimOffsetCorrection of [12] for details.	

FR566: The function Fr_GetClockCorrection shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Read the rate correction value (vInterimRateCorrection[4]) and write it as signed integer to the output parameter Fr_RateCorrectionPtr. Read the offset correction value (vInterimOffsetCorrection[5]) and write it as signed integer to the output parameter Fr_OffsetCorrectionPtr
2. Return E_OK.

FR568: If the function Fr_GetClockCorrection is able to and detects a hardware error while performing the requested functionality, then it shall call Dem_ReportErrorStatus (*FrDemCtrlTestResultRef*, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR569: If development error detection for the Fr module is enabled, and if the function Fr_GetClockCorrection is called before the successful initialization of Fr,

[4] vInterimRate Correction maps to vRateCorrection for FlexRay 2.1 compliant controllers, see [13]
[5] vInterimOffsetCorrection maps to vOffsetCorrection for FlexRay 2.1 compliant controllers, see [13]

then the function Fr_GetClockCorrection shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR570: If development error detection for the Fr module is enabled, then the function Fr_GetClockCorrection shall check the validity of the parameter Fr_CtrlIdx. If Fr_CtrlIdx is invalid, then the function Fr_GetClockCorrection shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR571: If development error detection for the Fr module is enabled, then the function Fr_GetClockCorrection shall check whether the parameter Fr_RateCorrectionPtr is a NULL pointer (NULL_PTR). If Fr_RateCorrectionPtr is a NULL pointer, then the function Fr_GetClockCorrection shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

FR572: If development error detection for the Fr module is enabled, then the function Fr_GetClockCorrection shall check whether the parameter Fr_OffsetCorrectionPtr is a NULL pointer (NULL_PTR). If Fr_OffsetCorrectionPtr is a NULL pointer, then the function Fr_GetClockCorrection shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.23 Fr_GetWakeupRxStatus
[FR670]

Service name:	Fr_GetWakeupRxStatus	

Syntax:	`Std_ReturnType Fr_GetWakeupRxStatus(`
	`  uint8 Fr_CtrlIdx,`
	`  uint8* Fr_WakeupRxStatusPtr`
	`)`	

Service ID[hex]:	0x2b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
Parameters (inout):	None	
Parameters (out):	Fr_WakeupRxStatusPtr	Address where bitcoded wakeup reception status shall be stored. Bit 0: Wakeup received on channel A indicator Bit 1: Wakeup received on channel B indicator Bit 2-7: Unused
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets the wakeup received information from the FlexRay controller.	

[FR669]The function Fr_GetWakeupRxStatus shall perform the following tasks on FlexRay CC Fr_CtrIdx:

1. Read the wakeup pattern received indicators for channel A and channel B and write it to the output parameter Fr_WakeupRxStatusPtr. The value of *Fr_WakeupRxStatusPtr is bitcoded with the following meaning (Bit 0 = LSB, Bit 7 = MSB):
   - Bit 0: Wakeup pattern received on channel A (1), otherwise (0)
   - Bit 1: Wakeup pattern received on channel B (1), otherwise (0)

Bit 2: Not used (always 0)
Bit 3: Not used (always 0)
Bit 4: Not used (always 0)
Bit 5: Not used (always 0)
Bit 6: Not used (always 0)
Bit 7: Not used (always 0)

2. Reset the wakeup received indication status information within the FlexRay controller.
3. Return E_OK.

[FR672] If the function Fr_GetWakeupRxStatus detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

[FR673]If development error detection for the Fr module is enabled, and if the function Fr_GetWakeupRxStatus is called before the successful initialization of Fr, then the function Fr_GetWakeupRxStatus shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

[FR674]If development error detection for the Fr module is enabled, then the function Fr_GetWakeupRxStatus shall check the validity of the parameter Fr_CtrlIdx. If Fr_CtrlIdx is invalid, then the function Fr_GetWakeupRxStatus shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

[FR675]If development error detection for the Fr module is enabled, then the function Fr_GetWakeupRxStatus shall check whether the parameter Fr_WakeupRxStatusPtr is a NULL pointer (NULL_PTR). If Fr_WakeupRxStatusPtr is a NULL pointer, then the function Fr_GetWakeupRxStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.24 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[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_AbsTimerIdx	Index of absolute timer within the context of the FlexRay CC.
	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 (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Sets the absolute FlexRay timer.	

FR374: The Fr module's environment shall only call Fr_SetAbsoluteTimer when the CC Fr_CtrlIdx is synchronized to FlexRay global time (at the moment of timer activation) (=CC precondition for the function Fr_SetAbsoluteTimer)

FR273: The function Fr_SetAbsoluteTimer 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.

FR274: The function Fr_SetAbsoluteTimer 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.

FR272: If the function Fr_SetAbsoluteTimer detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR267: If development error detection for the Fr module is enabled: if the function Fr_SetAbsoluteTimer is called before the Fr was initialized successfully, the function Fr_SetAbsoluteTimer shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR268: If development error detection for the Fr module is enabled: the function Fr_SetAbsoluteTimer shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_SetAbsoluteTimer shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR269: If development error detection for the Fr module is enabled: the function Fr_SetAbsoluteTimer shall check the parameter Fr_AbsTimerIdx for being valid. If Fr_AbsTimerIdx is invalid, the function Fr_SetAbsoluteTimer shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

FR270: If development error detection for the Fr module is enabled: the function Fr_SetAbsoluteTimer shall check the parameter Fr_Cycle for being valid. If Fr_Cycle is invalid, the function Fr_SetAbsoluteTimer shall raise the development error FR_E_INV_CYCLE and return E_NOT_OK.

FR271: If development error detection for the Fr module is enabled: the function Fr_SetAbsoluteTimer shall check the parameter Fr_Offset for being valid. If Fr_Offset is invalid, the function Fr_SetAbsoluteTimer shall raise the development error FR_E_INV_OFFSET and return E_NOT_OK.

FR436: If development error detection for the Fr module is enabled: the function Fr_SetAbsoluteTimer shall check the CC Fr_CtrlIdx for being synchronized to the

FlexRay global time. If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time, the function Fr_SetAbsoluteTimer shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.25 Fr_SetRelativeTimer

FR037:

Service name:	Fr_SetRelativeTimer	
Syntax:	`Std_ReturnType Fr_SetRelativeTimer( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx, uint16 Fr_Offset )`	
Service ID[hex]:	0x12	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_RelTimerIdx	Index of relative timer within the context of the FlexRay CC.
	Fr_Offset	Relative offset from time of service invocation the timer shall elapse at in units of macrotick
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Sets the FlexRay timer.	

FR377: The Fr module's environment shall only call Fr_SetRelativeTimer when the CC Fr_CtrlIdx is synchronized to FlexRay global time (at the moment of timer activation) (=CC precondition for the function Fr_SetRelativeTimer)

FR280: The function Fr_SetRelativeTimer shall perform the following tasks:

1. Program the relative FlexRay timer Fr_RelTimerIdx according to the parameter Fr_Offset.
2. Return E_OK.

FR281: The function Fr_SetRelativeTimer 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.

FR282: The function Fr_SetRelativeTimer shall be pre compile time configurable On/Off by the configuration parameter: FrRelativeTimerEnable.

FR279: If the function Fr_SetRelativeTimer detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR275: If development error detection for the Fr module is enabled: if the function Fr_SetRelativeTimer is called before the Fr was initialized successfully, the function

Fr_SetRelativeTimer shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR276: If development error detection for the Fr module is enabled: the function Fr_SetRelativeTimer shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_SetRelativeTimer shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR277: If development error detection for the Fr module is enabled: the function Fr_SetRelativeTimer shall check the parameter Fr_RelTimerIdx for being valid. If Fr_RelTimerIdx is invalid, the function Fr_SetRelativeTimer shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

FR278: If development error detection for the Fr module is enabled: the function Fr_SetRelativeTimer shall check the parameter Fr_Offset for being valid. If Fr_Offset is invalid, the function Fr_SetRelativeTimer shall raise the development error FR_E_INV_OFFSET and return E_NOT_OK.

FR437: If development error detection for the Fr module is enabled: the function Fr_SetRelativeTimer shall check the CC Fr_CtrlIdx for being synchronized to the FlexRay global time. If the CC Fr_CtrlIdx is not synchronized to the FlexRay global time, the function Fr_SetRelativeTimer shall raise the development error FR_E_INV_POCSTATE and return E_NOT_OK.

8.4.26 Fr_CancelAbsoluteTimer

FR095:

Service name:	Fr_CancelAbsoluteTimer	
Syntax:	`Std_ReturnType Fr_CancelAbsoluteTimer( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx )`	
Service ID[hex]:	0x13	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_AbsTimerIdx	Index of absolute timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Stops an absolute timer.	

CC precondition for the function Fr_CancelAbsoluteTimer: None.

FR287: The function Fr_CancelAbsoluteTimer shall perform the following tasks:

1. Stop the absolute timer Fr_AbsTimerIdx.
2. Return E_OK.

FR286: If the function Fr_CancelAbsoluteTimer detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR283: If development error detection for the Fr module is enabled: if the function Fr_CancelAbsoluteTimer is called before the Fr was initialized successfully, the function Fr_CancelAbsoluteTimer shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR284: If development error detection for the Fr module is enabled: the function Fr_CancelAbsoluteTimer shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_CancelAbsoluteTimer shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR285: If development error detection for the Fr module is enabled: the function Fr_CancelAbsoluteTimer shall check the parameter Fr_AbsTimerIdx for being valid. If Fr_AbsTimerIdx is invalid, the function Fr_CancelAbsoluteTimer shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.27 Fr_CancelRelativeTimer

FR096:

Service name:	Fr_CancelRelativeTimer	
Syntax:	`Std_ReturnType Fr_CancelRelativeTimer( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx )`	
Service ID[hex]:	0x14	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_RelTimerIdx	Index of relative timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Stops a relative timer.	

CC precondition for the function Fr_CancelRelativeTimer: None.

FR292: The function Fr_CancelRelativeTimer shall perform the following tasks:

1. Stop the relative timer Fr_RelTimerIdx.
2. Return E_OK.

FR293: The function Fr_CancelRelativeTimer shall be pre compile time configurable On/Off by the configuration parameter: FrRelativeTimerEnable.

FR291: If the function Fr_CancelRelativeTimer detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR288: If development error detection for the Fr module is enabled: if the function Fr_CancelRelativeTimer is called before the Fr was initialized successfully, the function Fr_CancelRelativeTimer shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR289: If development error detection for the Fr module is enabled: the function Fr_CancelRelativeTimer shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_CancelRelativeTimer shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR290: If development error detection for the Fr module is enabled: the function Fr_CancelRelativeTimer shall check the parameter Fr_RelTimerIdx for being valid. If Fr_RelTimerIdx is invalid, the function Fr_CancelRelativeTimer shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.28 Fr_EnableAbsoluteTimerIRQ

FR034:

Service name:	Fr_EnableAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType Fr_EnableAbsoluteTimerIRQ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx )`	
Service ID[hex]:	0x15	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_AbsTimerIdx	Index of absolute timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Enables the interrupt line of an absolute timer.	

CC precondition for the function Fr_EnableAbsoluteTimerIRQ: None.

FR298: The function Fr_EnableAbsoluteTimerIRQ shall perform the following tasks:
1. Enable the interrupt line related to timer Fr_AbsTimerIdx.
2. Return E_OK.

FR297: If the function Fr_EnableAbsoluteTimerIRQ detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR294: If development error detection for the Fr module is enabled: if the function Fr_EnableAbsoluteTimerIRQ is called before the Fr was initialized successfully, the function Fr_EnableAbsoluteTimerIRQ shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR295: If development error detection for the Fr module is enabled: the function Fr_EnableAbsoluteTimerIRQ shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_EnableAbsoluteTimerIRQ shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR296: If development error detection for the Fr module is enabled: the function Fr_EnableAbsoluteTimerIRQ shall check the parameter Fr_AbsTimerIdx for being valid. If Fr_AbsTimerIdx is invalid, the function Fr_EnableAbsoluteTimerIRQ shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.29 Fr_EnableRelativeTimerIRQ

FR038:

Service name:	Fr_EnableRelativeTimerIRQ	
Syntax:	`Std_ReturnType Fr_EnableRelativeTimerIRQ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx )`	
Service ID[hex]:	0x16	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_RelTimerIdx	Index of relative timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Enables the interrupt line of a relative timer.	

CC precondition for the function Fr_EnableRelativeTimerIRQ: None.

FR303: The function Fr_EnableRelativeTimerIRQ shall perform the following tasks:
1. Enable the interrupt line related to timer Fr_RelTimerIdx.
2. Return E_OK.

FR304: The function function Fr_EnableRelativeTimerIRQ shall be pre compile time configurable On/Off by the configuration parameter: FrRelativeTimerEnable.

FR302: If the function Fr_EnableRelativeTimerIRQ detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR299: If development error detection for the Fr module is enabled: if the function Fr_EnableRelativeTimerIRQ is called before the Fr was initialized successfully, the

function Fr_EnableRelativeTimerIRQ shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR300: If development error detection for the Fr module is enabled: the function Fr_EnableRelativeTimerIRQ shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_EnableRelativeTimerIRQ shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR301: If development error detection for the Fr module is enabled: the function Fr_EnableRelativeTimerIRQ shall check the parameter Fr_RelTimerIdx for being valid. If Fr_RelTimerIdx is invalid, the function Fr_EnableRelativeTimerIRQ shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.30 Fr_AckAbsoluteTimerIRQ

FR036:

Service name:	Fr_AckAbsoluteTimerIRQ	

Syntax:	`Std_ReturnType Fr_AckAbsoluteTimerIRQ(`
	`  uint8 Fr_CtrlIdx,`
	`  uint8 Fr_AbsTimerIdx`
	`)`	

Service ID[hex]:	0x17	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_AbsTimerIdx	Index of absolute timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Resets the interrupt condition of an absolute timer.	

CC precondition for the function Fr_AckAbsoluteTimerIRQ: None.

FR309: The function Fr_AckAbsoluteTimerIRQ shall perform the following tasks:
1. Reset the interrupt condition of absolute timer Fr_AbsTimerIdx.
2. Return E_OK.

FR308: If the function Fr_AckAbsoluteTimerIRQ detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR305: If development error detection for the Fr module is enabled: if the function Fr_AckAbsoluteTimerIRQ is called before the Fr was initialized successfully, the function Fr_AckAbsoluteTimerIRQ shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR306: If development error detection for the Fr module is enabled: the function Fr_AckAbsoluteTimerIRQ shall check the parameter Fr_CtrlIdx for being valid. If

Fr_CtrlIdx is invalid, the function Fr_AckAbsoluteTimerIRQ shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR307: If development error detection for the Fr module is enabled: the function Fr_AckAbsoluteTimerIRQ shall check the parameter Fr_AbsTimerIdx for being valid. If Fr_AbsTimerIdx is invalid, the function Fr_AckAbsoluteTimerIRQ shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.31 Fr_AckRelativeTimerIRQ

FR040:

Service name:	Fr_AckRelativeTimerIRQ	
Syntax:	`Std_ReturnType Fr_AckRelativeTimerIRQ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx )`	
Service ID[hex]:	0x18	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_RelTimerIdx	Index of relative timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Resets the interrupt condition of a releative timer.	

CC precondition for the function Fr_AckRelativeTimerIRQ: None.

FR314: The function Fr_AckRelativeTimerIRQ shall perform the following tasks:
1. Reset the interrupt condition of relative timer Fr_RelTimerIdx.
2. Return E_OK.

FR315: The function Fr_AckRelativeTimerIRQ shall be pre compile time configurable On/Off by the configuration parameter: FrRelativeTimerEnable.

FR313: If the function Fr_AckRelativeTimerIRQ detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR310: If development error detection for the Fr module is enabled: if the function Fr_AckRelativeTimerIRQ is called before the Fr was initialized successfully, the function Fr_AckRelativeTimerIRQ shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR311: If development error detection for the Fr module is enabled: the function Fr_AckRelativeTimerIRQ shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_AckRelativeTimerIRQ shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR312: If development error detection for the Fr module is enabled: the function Fr_AckRelativeTimerIRQ shall check the parameter Fr_RelTimerIdx for being valid. If Fr_RelTimerIdx is invalid, the function Fr_AckRelativeTimerIRQ shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.32 Fr_DisableAbsoluteTimerIRQ

FR035:

Service name:	Fr_DisableAbsoluteTimerIRQ	
Syntax:	`Std_ReturnType Fr_DisableAbsoluteTimerIRQ( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx )`	
Service ID[hex]:	0x19	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_AbsTimerIdx	Index of absolute timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Disables the interrupt line of an absolute timer.	

CC precondition for the function Fr_DisableAbsoluteTimerIRQ: None.

FR320: The function Fr_DisableAbsoluteTimerIRQ shall perform the following tasks:
1. Disable the interrupt line related to absolute timer Fr_AbsTimerIdx.
2. Return E_OK.

FR319: If the function Fr_DisableAbsoluteTimerIRQ detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR316: If development error detection for the Fr module is enabled: if the function Fr_DisableAbsoluteTimerIRQ is called before the Fr was initialized successfully, the function Fr_DisableAbsoluteTimerIRQ shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR317: If development error detection for the Fr module is enabled: the function Fr_DisableAbsoluteTimerIRQ shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_DisableAbsoluteTimerIRQ shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR318: If development error detection for the Fr module is enabled: the function Fr_DisableAbsoluteTimerIRQ shall check the parameter Fr_AbsTimerIdx for being valid. If Fr_AbsTimerIdx is invalid, the function Fr_DisableAbsoluteTimerIRQ shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.33 Fr_DisableRelativeTimerIRQ

FR039:

Service name:	Fr_DisableRelativeTimerIRQ	
Syntax:	`Std_ReturnType Fr_DisableRelativeTimerIRQ( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx )`	
Service ID[hex]:	0x1a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_RelTimerIdx	Index of relative timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Disables the interrupt line of a timer.	

CC precondition for the function Fr_DisableRelativeTimerIRQ: None.

FR325: The function Fr_DisableRelativeTimerIRQ shall perform the following tasks:
1. Disable the interrupt line related to relative timer Fr_RelTimerIdx.
2. Return E_OK.

FR326: The function Fr_DisableRelativeTimerIRQ shall be pre compile time configurable On/Off by the configuration parameter: FrRelativeTimerEnable.

FR324: If the function Fr_DisableRelativeTimerIRQ detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR321: If development error detection for the Fr module is enabled: if the function Fr_DisableRelativeTimerIRQ is called before the Fr was initialized successfully, the function Fr_DisableRelativeTimerIRQ shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR322: If development error detection for the Fr module is enabled: the function Fr_DisableRelativeTimerIRQ shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_DisableRelativeTimerIRQ shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR323: If development error detection for the Fr module is enabled: the function Fr_DisableRelativeTimerIRQ shall check the parameter Fr_RelTimerIdx for being valid. If Fr_RelTimerIdx is invalid, the function Fr_DisableRelativeTimerIRQ shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

8.4.34 Fr_GetAbsoluteTimerIRQStatus

FR108:

Service name:	Fr_GetAbsoluteTimerIRQStatus	
Syntax:	`Std_ReturnType Fr_GetAbsoluteTimerIRQStatus( uint8 Fr_CtrlIdx, uint8 Fr_AbsTimerIdx, boolean* Fr_IRQStatusPtr )`	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_AbsTimerIdx	Index of absolute timer within the context of the FlexRay CC.
Parameters (inout):	None	
Parameters (out):	Fr_IRQStatusPtr	Address the output value is stored to.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets IRQ status of an absolute timer.	

CC precondition for the function Fr_GetAbsoluteTimerIRQStatus: None.

FR332: The function Fr_GetAbsoluteTimerIRQStatus 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.

FR331: If the function Fr_GetAbsoluteTimerIRQStatus detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR327: If development error detection for the Fr module is enabled: if the function Fr_GetAbsoluteTimerIRQStatus is called before the Fr was initialized successfully, the function Fr_GetAbsoluteTimerIRQStatus shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR328: If development error detection for the Fr module is enabled: the function Fr_GetAbsoluteTimerIRQStatus shall check the parameter Fr_CtrlIdx for being valid. If Fr_CtrlIdx is invalid, the function Fr_GetAbsoluteTimerIRQStatus shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR329: If development error detection for the Fr module is enabled: the function Fr_GetAbsoluteTimerIRQStatus shall check the parameter Fr_AbsTimerIdx for being valid. If Fr_AbsTimerIdx is invalid, the function Fr_GetAbsoluteTimerIRQStatus shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

FR330: If development error detection for the Fr module is enabled: the function Fr_GetAbsoluteTimerIRQStatus shall check the parameter Fr_IRQStatusPtr for not

being a NULL pointer (NULL_PTR). If Fr_IRQStatusPtr is a NULL pointer, the function Fr_GetAbsoluteTimerIRQStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.35 Fr_GetRelativeTimerIRQStatus

FR109:

Service name:	Fr_GetRelativeTimerIRQStatus	
Syntax:	`Std_ReturnType Fr_GetRelativeTimerIRQStatus( uint8 Fr_CtrlIdx, uint8 Fr_RelTimerIdx, boolean* Fr_IRQStatusPtr )`	
Service ID[hex]:	0x21	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_RelTimerIdx	Index of relative timer within the context of the FlexRay CC.
	Fr_IRQStatusPtr	Address the output value is stored to.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Gets IRQ status of a relative timer.	

CC precondition for the function Fr_GetRelativeTimerIRQStatus: None.

FR338: The function Fr_GetRelativeTimerIRQStatus 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.

FR339: The function Fr_GetRelativeTimerIRQStatus shall be pre compile time configurable On/Off by the configuration parameter: FrRelativeTimerEnable.

FR337: If the function Fr_GetRelativeTimerIRQStatus detects errors while accessing the CC, it shall call Dem_ReportErrorStatus(FR_E_ACCESS, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR333: If development error detection for the Fr module is enabled: if the function Fr_GetRelativeTimerIRQStatus is called before the Fr was initialized successfully, the function Fr_GetRelativeTimerIRQStatus shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR334: If development error detection for the Fr module is enabled: the function Fr_GetRelativeTimerIRQStatus shall check the parameter Fr_CtrlIdx for being valid.

If Fr_CtrlIdx is invalid, the function Fr_GetRelativeTimerIRQStatus shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR335: If development error detection for the Fr module is enabled: the function Fr_GetRelativeTimerIRQStatus shall check the parameter Fr_RelTimerIdx for being valid. If Fr_RelTimerIdx is invalid, the function Fr_GetRelativeTimerIRQStatus shall raise the development error FR_E_INV_TIMER_IDX and return E_NOT_OK.

FR336: If development error detection for the Fr module is enabled: the function Fr_GetRelativeTimerIRQStatus shall check the parameter Fr_IRQStatusPtr for not being a NULL pointer (NULL_PTR). If Fr_IRQStatusPtr is a NULL pointer, the function Fr_GetRelativeTimerIRQStatus shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

8.4.36 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 (inout):	None	
Parameters (out):	VersioninfoPtr	Pointer to where to store the version information of this module.
Return value:	None	
Description:	Returns the version information of this module.	

FR340: If development error detection for the Fr module is enabled: the function Fr_GetVersionInfo shall check the parameter VersioninfoPtr for not being a NULL pointer (NULL_PTR). If VersioninfoPtr is a NULL pointer, the function Fr_GetVersionInfo shall raise the development error FR_E_INV_POINTER and return.

FR341: The function Fr_GetVersionInfo shall return the version information of this module. The version information includes:

- Module Id
- Vendor Id
- Vendor specific version numbers.

FR342: The function Fr_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: FrVersionInfoApi.

8.4.37 Fr_ReadCCConfig

FR651:

Service name:	Fr_ReadCCConfig

Syntax:	`Std_ReturnType Fr_ReadCCConfig(`
	`  uint8 Fr_CtrlIdx,`
	`  uint8 Fr_ConfigParamIdx,`
	`  uint32* Fr_ConfigParamValuePtr`
	`)`	

Service ID[hex]:	0x2e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same device	
Parameters (in):	Fr_CtrlIdx	Index of FlexRay CC within the context of the FlexRay Driver.
	Fr_ConfigParamIdx	Index that identifies the configuration parameter to read. See macros FR_CIDX_<config_parameter_name>.
Parameters (inout):	None	
Parameters (out):	Fr_ConfigParamValuePtr	Address the output value is stored to.
Return value:	Std_ReturnType	E_OK: API call finished successfully. E_NOT_OK: API call aborted due to errors.
Description:	Reads a FlexRay protocol configuration parameter for a particular FlexRay controller out of the module's configuration.	

The function Fr_ReadCCConfig shall perform the following tasks:

1. Read the value of the configuration parameter requested by Fr_ConfigParamIdx from the configuration and write it to output parameter *Fr_ConfigParamValuePtr.
2. Return E_OK.

FR652: If the function Fr_ReadCCConfig is able to and detects a hardware error while performing the requested functionality, then it shall call Dem_ReportErrorStatus (*FrDemCtrlTestResultRef*, DEM_EVENT_STATUS_FAILED) and return E_NOT_OK.

FR653: If development error detection for the Fr module is enabled, and if the function Fr_ReadCCConfig is called before the successful initialization of Fr, then the function Fr_ReadCCConfig shall raise the development error FR_E_NOT_INITIALIZED and return E_NOT_OK.

FR654: If development error detection for the Fr module is enabled, then the function Fr_ReadCCConfig shall check the validity of the parameter Fr_CtrlIdx. If Fr_CtrlIdx is invalid, then the function Fr_ReadCCConfig shall raise the development error FR_E_INV_CTRL_IDX and return E_NOT_OK.

FR655: If development error detection for the Fr module is enabled, then the function Fr_ReadCCConfig shall check the validity of the parameter Fr_ConfigParamIdx. If Fr_ConfigParamIdx is invalid[6], then the function Fr_ReadCCConfig shall raise the development error FR_E_INV_CONFIG_IDX and return E_NOT_OK.

FR656: If development error detection for the Fr module is enabled, then the function Fr_ReadCCConfig shall check whether the parameter Fr_ConfigParamValuePtr is a NULL pointer (NULL_PTR). If Fr_ConfigParamValuePtr is a NULL pointer, then the

[6] Valid values are listed in chapter 8.2.1 and in requirements FR662, FR663, FR664, FR665, FR666.

function Fr_ReadCCConfig shall raise the development error FR_E_INV_POINTER and return E_NOT_OK.

Configuration parameters values are specified as *integer*, *float*, *enumeration* or *boolean*. In order to map those values to the output parameter of type uint32, the following generic rules for conversion shall be applied for *integer* and *float*:

- **FR658:** *integers* are mapped 1 to 1.
- **FR659:** *floats* (units of seconds) are converted to units of nanoseconds (with nanosecond granularity) and converted to uint32.
- **FR661:** *booleans* shall output 1 for true and 0 for false.

For configuration parameters specified as enumeration type, the following mappings shall be applied:

FR662: If parameter Fr_ConfigParamIdx is set to FR_CIDX_PCHANNELS (FrPChannels) then the value stored at Fr_ConfigParamValuePtr shall be interpreted as the following literals
0 FR_CHANNEL_A
1 FR_CHANNEL_B
2 FR_CHANNEL_AB

FR663: If parameter Fr_ConfigParamIdx is set to FR_CIDX_PSAMPLESPERMICROTICK (FrPSamplesPerMicrotick) then the value stored at Fr_ConfigParamValuePtr shall be interpreted as the following literals
0 N1SAMPLES
1 N2SAMPLES
2 N4SAMPLES

FR664: If parameter Fr_ConfigParamIdx is set to FR_CIDX_PWAKEUPCHANNEL (FrPWakeupChannel) then the value stored at Fr_ConfigParamValuePtr shall be interpreted as the following literals
0 FR_CHANNEL_A
1 FR_CHANNEL_B

FR665: If parameter Fr_ConfigParamIdx is set to FR_CIDX_PDMICROTICK (FrPdMicrotick) then the value stored at Fr_ConfigParamValuePtr shall be interpreted as the following literals
0 T12_5NS
1 T25NS
2 T50NS
3 T100NS
4 T200NS

FR666: If parameter Fr_ConfigParamIdx is set to FR_CIDX_GDSAMPLECLOCKPERIOD (FrIfGdSampleClockPeriod) then the value stored at Fr_ConfigParamValuePtr shall be interpreted as the following literals
0 T12_5NS
1 T25NS
2 T50NS

8.5 Call-back notifications

The FlexRay driver does not call any callbacks.

8.6 Scheduled functions

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

8.7 Expected Interfaces

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

8.7.1 Mandatory Interfaces

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

FR390:

API function	*Description*
Dem_ReportErrorStatus	Reports errors to the DEM.

8.7.2 Optional Interfaces

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

FR391:

API function	*Description*
Det_ReportError	Service to report development errors.
SchM_Enter_Fr	--
SchM_Exit_Fr	--

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

8.7.3 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

VARIANT-POST-BUILD: All configuration parameters in container 'FrGeneral' shall be configurable at pre-compile time. All other configuration parameters shall be configurable at post-build-time.
Use case: Object code delivery, selectable configuration

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

10.2.2 Fr

Module Name	*Fr*
Module Description	Configuration of the Fr (FlexRay driver) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrGeneral	1	General configuration (parameters) of the FlexRay Driver module.
FrMultipleConfiguration	1	Configuration of the individual controllers.

10.2.3 FrGeneral

SWS Item	**FR392 :**
Container Name	FrGeneral
Description	General configuration (parameters) of the FlexRay Driver module.
Configuration Parameters	

SWS Item	**FR900_Conf :**		
Name	FrCCReadbackSupport		
Description	Enables/disbales the "read back and compare" functionality (FR649, FR647, FR598) during the execution of Fr_ControllerInit.		
Multiplicity	0..1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FR393 :**
Name	FrDevErrorDetect
Description	Switches the Development Error Detection and Notification on or off. true: Development Error Detection and Notification enabled. false: Development Error Detection and Notification disabled.
Multiplicity	1

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

SWS Item	**FR439 :**		
Name	FrIndex		
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FR394 :**		
Name	FrNumCtrlSupported		
Description	Determines the maximum number of communication controllers that the driver supports.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FR395 :**		
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.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FR396 :**
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.
Multiplicity	1
Type	BooleanParamDef
Default value	--

ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 FrController

SWS Item	**FR083 :**
Container Name	FrController
Description	Configuration of the individual controller.
Configuration Parameters	

SWS Item	**FR399 :**		
Name	FrCtrlClock		
Description	Determines clock connected to the CC [Hz].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 80000000		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR400 :**		
Name	FrCtrlIdx		
Description	Determines index of CC within Fr.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR402 :**		
Name	FrPAllowHaltDueToClock		
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)		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR403 :**

Name	FrPAllowPassiveToActive		
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		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 31		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR404 :**		
Name	FrPChannels		
Description	Channels to which the node is connected Implementation Type: Fr_ChannelType		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FR_CHANNEL_A	Cluster uses channel A	
	FR_CHANNEL_AB	Cluster uses channel A and B	
	FR_CHANNEL_B	Cluster uses channel B	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR405 :**		
Name	FrPClusterDriftDamping		
Description	Local cluster drift damping factor used for rate correction [Microticks]. Remark: Upper limit 10 for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 20		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR406 :**		
Name	FrPDecodingCorrection		
Description	Value used by the receiver to calculate the difference between primary time reference point and secondary time reference point [Microticks]. Remark: Lower limit 14 for FlexRay Protocol 2.1 Rev. A compliance. Upper limit 136 for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	12 .. 143		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR407 :**

Name	FrPDelayCompensationA		
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 µs to 0.05 µs. In practice, the minimum of the propagation delays of all sync nodes should be applied [Microticks]. Remark: Lower limit 4 for FlexRay Protocol 3.0 compliance. Remark: Upper limit 200 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 211		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR408 :**		
Name	FrPDelayCompensationB		
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 µs to 0.05 µs. In practice, the minimum of the propagation delays of all sync nodes should be applied [Microticks]. Remark: Lower limit 4 for FlexRay Protocol 3.0 compliance. Remark: Upper limit 200 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 211		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR409 :**		
Name	FrPExternOffsetCorrection		
Description	Number of microticks added or subtracted to the NIT to carry out a host-requested external offset correction [Microticks]. Remark: Upper limit 7 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 28		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR410 :**		
Name	FrPExternRateCorrection		
Description	Number of microticks added or subtracted to the cycle to carry out a host-requested external rate correction [Microticks]. Remark: Upper limit 7 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 28		
Default value	--		

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

SWS Item	**FR448 :**		
Name	FrPExternalSync		
Description	Flag indicating whether the node is externally synchronized (operating as time gateway sink in an TT-E cluster) or locally synchronized. If FrPExternalSync is set to 'true' then FrPTwoKeySlotMode must also be set to 'true'. Remarks: Set to 'false' for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR447 :**		
Name	FrPFallBackInternal		
Description	Flag indicating whether a time gateway sink node will switch to local clock operation when synchronization with the time gateway source node is lost (FrPFallBackInternal = true) or will instead go to POC:ready (FrPFallBackInternal =false). Remarks: Set to 'false' for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR411 :**		
Name	FrPKeySlotId		
Description	ID of the key slot, i.e., the slot used to transmit the startup frame, sync frame, or designated key slot frame. If this parameter is set to zero the node does not have a key slot. For Fr3.0: if the value is not provided in System Template it shall be configured to 0. For Fr2.1: if the value is not provided in System Template it is driver implementation specific which value to configure.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 1023		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR425 :**
Name	FrPKeySlotOnlyEnabled
Description	Flag indicating whether or not the node shall enter key slot only mode following

	startup. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter pSingleSlotEnabled.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR412 :**		
Name	FrPKeySlotUsedForStartup		
Description	Flag indicating whether the Key Slot is used to transmit a startup frame		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR413 :**		
Name	FrPKeySlotUsedForSync		
Description	Flag indicating whether the Key Slot is used to transmit a sync frame		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR414 :**		
Name	FrPLatestTx		
Description	Number of the last minislot in which a frame transmission can start in the dynamic segment [Minislots]. Remark: Upper limit 7980 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 7988		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR415 :**
Name	FrPMacroInitialOffsetA
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].		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 68		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR416 :**		
Name	FrPMacroInitialOffsetB		
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].		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 68		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR417 :**		
Name	FrPMicroInitialOffsetA		
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].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 239		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR418 :**		
Name	FrPMicroInitialOffsetB		
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].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 239		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR419 :**
Name	FrPMicroPerCycle

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]. Remark: Lower limit 960 for FlexRay Protocol 3.0 compliance. Upper limit 640000 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	640 .. 1280000		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR420 :**		
Name	FrPMicroPerMacroNom		
Description	This parameter is deprecated and will be removed in future. Old description: Number of microticks per nominal macrotick that all implementations must support [Microticks].		
Multiplicity	0..1		
Type	IntegerParamDef		
Range	40 .. 240		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR444 :**		
Name	FrPNmVectorEarlyUpdate		
Description	Flag indicating when the update of the Network Management Vector in the CHI shall take place. If FrPNmVectorEarlyUpdate is set to false, the update shall take place after the NIT. If FrPNmVectorEarlyUpdate is set to true, the update shall take place after the end of the static segment. Remarks: Set to 'false' for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR421 :**		
Name	FrPOffsetCorrectionOut		
Description	Magnitude of the maximum permissible offset correction value [Microticks]. Remark: Upper limit 15567 for FlexRay Protocol 2.1 Rev A compliance. Remark: Lower limit 15 for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	13 .. 16082		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD

Scope / Dependency	scope: Module

SWS Item	**FR460 :**		
Name	FrPOffsetCorrectionStart		
Description	Start of the offset correction phase within the NIT, expressed as the number of macroticks from the start of cycle [Macroticks]. Remark: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter gOffsetCorrectionStart. Remark: Lower limit 9 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	7 .. 15999		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR422 :**		
Name	FrPPayloadLengthDynMax		
Description	Maximum payload length for dynamic frames [16 bit words].		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 127		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR430 :**		
Name	FrPRateCorrectionOut		
Description	Magnitude of the maximum permissible rate correction value and the maximum drift offset between two nodes operating with unsynchronized clocks for one communication cycle [Microticks]. Remarks: This parameter maps to FlexRay Protocol 2.1 Rev. A parameter pdMaxDrift. Lower limit 3 for FlexRay Protocol 3.0 compliance. Upper limit 1923 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	2 .. 3846		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR424 :**		
Name	FrPSamplesPerMicrotick		
Description	Number of samples per microtick. Remark: Allowed range N1SAMPLES, N2SAMPLES for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	N1SAMPLES	1 sample	
	N2SAMPLES	2 samples	
	N4SAMPLES	4 samples	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	

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

SWS Item	**FR445 :**		
Name	FrPSecondKeySlotId		
Description	ID of the second key slot, in which a second startup frame shall be sent when operating as a coldstart node in a TT-L or TT-D cluster. If this parameter is set to zero the node does not have a second key slot. Remark: Set to 0 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 1023		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR446 :**		
Name	FrPTwoKeySlotMode		
Description	Flag indicating whether node operates as a coldstart node in a TT-E or TT-L cluster. If pTwoKeySlotMode is set to true then both pKeySlotUsedForSync and pKeySlotUsedForStartup must also be set to true. If pExternalSync is set to true then pTwoKeySlotMode must also be set to true. Remark: Set to false for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR426 :**		
Name	FrPWakeupChannel		
Description	Channel used by the node to send a wakeup pattern.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FR_CHANNEL_A		
	FR_CHANNEL_B		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR427 :**		
Name	FrPWakeupPattern		
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. Remark: Lower limit 2 for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 63		
Default value	--		

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

SWS Item	**FR428 :**		
Name	FrPdAcceptedStartupRange		
Description	Expanded range of measured clock deviation allowed for startup frames during integration [Microticks]. Remark: Upper limit 1875 for FlexRay Protocol 2.1 Rev A compliance. Remark: Lower limit 29 for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 2743		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR429 :**		
Name	FrPdListenTimeout		
Description	Value for the startup listen timeout and wakeup listen timeout. Although this is a node local parameter, the real time equivalent of this value should be the same for all nodes in the cluster [Microticks]. Remark: Lower limit 1926 for FlexRay Protocol 3.0 compliance. Upper limit 1283846 for FlexRay Protocol 2.1 Rev. A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1284 .. 2567692		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR431 :**		
Name	FrPdMicrotick		
Description	Duration of a microtick. Remark: Allowed range T12_5NS, T25NS, T50NS for FlexRay Protocol 3.0 compliance.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	T100NS	100 ns	
	T12_5NS	12.5 ns	
	T200NS	200 ns	
	T25NS	25 ns	
	T50NS	50 ns	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrAbsoluteTimer	1..*	Specifies the absolute timer configuration parameters of the Fr.
FrFifo	0..*	One First In First Out (FIFO) queued receive structure,

		defining the admittance criteria to the FIFO, and mandating the ability to admit messages into the FIFO based on Message Id filtering criteria.
FrRelativeTimer	0..*	Specifies the relative timer configuration parameters of the Fr.

10.2.5 FrAbsoluteTimer

SWS Item	**FR432 :**
Container Name	FrAbsoluteTimer
Description	Specifies the absolute timer configuration parameters of the Fr.
Configuration Parameters	

SWS Item	**FR433 :**		
Name	FrAbsTimerIdx		
Description	Contains the index of an absolute timer contained in Fr on a certain FlexRay CC.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 254		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.6 FrRelativeTimer

SWS Item	**FR434 :**
Container Name	FrRelativeTimer
Description	Specifies the relative timer configuration parameters of the Fr.
Configuration Parameters	

SWS Item	**FR435 :**		
Name	FrRelTimerIdx		
Description	Contains the index of a relative timer contained in Fr on a certain FlexRay CC.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 254		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.7 FrFifo

SWS Item	**FR449 :**
Container Name	FrFifo
Description	One First In First Out (FIFO) queued receive structure, defining the admittance criteria to the FIFO, and mandating the ability to admit messages into the FIFO based on Message Id filtering criteria.
Configuration Parameters	

SWS Item	**FR455 :**
Name	FrAdmitWithoutMessageId

Description	Determines whether or not frames received in the dynamic segment that don't contain a message ID will be admitted into the FIFO.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR456 :**		
Name	FrBaseCycle		
Description	FIFO cycle counter acceptance criteria.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 63		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR458 :**		
Name	FrChannels		
Description	FIFO channel admittance criteria.		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FR_CHANNEL_A	Frames received on channel A	
	FR_CHANNEL_AB	Frames received on channel A and B	
	FR_CHANNEL_B	Frames received on channel B	
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR457 :**		
Name	FrCycleRepetition		
Description	FIFO cylce counter acceptance criteria. Valid values are 1,2,4,5,8,10,16,20,32,40,50,64. Remark: Values 1,2,4,8,16,32,64 are valid only for FlexRay Protocol 2.1 Rev A compliance.		
Multiplicity	1		
Type	IntegerParamDef		
Range	1 .. 64		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR459 :**
Name	FrFifoDepth
Description	Fifo Depth.
Multiplicity	1
Type	IntegerParamDef

Range	1 .. 2048		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR453 :**		
Name	FrMsgIdMask		
Description	FIFO message identifier acceptance criteria (Mask filter).		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR454 :**		
Name	FrMsgIdMatch		
Description	FIFO message identifier acceptance criteria (Match filter).		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrRange	0..*	FIFO Frame Id range acceptance criteria.

10.2.8 FrRange

SWS Item	**FR450 :**
Container Name	FrRange
Description	FIFO Frame Id range acceptance criteria.
Configuration Parameters	

SWS Item	**FR452 :**		
Name	FrRangeMax		
Description	Last FrameId of this range that will be accepted by the FIFO.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 2047		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FR451 :**

Name	FrRangeMin		
Description	First FrameId of this range that will be accepted by the FIFO.		
Multiplicity	1		
Type	IntegerParamDef		
Range	0 .. 2047		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

No Included Containers

10.2.9 FrMultipleConfiguration

SWS Item	**FR397 :**
Container Name	FrMultipleConfiguration [Multi Config Container]
Description	Configuration of the individual controllers.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrController	1..*	Container to hold multiple configuration sets.

10.3 Published parameters

The standard common published information like

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

is provided in the BSW Module Description Template (see [11] Figure 4.1 and Figure 7.1).

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