Document Title	Specification of FlexRay Transceiver Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	074
Document Classification	Standard

Document Version	1.6.0
Document Status	Final
Part of Release	3.2
Revision	3

Document Change History

Date	Version	Changed by	Change Description
28.02.2014	1.6.0	AUTOSAR Release Management	• Removed incorrect sequence diagram • Editorial changes • Removed chapter(s) on change documentation
23.05.2012	1.5.0	AUTOSAR Administration	• Minor editorial changes (resolved duplicate service Ids)
04.04.2011	1.4.0	AUTOSAR Administration	• Improved interrupt support by ICU • Legal disclaimer revised
20.09.2010	1.3.0	AUTOSAR Administration	• Clarification of transitions- added FrTrcv474 • Legal disclaimer revised
23.06.2008	1.2.2	AUTOSAR Administration	Legal disclaimer revised
30.01.2008	1.2.1	AUTOSAR Administration	Chapter 9 regenerated from BSW UML Model
17.12.2007	1.2.0	AUTOSAR Administration	• Converted FrTrcv999 into SWS items • Wakeup consolidation • Resolve improvement ambiguities • Tables generated in Chapter 8 and 10 • Document meta information extended • Small layout adaptations made

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.01.2007	1.1.0	AUTOSAR Administration	• Added header file includes: MemMap_<ModuleId>.h and SchM_<ModuleId>.h • Renamed error codes • Support of wake up interrupt sharing (callback only if wake up occurred) • FrTrcv API is only called via FrIf FlexRay Interface, which is transparent to the transceiver driver • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
18.05.2006	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

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module FlexRay Transceiver Driver, which handles the FlexRay transceivers on an ECU.

The FlexRay Transceiver is a hardware device, which mainly transforms the logical 1/0 signals of the µC ports to the bus compliant electrical levels, currents and timings.

Within an automotive environment, there is currently only one single physical layer specification for FlexRay.

In addition, the transceivers could be able to detect electrical malfunctions like break of cable harness in the lifetime of the ECU, ground offsets (a certain ground shift is tolerated) or bus collisions. Depending on the interface, they flag the detected error summarized by a single port pin or very detailed via SPI.

Currently wakeup via bus is mandatory for FlexRay transceivers. Some transceivers also support power supply control. Future markets will probably see a lot of different wakeup/sleep and power supply concepts.

A typical FlexRay transceiver is the TJA1080 for a FlexRay bus with support for fault tolerant communication and wakeup via bus.

Figure 1 depicts the basic structure of the FlexRay stack. One FlexRay Interface accesses several FlexRay Transceivers (Trcv Type X .. Z) using one or several FlexRay Transceiver Driver(s) (FrTrcv Driver Vendor A…C) from different vendors.
A zero based index (FrTrcv_TrcvIdx) identifies the transceiver within the context of the transceiver driver:

E.g. FlexRay transceiver node A of FlexRay transceiver type Z is adressed by the index 0, node B by the index 1 in the example in **Figure 1**.

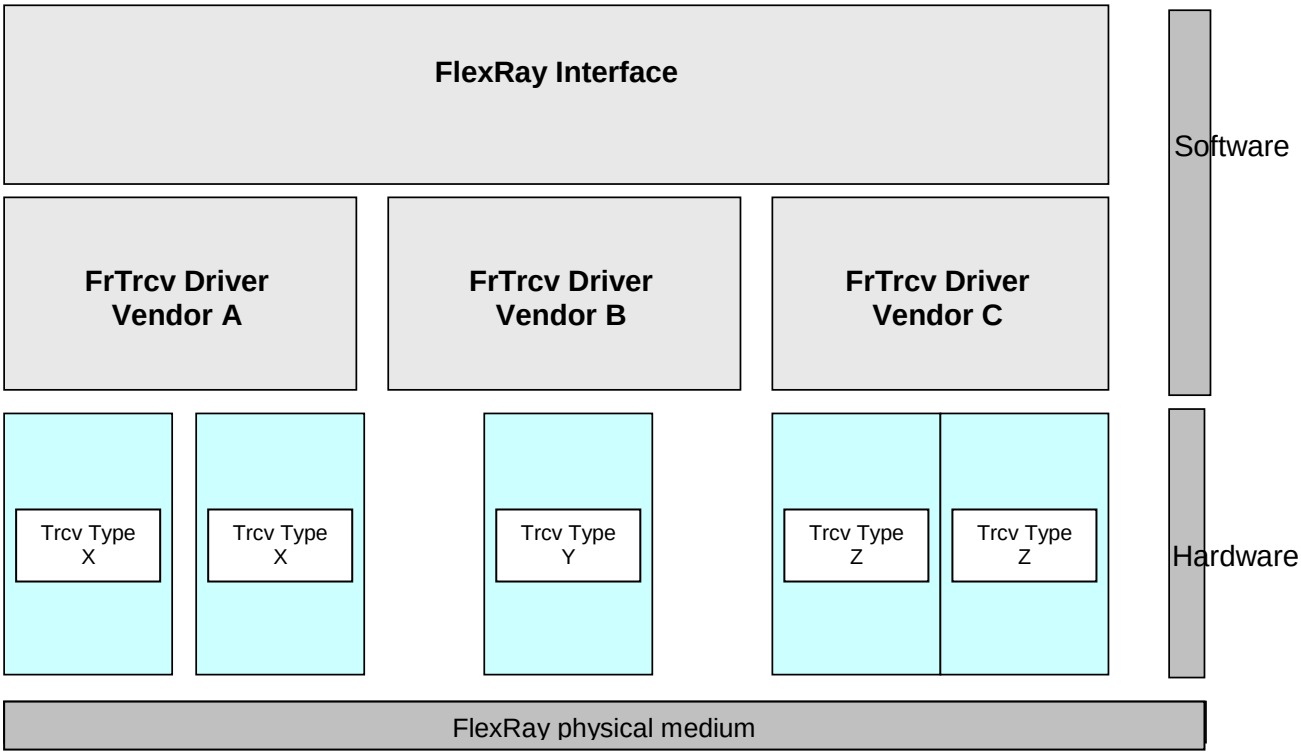


Figure 1: Description of the basic structure of the FlexRay Stack

1.1 Goal of FlexRay transceiver driver

This document specifies interfaces and sequence models, which apply to current and future FlexRay transceiver hardware devices.

The FlexRay transceiver driver abstracts the usage of FlexRay transceiver hardware chips. It offers a hardware independent interface to the higher layers.

The FrTrcv module abstracts from ECU layout by using APIs of MCAL layer to access FlexRay transceiver hardware.

1.2 Explicitly uncovered FlexRay Transceiver Functionality

Some FlexRay Transceivers offer additional functionality like ECU self test or error detection capability for diagnostics.

ECU self test and error detection are not defined within AUTOSAR and requiring such functionality in general will lock out most currently used (and cheap) transceiver devices. Therefore features like “ground shift detection”, “selective wake up”, “slope control” and others are not supported.

1.3 System Basis Chip and FlexRay Transceiver Driver

System basis chips (SBCs) contain beside FlexRay transceiver hardware additional hardware like voltage regulators or watchdogs.

The AUTOSAR concept provides a separate driver for each identified hardware. For AUTOSAR releases 1 and 2 such a driver is missing. Also the application of available drivers which cover functionality inside the SBC is not possible due to shared communication and dependencies of the integrated functions.

Thus AUTOSAR releases 1 and 2 do not support SBCs.

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
CC	Communication Controller
ComM	Communication Manager, See [15] for details
Dem	Diagnostic Event Manager
Det	Development Error Tracer
Dio	Digital input output, one of the SPAL SW modules
EB	Externally buffered channel. Buffers containing data to transfer are outside the SPI Handler/Driver.
EcuM	ECU State Manager, see [14] for details
FlexRay Node	A logical entity connected to the FlexRay Network that is capable of sending and/or receiving frames.
IB	Internally buffered channel. Buffers containing data to transfer are inside the SPI Handler/Driver.
Icu	Input Capture Unit
ISR	Interrupt service routine
MCAL	Micro controller Abstraction Layer
Port	Port, one of the SPAL SW modules
n/a	Not applicable
SBC	System Basis Chip; a device, which integrates e.g. CAN and/or FlexRay and/or LIN transceiver, watchdog and power control.
SPI Channel	A channel is a software exchange medium for data that are defined with the same criteria: configuration parameters, number of data elements with same size and data pointers (source & destination) or location. See specification of SPI driver for more details.
SPI Job	A job is composed of one or several channels with the same chip select. A job is considered to be atomic and therefore cannot be interrupted. A job has also an assigned priority. See specification of SPI driver for more details.
SPI Sequence	A sequence is a number of consecutive jobs to be transmitted. A sequence depends on a static configuration. See specification of SPI driver for more details.

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules.pdf

[2] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture.pdf

[3] ECU Configuration Specification
AUTOSAR_ECU_Configuration.pdf

[4] General Requirements on Basic Software
AUTOSAR_SRS_General.pdf

[5] FlexRay - EPL-Specification - V2.1
FlexRay - EPL-Specification - V2.1.pdf

[6] FlexRay - EPL-Application Notes - V2.1
FlexRay - EPL-Application Notes - V2.1.pdf

[8] FlexRay Communications System - EPL - V2.1 - Errata_3.pdf
FlexRay Communications System - EPL - V2.1 - Errata_3.pdf

3.2 Related standards and norms

[14] Specification of ECU State Manager
AUTOSAR_SWS_ECU_StateManager.pdf

[15] Specification of Communication Manager
AUTOSAR_SWS_ComManager.pdf

[16] Specification of ICU driver
AUTOSAR_SWS_ICU_Driver.pdf

[17] Specification of DIO Driver
AUTOSAR_SWS_DIO_Driver.pdf

[18] Specification of SPI Handler/Driver
AUTOSAR_SWS_SPI_HandlerDriver.pdf

[19] Requirements on FlexRay
AUTOSAR_SRS_FlexRay.pdf

[20] Specification of Communication Stack Types
AUTOSAR_SWS_ComStackTypes.pdf

[21] Specification of BSW Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

[22] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

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

4 Constraints and assumptions

4.1 Limitations

The FlexRay Transceiver must provide functionality and an interface, mapped to the operation mode model assumed for the AUTOSAR FlexRay Transceiver Driver. See 7.1 AUTOSAR FlexRay Transceiver Operation Modes.

FrTrcv231: The FrTrcv module shall use the APIs of underlying drivers (SPI and Dio) synchronously.

Implementations of underlying drivers that do not support synchronous behavior cannot be used together with FlexRay Transceiver Driver.

4.2 Applicability to car domains

This driver shall be applicable in all car domains using FlexRay for communication.

5 Dependencies to other modules

Module	*Dependencies*
ComM	ComM steers FlexRay Transceiver Driver communication modes. Independent steering of each single FlexRay transceiver node via FrIf.
Det	Det gets development error information from FlexRay Transceiver Driver.
Dem	Dem gets production error information from FlexRay Transceiver Driver.
Dio	Dio module is used to access FlexRay transceiver hardware connected via ports.
EcuM	EcuM gets wake up event information from FlexRay Transceiver Driver.
Icu	Icu module performs FlexRay transceiver hardware interrupts and calls appropriate call-back function inside FlexRay Transceiver Driver.
OS	FlexRay Transceiver Driver cannot access the AUTOSAR OS directly but must go through the BSW Module Scheduler!
SPI	SPI module is used to access FlexRay transceiver hardware connected via SPI.

Please be aware although this documentation of the FlexRay transceiver consumes more of 50 pages of paper, in the end it will still resolve to setting a few bits in RAM and transferring them via SPI or setting a few port pins. This can be VERY small code (e.g. inline functions) in case post build time configuration is not required.

If an upper layer wants to call any FlexRay transceiver node specific FlexRay API, knowledge which FlexRay transceiver driver it has to call for a specific communication FlexRay transceiver node **is not required**. Only a mapping (=knowledge) generated by configuration is required!

Here is an example:

Upper layer:

"Set transceivers of cluster C (within a single ECU) to state NORMAL"

FrIf (has cluster knowledge):
Cluster C uses CC Y which is connected to Xcvr Xa (FlexRay transceiver node A) and Xb (FlexRay transceiver node B)
"Set transceivers Xa and Xb to state NORMAL"

FrXcvr (has transceiver driver knowledge, assuming different drivers):
transceiver Xa is the 1st device within driver D1
transceiver Xb is the 3rd device within driver D2
"set Xa to normal via D1(1st device)"
"set Xb to normal via D2(3rd device)"

FlexRay Transceiver Driver FrXcfrD1 (has Xcvr HW knowledge):
NORMAL for 1st device is achieved by setting Dio signal S1 to HIGH and DIO Signal S2 to HIGH
"DIO set S1 and S2 to HIGH"

ECU Abstraction Layer (has ECU layout information):
Signal S1 is mapped to DIO channel C7
Signal S2 is mapped to DIO channel C8

DIO (has port/pin knowledge)
configuration maps C7 to PORTs.PINn and C8 to PORTt.PINm

set S1 to HIGH via PORTs.PINn ((Dio_WriteChannel(S1, Std_High);)
set S2 to HIGH via PORTt.PINm ((Dio_WriteChannel(S2, Std_High);)

5.1 File structure

5.1.1 Naming convention for transceiver driver implementation

FrTrcv059: A FlexRay Transceiver Driver implementation may support different FlexRay transceiver hardware. Thus BSW00347 is applied for the naming in a way that no FlexRay transceiver hardware specific naming extensions are used.

5.1.2 Code file structure

FrTrcv021: Naming convention applies to all files.

FrTrcv058: The FrTrcv module configuration is contained in *_Cfg.c

The FrTrcv module consists of the following files:

File name	*Requirements*	*Description*
FrTrcv.c	**FrTrcv033**:	The implementation general c file. It does not contain interrupt routines.
FrTrcv_Cfg.c	**FrTrcv058**	Pre compile time configuration code file. It is generated by the configuration tool.

5.1.3 Header file structure

The header file structure shall include the following FlexRay-specific header files:

File name	*Requirements*	*Description*
FrTrcv.h	FrTrcv022: FrTrcv113:	General header file of the FlexRay Transceiver Driver. It contains only information relevant for other BSW modules (API). Differences in API depending on configuration are encapsulated.
FrTrcv_Cfg.h	FrTrcv110:	Pre compile time configuration parameter file. It's generated by the configuration tool.
Fr_GeneralTypes.h		contains declarations shared by all AUTOSAR FlexRay BSW modules
SchM_FrTrcv.h	FrTrcv266:	contains schedule manager declarations used by the FlexRay Transceiver Driver specified by [21]
MemMap.h	FrTrcv267:	apply the memory mapping abstraction mechanisms specified by [22]

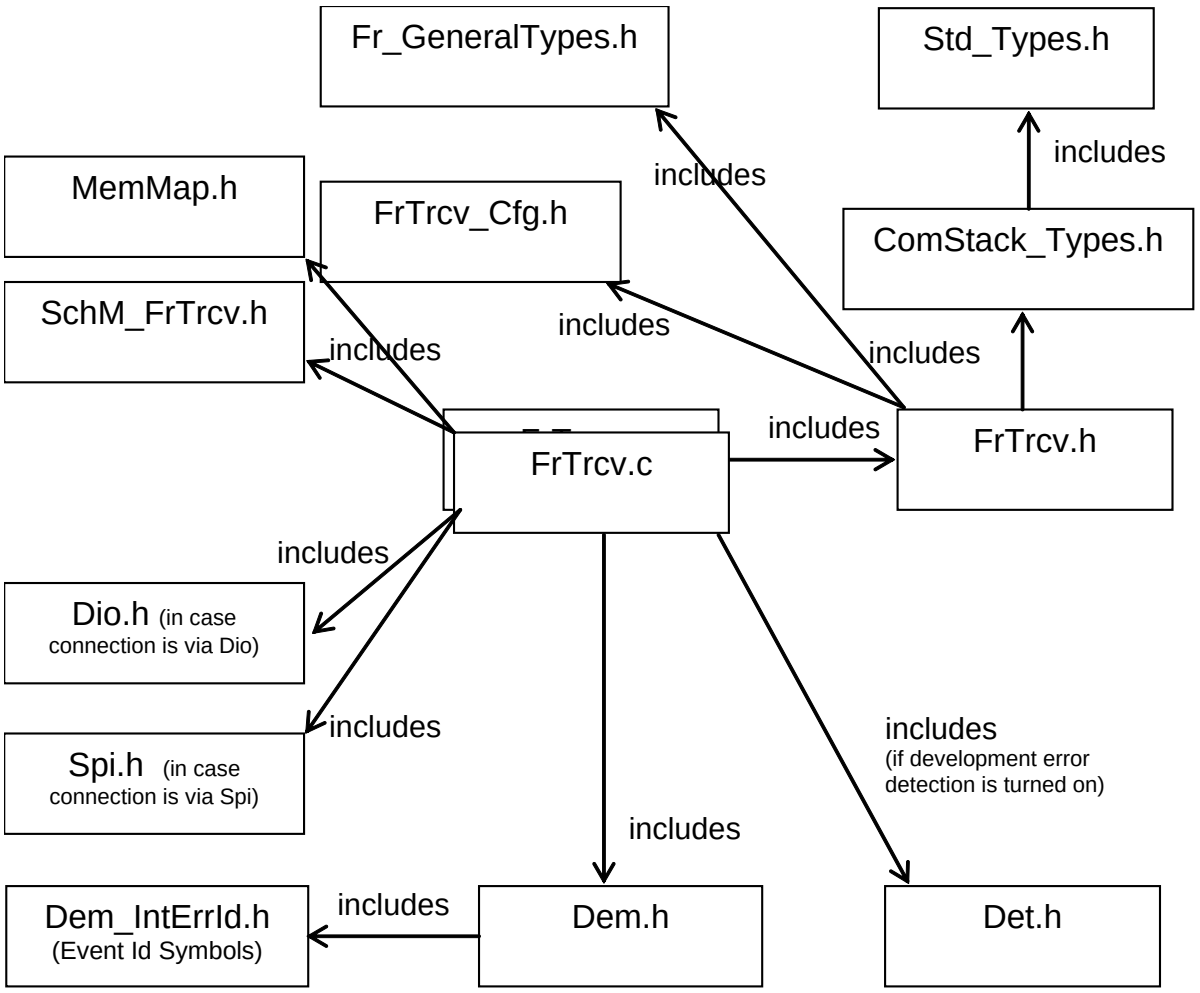


Figure 2 FlexRay Transceiver Driver Header File Structure

FrTrcv335: The module FrTrcv 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.

FrTrcv060: Std_Types.h includes platform specifc header files and compiler specifc header files. It defines standard data types and values for standard defines.
This file is indirectly included via ComStack_Types.h.

FrTrcv068: Name of compiler specific header file is Compiler.h. All mappings of not standardized keywords of compiler specific scope shall be placed and organized in this compiler specific type and keyword header.
This file is indirectly included via ComStack_Types.h.

FrTrcv062: Name of platform specific header file is Platform_Types.h. All integer type definitions of target and compiler specific scope shall be placed and organized in this single type header.
This file is indirectly included via ComStack_Types.h.

Fr_GeneralTypes.h contains definitions shared by the FlexRay Transceiver Driver with the FlexRay driver and the FlexRay interface.
FrTrcv266: BSW scheduler information is included via SchM_FrTrcv.h.
FrTrcv267: BSW memory mapping information is included via MemMap.h.

6 Requirements traceability

6.1 Document: AUTOSAR requirements on Basic Software, general

Requirement	*Description*	*Satisfied by*
[BSW00003]	Version identification	FrTrcv001
[BSW00004]	Version check	not applicable to this SWS, general implementation requirement
[BSW00005]	No hard coded horizontal interfaces within MCAL	not applicable, FlexRay transceiver driver is part of ECU abstraction layer. It is not part of uC abstraction layer.
[BSW00006]	Platform independency	FrTrcv267
[BSW00007]	HIS MISRA C	not applicable to this SWS, general implementation requirement
[BSW00009]	Module User Documentation	not applicable, General documentation requirement
[BSW00010]	Memory resource documentation	not applicable, General documentation requirement
[BSW00101]	Initialization interface	FrTrcv008
[BSW00158]	Separation of configuration from implementation	See section 5.1.2
[BSW00159]	Tool-based configuration	FrTrcv010
[BSW00160]	Human-readable configuration data	FrTrcv011
[BSW00161]	Microcontroller abstraction	not applicable, FlexRay transceiver driver is part of ECU abstraction layer. It is not part of uC abstraction layer.
[BSW00162]	ECU layout abstraction	See Section 1.1
[BSW00164]	Implementation of interrupt service routines	not applicable, FlexRay transceiver driver does not implement interrupt service routines
[BSW00167]	Static configuration checking	FrTrcv016
[BSW00168]	Diagnostic Interface of SW components	not applicable, FlexRay transceiver driver does not implement tester routines
[BSW00170]	Data for reconfiguration of AUTOSAR SW-Components	FrTrcv018
[BSW00171]	Configurability of optional functionality	FrTrcv019
[BSW00172]	Compatibility and documentation of scheduling strategy	FrTrcv020
[BSW00300]	Module naming convention	FrTrcv021
[BSW00301]	Limit imported information	FrTrcv022
[BSW00302]	Limit exported information	FrTrcv023
[BSW00304]	AUTOSAR integer data types	not applicable to this SWS, general implementation requirement
[BSW00305]	Self-defined data types naming convention	FrTrcv025
[BSW00306]	Avoid direct use of compiler and platform specific keywords	not applicable to this SWS, general implementation requirement
[BSW00307]	Global variables naming convention	not applicable to this SWS, general implementation requirement

[BSW00308]	Definition of global data	not applicable to this SWS, general implementation requirement
[BSW00309]	Global data with read-only constraint	not applicable to this SWS, general implementation requirement
[BSW00310]	API naming convention	FrTrcv030
[BSW00312]	Shared code shall be reentrant	not applicable to this SWS, general implementation requirement
[BSW00314]	Separation of interrupt frames and service routines	FrTrcv033
[BSW00318]	Format of module version numbers	FrTrcv034
[BSW00321]	Enumeration of module version numbers	not applicable to this SWS, general implementation requirement
[BSW00323]	API parameter checking	FrTrcv037
[BSW00325]	Runtime of interrupt service routines	not applicable, FlexRay transceiver driver does not implement interrupt service routines
[BSW00326]	Transition from ISRs to OS tasks	not applicable, no such transitions are performed
[BSW00327]	Error values naming convention	FrTrcv041
[BSW00328]	Avoid duplication of code	not applicable to this SWS, general implementation requirement
[BSW00329]	Avoidance of generic interfaces	FrTrcv043
[BSW00330]	Usage of macros / inline functions instead of functions	not applicable to this SWS, general implementation requirement
[BSW00331]	Separation of error and status values	FrTrcv045
[BSW00333]	Documentation of call-back function context	not applicable, General documentation requirement
[BSW00334]	Provision of XML file	FrTrcv011not applicable to this SWS, general implementation requirement
[BSW00335]	Status values naming convention	FrTrcv048
[BSW00336]	Shutdown interface	not applicable, FlexRay transceiver driver does not have a need for such an interface
[BSW00337]	Classification of errors	FrTrcv050
[BSW00338]	Detection and Reporting of development errors	FrTrcv051
[BSW00339]	Reporting of production relevant error status	FrTrcv052
[BSW00341]	Microcontroller compatibility documentation	not applicable, General documentation requirement
[BSW00342]	Usage of source code and object code	not applicable to this SWS, general implementation requirement
[BSW00343]	Specification and configuration of time	FrTrcv055
[BSW00344]	Reference to link-time configuration	not applicable, FlexRay transceiver driver supports only pre compile time configuration
[BSW00345]	Pre-compile-time configuration	FrTrcv057
[BSW00346]	Basic set of module files	FrTrcv058
[BSW00347]	Naming separation of different instances of BSW drivers	FrTrcv059

[BSW00348]	Standard type header	FrTrcv060
[BSW00350]	Development error detection keyword	FrTrcv061
[BSW00353]	Platform specific type header	FrTrcv062
[BSW00355]	Do not redefine AUTOSAR integer data types	not applicable to this SWS, general implementation requirement
[BSW00357]	Standard API return type	FrTrcv064
[BSW00358]	Return type of init() functions	
[BSW00359]	Return type of call-back functions	
[BSW00360]	Parameters of call-back functions	
[BSW00361]	Compiler specific language extension header	FrTrcv068
[BSW00369]	Do not return development error codes via API	FrTrcv069
[BSW00370]	Separation of call-back interface from API	
[BSW00371]	Do not pass function pointers via API	FrTrcv071
[BSW00373]	Main processing function naming convention	FrTrcv072
[BSW00374]	Module vendor identification	FrTrcv073
[BSW00375]	Notification of wake-up reason	FrTrcv074
[BSW00376]	Return type and parameters of main processing functions	FrTrcv075
[BSW00377]	Module specific API return types	
[BSW00378]	AUTOSAR boolean type	not applicable to this SWS, general implementation requirement
[BSW00379]	Module identification	FrTrcv078
[BSW00380]	Separate C-Files for configuration parameters	FrTrcv079
[BSW00381]	Separate configuration header file for pre-compile time parameters	FrTrcv080
[BSW00382]	Not-used configuration elements need to be listed [rejected]	not applicable, General documentation requirement
[BSW00383]	List dependencies of configuration files	not applicable, General documentation requirement
[BSW00384]	List dependencies to other modules	not applicable, General documentation requirement
[BSW00385]	List possible error notifications	
[BSW00386]	Configuration for detecting an error	FrTrcv085
[BSW00387]	Specify the configuration class of call-back function	FrTrcv086
[BSW00388]	Introduce containers	FrTrcv087
[BSW00389]	Containers shall have names	FrTrcv088
[BSW00390]	Parameter content shall be unique within the module	FrTrcv089
[BSW00391]	Parameter shall have unique names	FrTrcv090
[BSW00392]	Parameters shall have a type	FrTrcv091

[BSW00393]	Parameters shall have a range	FrTrcv092
[BSW00394]	Specify the scope of the parameters	FrTrcv093
[BSW00395]	List the required parameters (per parameter)	FrTrcv094
[BSW00396]	Configuration classes	FrTrcv095
[BSW00397]	Pre-compile-time parameters	FrTrcv317
[BSW00398]	Link-time parameters	not applicable, FlexRay transceiver driver supports only pre compile time configuration
[BSW00399]	Loadable Post-build time parameters	not applicable, FlexRay transceiver driver supports only pre compile time configuration
[BSW00400]	Selectable Post-build time parameters	not applicable, FlexRay transceiver driver supports only pre compile time configuration
[BSW00401]	Documentation of multiple instances of configuration parameters	not applicable, General documentation requirement
[BSW00402]	Published information	FrTrcv101
[BSW00404]	Reference to post build time configuration	not applicable, FlexRay transceiver driver supports only pre compile time configuration
[BSW00405]	Reference to multiple configuration sets	not applicable, FlexRay transceiver driver supports only pre compile time configuration
[BSW00406]	Check module initialization	FrTrcv104
[BSW00407]	Function to read out published parameters	FrTrcv105
[BSW00408]	Configuration parameter naming convention	FrTrcv106
[BSW00409]	Header files for production code error IDs	FrTrcv107
[BSW00410]	Compiler switches shall have defined values	not applicable to this SWS, general implementation requirement
[BSW00411]	Get version info keyword	FrTrcv109
[BSW00412]	Separate H-File for configuration parameters	FrTrcv110
[BSW00413]	Accessing instances of BSW modules	not applicable, this is out of FlexRay transceiver driver's scope
[BSW00414]	Parameter of init function	FrTrcv112
[BSW00415]	User dependent include files	FrTrcv113
[BSW00416]	Sequence of Initialization	not applicable, this is out of FlexRay transceiver driver's scope
[BSW00417]	Reporting of Error Events by Non-Basic Software	not applicable, requirement concerns application components only
[BSW00419]	Separate C-Files for pre-compile time configuration parameters	FrTrcv117
[BSW00420]	Production relevant error event rate detection	not applicable, it's an Dem requirement
[BSW00421]	Reporting of production relevant error events	FrTrcv119

[BSW00422]	Debouncing of production relevant error status	not applicable, it's an Dem requirement
[BSW00423]	Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	not applicable to this SWS, general implementation requirement
[BSW00424]	BSW main processing function task allocation	FrTrcv122
[BSW00425]	Trigger conditions for schedulable objects	FrTrcv123
[BSW00426]	Exclusive areas in BSW modules	not applicable (no exclusive areas specified for this module)
[BSW00427]	ISR description for BSW modules	not applicable, no such areas or function in FlexRay transceiver driver
[BSW00428]	Execution order dependencies of main processing functions	FrTrcv126
[BSW00429]	Restricted BSW OS functionality access	not applicable to this SWS, general implementation requirement
[BSW00431]	The BSW Scheduler module implements task bodies	not applicable, requirement concerns BSW scheduler module
[BSW00432]	Modules should have separate main processing functions for read/receive and write/transmit data path	not applicable, FlexRay transceiver driver does not propagate data
[BSW00433]	Calling of main processing functions	not applicable, requirement concerns BSW scheduler module
[BSW00434]	The Schedule Module shall provide an API for exclusive areas	not applicable, requirement concerns BSW scheduler module
[BSW00435]	Header File Structure for the Basic Software Scheduler	FrTrcv266
[BSW00436]	Module Header File Structure for the Memory Mapping	FrTrcv267

6.2 Document: AUTOSAR_SRS_FlexRay.SRS

Requirement	*Description*	*Satisfied by*
[BSW05000]	Support of Synchronous SW Modules	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05001]	Support of Asynchronous SW Modules	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05002]	FlexRay Modules as Only Necessarily Synchronous SW Modules	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05003]	Support of Slot/Cycle Multiplexing	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05004]	PDU-Based Data API	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05005]	Support of Hardware FIFO	not applicable, this is out of FlexRay

	Mechanism	transceiver driver's scope
[BSW05006]	Abstraction of FlexRay-Specific Features	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05007]	Number of FlexRay CCs per Interface	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05009]	Local Memory Space Usage	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05010]	Unique PDU-ID	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05011]	Initialize Low-Level Parameters	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05012]	Initialize FlexRay CC Transmit/Receive Buffers	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05013]	Initialize Local Memory Space	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05015]	Start-up of a FlexRay CC	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05016]	Abortion of a FlexRay CC Communication	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05018]	Sending of a Wake-Up Pattern	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05019]	Get FlexRay Global Time	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05022]	Get FlexRay CC POC Status	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05023]	Get FlexRay CC Sync State	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05024]	Abstraction from CC Buffer Configuration	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05025]	Consistent Access to All Local Memory	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05027]	Transmit PDU	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05031]	Initialization of a FlexRay CC	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05033]	Tick Conversion	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05034]	Configuration Modifiable by a Flashing Process	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05035]	MTS Sending	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05038]	Get MTS Reception Status	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05039]	Set FlexRay Transceiver Operation Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05040]	Get FlexRay Transceiver Error State	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05041]	Consistent Access to a PDU's Local Memory	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05042]	Switch Configuration in Normal Active Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05044]	Set Absolute Timer	not applicable, this is out of FlexRay transceiver driver's scope

[BSW05045]	Set Relative Timer	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05046]	Enable Absolute Alarms	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05047]	Disable Absolute Alarms	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05048]	Acknowledge Absolute Alarms	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05049]	Enable Relative Alarms	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05050]	Disable Relative Alarms	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05051]	Acknowledge Relative Alarms	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05052]	Get Cycle Length in Macroticks	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05053]	Cluster External Clock Synchronization	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05055]	Avoid Timer Interrupts during Shutdown	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05056]	Configuration of the FlexRay Interface at System Configuration Time	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05058]	Configuration of FlexRay Driver at System Configuration Time	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05059]	Transmit/Receive Buffer Configuration	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05060]	Scheduling of Copy Operation into/from FlexRay CC	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05063]	Halt of a FlexRay CC Communication	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05064]	Abstraction of FlexRay CC-specific Implementation	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05065]	Number of FlexRay CCs per Driver	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05066]	L-SDU-Based API	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05067]	Set FlexRay Cluster Offline Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05068]	Set FlexRay Cluster Online Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05069]	Get FlexRay Cluster Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05072]	FlexRay Time Services Access if CC is Out of Sync	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05073]	Usage of ISO 15765-2 and ISO 15765-4 Specifications	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05074]	FlexRay Transport Layer Interfaces	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05075]	Independence of the Network Configuration	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05076]	Multiple Logical FlexRay Transport Layer Channels	not applicable, this is out of FlexRay transceiver driver's scope

[BSW05077]	Unique Identifier of N-SDU	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05078]	Initialization of a FlexRay Cluster	not applicable, cluster abstraction is handled by the FlexRay interface FrIf
[BSW05079]	Transport Connection Properties	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05082]	Acknowledgement without Retry	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05083]	Acknowledgement with Retry	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05084]	PDU Length	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05085]	Segmented 1:n Connections without Flow Control	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05088]	FlexRay Transport Layer Initialization	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05089]	FlexRay Transport Layer Availability	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05090]	Support of Optional ISO 15765-2 Service	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05093]	Transmit Cancellation	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05095]	Bandwidth Control	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05096]	Assignment of Drivers to Controllers	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05097]	Number of FlexRay Drivers per FlexRay Interface	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05101]	Start-up of a FlexRay Cluster	not applicable, cluster abstraction is handled by the FlexRay interface FrIf
[BSW05102]	Halt of a FlexRay Cluster Communication	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05104]	Default Separation Time	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05106]	Buffer Reconfiguration in Normal Active Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05107]	MTS Sending	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05109]	Start-up of a FlexRay CC	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05111]	Get MTS Reception Status	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05113]	Abortion of a FlexRay Cluster Communication	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05114]	Abortion of FlexRay CC Communication	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05115]	Halt of FlexRay CC Communication	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05116]	Initialization of FlexRay CC	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05117]	Sending of Wake-Up Pattern	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05120]	Get FlexRay CC POC Status	not applicable, this is out of FlexRay

		transceiver driver's scope
[BSW05121]	Get FlexRay CC Sync State	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05123]	Configuration Modifiable by a Flashing Process	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05124]	Global Transport Layer Properties	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05125]	Interrupt Handling	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05126]	PDU Update/Valid Information	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05130]	Transmit Request Queuing	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05131]	Configuration Data for FlexRay Transceiver	FrTrcv225
[BSW05132]	Support for More than One FlexRay Transceiver	FrTrcv226
[BSW05133]	Configuration of Bus Operation Mode after Initialization for Each FlexRay Transceiver	FrTrcv227
[BSW05134]	Initialization Sequence for FlexRay Transceiver Driver	
[BSW05136]	Configuration "Notification for Wake-up by Bus"	FrTrcv229
[BSW05137]	Initialize the FlexRay Transceiver Driver	FrTrcv230
[BSW05138]	FlexRay Transceiver Driver API shall be Synchronous	FrTrcv231
[BSW05144]	Get FlexRay Transceiver Wake-up Reason	FrTrcv232
[BSW05147]	Notification for Wake-up by Bus	FrTrcv233
[BSW05148]	Support for Wake-up During Sleep Transition	FrTrcv234
[BSW05149]	Support API to Enable/Disable and Clear Wake-up Events	FrTrcv235
[BSW05150]	Safe System Shutdown for FlexRay Transceiver Driver	FrTrcv236
[BSW05151]	FlexRay Transceiver Driver Must Check Transceiver Control	FrTrcv237
[BSW05152]	Transceiver-specific Timing Requirements	FrTrcv238
[BSW05153]	Get FlexRay Controller Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05154]	Set FlexRay Controller Offline Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05155]	Set FlexRay Controller Online Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05156]	Controller External Clock Synchronization	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05157]	Get FlexRay Transceiver Operation Mode	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05158]	Get FlexRay Transceiver Wake-	not applicable, this is out of FlexRay

	up Reason	transceiver driver's scope
[BSW05159]	Enable FlexRay Transceiver Wake-up Indication	FrTrcv245
[BSW05160]	Disable FlexRay Transceiver Wake-up Indication	FrTrcv246
[BSW05161]	Clear FlexRay Transceiver Wake-up Events	FrTrcv247
[BSW05162]	Set Cluster-wide FlexRay Transceiver Operation Mode	not applicable, cluster abstraction is handled by the FlexRay interface FrIf
[BSW05163]	Cluster-wide Enable FlexRay Transceiver Wake-up Indication	not applicable, cluster abstraction is handled by the FlexRay interface FrIf
[BSW05164]	Cluster-wide Disable FlexRay Transceiver Wake-up Indication	not applicable, cluster abstraction is handled by the FlexRay interface FrIf
[BSW05165]	Cluster-wide Clear FlexRay Transceiver Wake-up Events	not applicable, cluster abstraction is handled by the FlexRay interface FrIf
[BSW05166]	Set FlexRay Transceiver Operation Mode	FrTrcv252
[BSW05167]	Get FlexRay Transceiver Operation Mode	FrTrcv253
[BSW05168]	Indicate FlexRay Transceiver Error State	FrTrcv254
[BSW05169]	Avoid Timer Interrupts during Start-up	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05170]	Receive PDU	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05171]	Provide PDU Transmit Confirmation	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05172]	Get NMVector	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05173]	Get NMVector	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05174]	Interrupt Handling	not applicable, this is out of FlexRay transceiver driver's scope
[BSW05175]	Provide Error Information	not applicable, this is out of FlexRay transceiver driver's scope

6.3 Document: AUTOSAR_SWS_ECU_StateManager.pdf

Requirement	***Description***	***Satisfied by***
[EcuM2330]	Wakeup sources have to be handled and encapsulated by drivers. The implementation must follow the protocols and requirements presented in this document to ensure a seamless integration into AUTOSAR BSW.	FrTrcv074
EcuM2482:	To support the wakeup and validation protocol, the driver has to fulfill the following requirements:	FrTrcv074
EcuM2483:	The driver has to notify ECU State Manager by invoking the EcuM_SetWakeupEvent service once when a wakeup event is detected. The same service should also be invoked during initialization of the driver if a pending wakeup event is detected	FrTrcv262

	during the initialization. Preferably, the invocation is done from a callout or function stub of the caller, to decouple driver modules and ECU State Manager.	
EcuM2486:	The driver shall provide an explicit service to put the wakeup source to sleep. This service shall put the wakeup source into a energy saving and inert operation mode and re-arm the wakeup notification mechanism.	FrTrcv236
EcuM2485:	If the wakeup source is capable of generating faulty events then the driver or the software stack consuming the driver or another appropriate BSW module shall either provide a validation callout for the wakeup event under validation or directly call the wakeup validation service of the ECU State Manager. If validation is not necessary, then this requirement is not applicable for the according wakeup source.	not applicable, FlexRay transceiver driver has no such needs
EcuM2560:	Some drivers may need re-initialization when the ECU is woken up. This is especially true for drivers with wakeup sources. For re-initialization, a restart block is defined. The restart block is part of the WAKEUP state.	not applicable, FlexRay transceiver driver has no such needs
EcuM2561:	The restart list will typically only contain a subset of drivers. But drivers shall appear in the same order as in the combined list of init block I and init block II (see 10.3 Configurable Parameters, ECUM_DRIVER_RESTART_LIST).	not applicable, this is out of FlexRay transceiver driver's scope
EcuM2562:	Drivers which serve wakeup sources must be re-initialized in the restart block. The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back (see 7.7.4.1 WAKEUP I).	FrTrcv263
EcuM2563:	If hardware is put into a sleep mode during SHUTDOWN then this hardware must be restarted by its driver.	not applicable, this is out of FlexRay transceiver driver's scope
EcuM2745:	The restart list will be invoked in state WAKEUP I (see 7.1.5 WAKEUP State).	not applicable, this is out of FlexRay transceiver driver's scope
EcuM2627:	During the initialization, the driver must detect if the CAN transceiver switched on the power supply because of a passive wakeup. The driver shall notify this. In this case, the system designer is responsible to clear all previous wakeup sources and to set the CAN transceiver as the wakeup source by using the EcuM_SetWakeupEvent. This requirement only applies for systems where the CAN transceiver controls the power supply to implement the SLEEP state.	not applicable, this is out of FlexRay transceiver driver's scope
EcuM2569:	Pending events are validated with a call to EcuM_ValidateWakeupEvent. This call must be placed in the driver or the consuming stack on top of the driver (e.g. the handler). The best place to	not applicable, this is out of FlexRay transceiver driver's scope

	put this depends on hardware and software design. See also 7.8.5 Requirements for Drivers with Wakeup Sources.	

6.4 Document: AUTOSAR_SWS_ComStackTypes.doc

Requirement	*Description*	*Satisfied by*
COMTYPE021:	General Codes	
COMTYPE022:	The Communication System dependent Return codes shall be named as follows: <BUS>TRCV_E_FR_<Error Code Name>. Error Code Name: self explaining name of error return code.	FrTrcv265

6.5 Document: AUTOSAR_SWS_BSW_Scheduler.pdf

Requirement	*Description*	*Satisfied by*
INTEGR092:	Each BSW module implementation <ModulePrefix>.c shall include its respective header file SchM_<ModulePrefix>.h.	FrTrcv266

6.6 Document: AUTOSAR_SWS_MemoryMapping.pdf

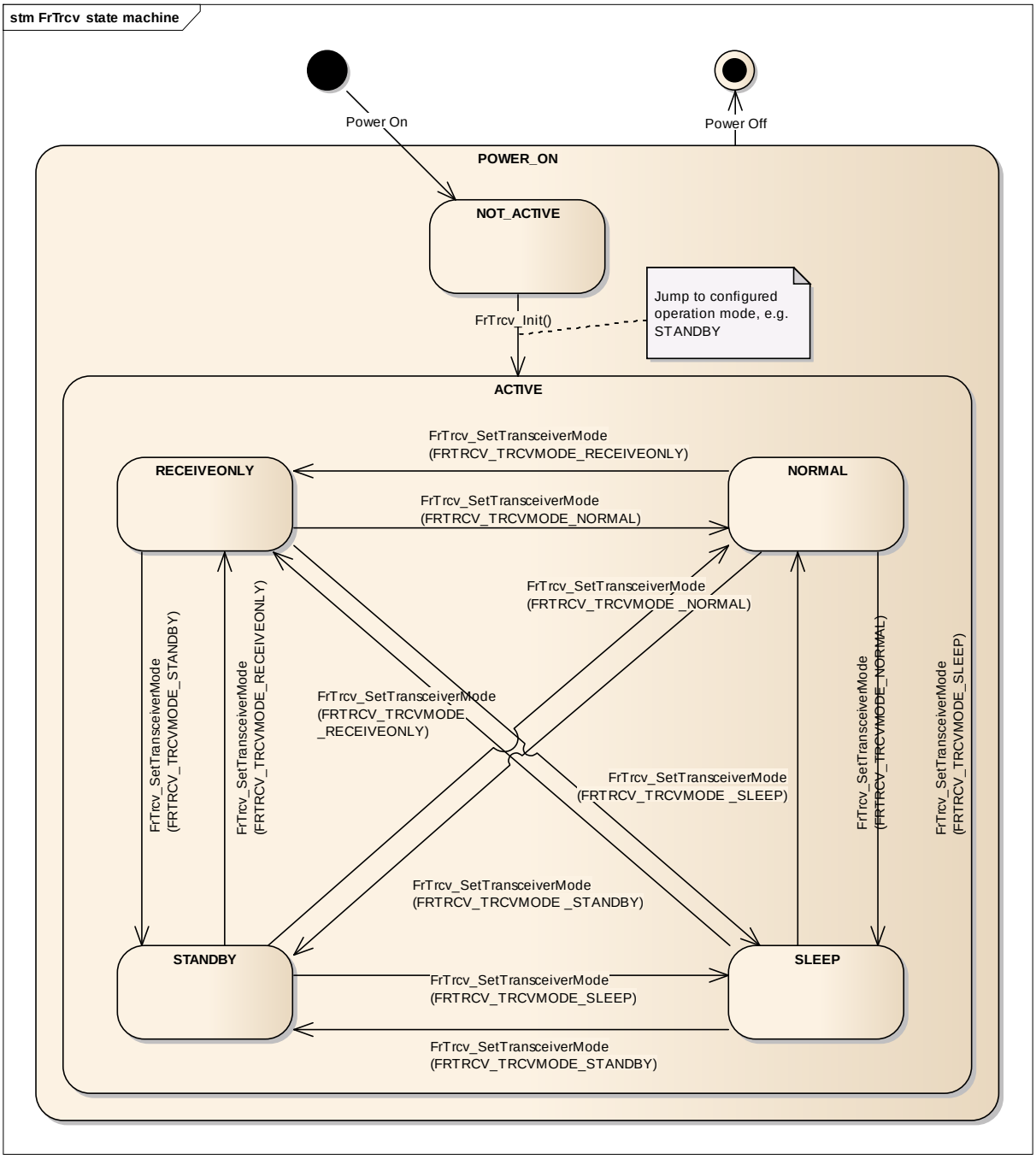
Requirement	*Description*	*Satisfied by*
MEMMAP003:	Each AUTOSAR software module shall wrap declaration and definition of code, variables and constants using the following mechanism: 1. Definition of start symbol for module memory section 2. Inclusion of MemMap.h 3. Declaration/definition of code, variables or constants belonging to the specified section 4. Definition of stop symbol for module memory section 5. Inclusion of MemMap.h The inclusion of MemMap.h within the code is a MISRA violation. As neither executable code nor symbols are included (only pragmas) this violation is an approved exception without side effects.	FrTrcv267

7 Functional specification

7.1 AUTOSAR FlexRay Transceiver Operation Mode Model

The FlexRay Transceiver operation modes are described in the state diagram below. The main idea behind this diagram is to support a lot of up to now available FlexRay Transceivers in a common model view. Depending on the transceiver device, the model may have one or two states more than necessary for a given device but this will clearly decouple the ComM and EcuM from the used hardware.

FrTrcv227:



State	*Description*
POWER_ON	**ECU is fully powered.**
NOT_ACTIVE	**State of FlexRay transceiver hardware depends on ECU hardware and on SPAL driver configuration. FlexRay Transceiver Driver is not initialized and therefore not active.**
ACTIVE	**The function FrTrcv_TrcvInit() was called. This moves FlexRay Transceiver Driver to the active state selected by configuration.**
NORMAL	**Full bus communication is possible depending on ComM state. If FlexRay transceiver hardware controls ECU power supply, ECU is fully powered. The FlexRay Transceiver Driver detects no further wake up information.**
STANDBY	**No communication is possible. ECU is still powered if FlexRay transceiver hardware controls ECU power supply. A wake up by bus or by a local wake up event is possible.**
SLEEP	**No communication is possible. ECU may be unpowered depending on responsibility to handle power supply. A wake up by bus or by a local wake up event is possible.**
RECEIVEONLY	**Similar to NORMAL, but only reception is possible.**

FrTrcv291: The FrTrcv module shall switch all covered FlexRay transceiver nodes into the state ACTIVE. In state ACTIVE each FlexRay transceiver node may be in a different sub state.
Only the states NORMAL and STANDBY are mandatory for FlexRay transceiver nodes; all other states are optional.
If a state is optional and NOT supported by the transceiver and ECU hardware (e.g. SLEEP or RECEIVEONLY), the transceiver driver substitutes an equivalent state (i.e. STANDBY instead of SLEEP; and NORMAL instead of RECEIVEONLY) and returns the real state by the FrTrcv_GetTransceiverMode() function.

7.2 FlexRay transceiver hardware operation modes

The FlexRay transceiver hardware may support more mode transitions than shown in the state diagram above. The dependencies and the recommended implementation are explained in this chapter.

7.2.1 Temporary “Go-To-Sleep” Mode

The mode often referred to as "Go-to-sleep” is a temporary mode when switching from NORMAL to (optional) SLEEP. The FlexRay transceiver driver encapsulates such a temporary mode within one of the FlexRay transceiver driver software states.

In addition, the FlexRay transceiver driver switches first from NORMAL to STANDBY and then with an additional (optional) API call from STANDBY to (optional) SLEEP.
The transition from NORMAL to STANDBY is not affected and will be performed directly.
FrTrcv352: The FlexRay transceiver driver encapsulates transient or temporary modes within one of the static optional or mandatory FlexRay transceiver driver software states.

7.2.2 "Active Star" Mode

If a transceiver supports active star mode, the driver must not incorrectly assume it is in node mode. This mode is transparent to the transceiver driver.
FrTrcv353: "Active Star" mode shall be transparent to the transceiver driver.

7.3 Wakeup Support

7.3.1 Wakeup types

There are three different scenarios related to wake up:

7.3.1.1 Scenario 1:

MCU is not powered.
Parts of ECU including FlexRay transceiver hardware are powered.
The considered FlexRay transceiver channel is in SLEEP mode.
A wake up event on FlexRay is detected by FlexRay transceiver hardware.
The FlexRay transceiver hardware causes powering of MCU.

In terms of AUTOSAR, this is kept as a cold start and NOT as a wake up.

7.3.1.2 Scenario 2:

MCU is in low power mode.
Parts of ECU including FlexRay transceiver hardware are powered.
The considered FlexRay transceiver channel is in STANDBY mode.
A wake up event on FlexRay is detected by FlexRay transceiver hardware.
The FlexRay transceiver hardware causes a SW interrupt for waking up.

In terms of AUTOSAR, this is kept as a wake up of the FlexRay channel and of the MCU.

7.3.1.3 Scenario 3:

MCU is in full power mode.
At least parts of ECU including FlexRay transceiver hardware are powered.
The considered FlexRay transceiver channel is in STANDBY mode.
A wake up event on FlexRay is detected by FlexRay transceiver hardware.

The FlexRay transceiver hardware either causes a SW interrupt for waking up or is polled cyclically for wake up events.

In terms of AUTOSAR, this is kept as a wake up of a FlexRay channel.

7.3.2 Enabling/Disabling wakeup notification

FrTrcv477: FrTrcv driver shall use the following APIs provided by Icu driver, to enable and disable the wakeup event notification:
- Icu_EnableNotification
- Icu_DisableNotification

FrTrcv480: The FlexRay Transceiver Driver shall enable/disable Icu channels only if reference is configured for the parameter FrTrcvIcuChannelRef. (see FrTrcv384).

FrTrcv478: If the reference FrTrcvIcuChannelRef (see FrTrcv384) is configured, the FlexRay Transceiver Driver shall enable the ICU channels when the transceiver transitions to Standby mode (FRTRCV_STANDBY).

FrTrcv479: If the reference FrTrcvIcuChannelRef (see FrTrcv384) is configured, the FlexRay Transceiver Driver shall disable the ICU channels when the transceiver transitions to Normal mode (FRTRCV_NORMAL).

Rationale: The FlexRay Transceiver Driver shall avoid the loss of wakeup events.

7.4 Error classification

FrTrcv107: 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.

FrTrcv050: Development error values are of type uint8.

FrTrcv085:

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service called with wrong parameter	Development	FRTRCV_E_FR_INVALID_TRCVIDX FlexRay transceiver index out of range	0x01
API Service used without initialization	Development	FRTRCV_E_FR _UNINIT	0x10
API service called in wrong transceiver operation mode	Development	FRTRCV_E_FR_TRCV_NOT_STANDBY FRTRCV_E_FR_TRCV_NOT_NORMAL FRTRCV_E_FR_TRCV_NOT_SLEEP FRTRCV_E_FR_TRCV_NOT_RECEIVEONLY	0x11 0x12 0x13 0x14
No error, all ok	Production	FRTRCV_E_FR_TRCV_NONE	*
No/incorrect	Production	FRTRCV_E_FR_NO_TRCV_CONTROL	*

communication to transceiver.			

* Assignment is done in a header file of module Dem.

7.5 Error detection

FrTrcv061: The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *FRTRCV_DEV_ERROR_DETECT* shall activate or deactivate the detection of all development errors.

[**FrTrcv048]: FrTrcv037:** If the *FRTRCV_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.3.

[FrTrcv058]: FrTrcv119: The detection of production code errors cannot be switched off.

FrTrcv237: The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness if supported by hardware.

FrTrcv354: In case of faults of the transceiver hardware, the FlexRay Transceiver Driver shall raise the production error FRTRCV_E_FR_NO_TRCV_CONTROL.

Depending on the supported transceiver device, the driver could check the correctness of the executed control communication and the operation mode of the transceiver in order to detect defective or faulty transceiver hardware and/or corrupted SPI communication.

This check only applies to errors within the transceiver or the transceiver control communication (ports or SPI), i.e. errors caused by malfunction of the µC, SW or a defect transceiver device.

FrTrcv295: The FrTrcv module shall not check errors outside of the transceiver (e.g. disturbed communication FlexRay transceiver nodes or ground offsets).

FrTrcv355: The application controller (host) has to ensure that the BD enters BD_Normal or BD_Receiveonly mode, before the CC enters one of its states where the CC starts to listen to the channel. This is ensured by the FlexRay State Manager,

FrTrcv356: In case the host commands BD_Normal and the BD does not enter BD_Normal, an error is indicated at the BD host interface. In this case the host shall force the CC to step back to a non listening state (i.e. BD_STANDY). The reason for this is that as long as the BD is not in BD_Normal or BD_Receiveonly mode, no information about the status of the channel is available via the signals RxD and RxEN.

7.6 Error notification

FrTrcv051: Detected development errors will be reported to the error hook of the Development Error Tracer (DET) if the pre-processor switch *FRTRCV_DEV_ERROR_DETECT* is set (see chapter 10).

FrTrcv052: Production errors shall be reported to Diagnostic Event Manager.

FrTrcv254: The function FrTrcv_MainFunction shall report periodically the state of the FlexRay transceiver to the Diagnostic Event Manager.

7.7 Preconditions for driver initialization

FrTrcv296: The FrTrcv module shall use drivers for SPI, Dio and or Icu to control the FlexRay bus transceiver hardware.

FrTrcv357 The environment of the FrTrcv module shall make sure that all necessary BSW drivers (used by the FrTrcv module) have been initialized and are usable before FrTrcv_Init is called.

The FlexRay bus transceiver driver uses drivers for SPI, Dio and/or Icu to control the FlexRay bus transceiver hardware.
Thus, these drivers must be available and ready to operate before the FlexRay bus transceiver driver is initialized (see FrTrcv320).

FrTrcv358 The FlexRay bus transceiver driver shall fulfill the FlexRay Transceiver hardware timing requirements also on initialization.

1. **FrTrcv359** The call of the FlexRay bus transceiver driver initialization after power up has to be performed sufficiently early in order to be able to read all necessary information out of the transceiver device in time for all other users within the ECU.
2. FrTrcv360 The runtime of the used underlying services used shall very short and synchronous to enable the driver to keep its own timing requirements limited by the hardware device used (FrTrcv231).
3. FrTrcv361 The FlexRay Transceiver Driver runtime shall support setup and hold times of the FlexRay Transceiver Hardware devices in all states including low power states, e.g. sleep.

7.8 Instance concept

An ECU may contain multiple FlexRay transceivers. These transceivers can be of different types. Each transceiver type is handled by a dedicated FlexRay Transceiver Driver.

For your convenience, assume that any API call is not executed directly but is resolved by configuration to a zero based index into a function pointer table (per driver).
This issue is already resolved for Flexray Interface FrIf and the FlexRay communication controller.

FrTrcv226:Multiple FlexRay transceivers of the same type are handled by a single FlexRay transceiver driver;

There is no need for multiple instances of this single FlexRay transceiver driver.

FrTrcv supports exactly one transceiver per CC and channel (i.e., it is not permitted that two CCs of one ECU share one FlexRay transceiver)!

8 API specification

8.1 Imported types

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

FrTrcv321:

Module	*Imported Type*
ComStack_Types	BusTrcvErrorType
Dem	Dem_EventIdType
Dio	Dio_ChannelType
	Dio_LevelType
	Dio_PortLevelType
	Dio_PortType
	Dio_ChannelGroupType
EcuM	EcuM_WakeupSourceType
Icu	Icu_ChannelType
Spi	Spi_ChannelType
	Spi_DataType
	Spi_NumberOfDataType
	Spi_SequenceType
	Spi_StatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

FrTrcv030: FrTrcv045: FrTrcv069: FrTrcv071:

The type definitions FrTrcv_TrcvModeType and FrTrcv_TrcvWUReasonType shall be kept in a file named Fr_GeneralTypes.h and be protected by a FR_GENERAL_TYPES define in order:
- to be shared between different FlexRay Transceiver Drivers
- to be included into the FrIf

If different FlexRay Transceiver Drivers are used, only one instance of this file has to be included in the source tree.

8.2.1 FrTrcv_TrcvModeType

FrTrcv252: FrTrcv048:

Name:	`FrTrcv_TrcvModeType`	
Type:	`Enumeration`	
Range:	`FRTRCV_TRCVMODE_NORMAL`	Transceiver is in state NORMAL
	`FRTRCV_TRCVMODE_STANDBY`	Transceiver is in state STANDBY
	`FRTRCV_TRCVMODE_SLEEP`	Transceiver is in state SLEEP
	`FRTRCV_TRCVMODE_RECEIVEONLY`	Transceiver is in state RECEIVEONLY
Description:	Transceiver modes in state ACTIVE.	

8.2.2 FrTrcv_TrcvWUReasonType

FrTrcv074:

Name:	FrTrcv_TrcvWUReasonType	
Type:	Enumeration	
Range:	FRTRCV_WU_NOT_SUPPORTED	The transceiver does not support any information for the wake up reason.
	FRTRCV_WU_BY_BUS	The transceiver has detected that the bus has caused the wake up of the ECU.
	FRTRCV_WU_INTERNALLY	The transceiver has detected that the bus has woken up by the ECU via FrTrcv_SetTransceiverMode() API call
	FRTRCV_WU_RESET	The transceiver has detected that the “wake up” is due to an ECU reset.
	FRTRCV_WU_POWER_ON	The transceiver has detected that the “wake up” is due to an ECU reset after power on.
Description:	This type to be used to specify the wake up reason detected by the FR transceiver in detail.	

8.3 Function definitions

FrTrcv043: FrTrcv089: FrTrcv090: FrTrcv091: FrTrcv092: FrTrcv093: FrTrcv094: FrTrcv104:

8.3.1 FrTrcv_TrcvInit

FrTrcv322:

Service name:	FrTrcv_TrcvInit	
Syntax:	`void FrTrcv_TrcvInit( uint8 FrTrcv_TrcvIdx )`	
Service ID[hex]:	0x00	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This service initializes the FrTrcv.	

FrTrcv008: FrTrcv263:FrTrcv228FrTrcv230:FrTrcv112:

FrTrcv270: The function FrTrcv_TrcvInit shall set the transceiver identified by the parameter FrTrcv_TrcvIdx in the state defined by the configuration parameter FRTRCV_INIT_STATE, i.e. either in state FRTRCV_TRCVMODE_NORMAL,

FRTRCV_TRCVMODE_STANDBY, FRTRCV_TRCVMODE_RECEIVEONLY or FRTRCV_TRCVMODE_SLEEP.

Note that in the time span between power up and the call FrTrcv_TrcvInit the FlexRay transceiver hardware may be in a different state. This depends on hardware and SPAL driver configuration.

The initialization sequence after reset (e.g. power up) is a critical phase for the FlexRay transceiver driver.

FrTrcv320: The FrTrcv module's environment shall make sure that all SPAL drivers that are used by the FrTrcv module to access the transceiver hardware, are initialized and usable before FrTrcv_TrcvInit is called.

FrTrcv268: In case of a fault during transceiver access, the function FrTrcv_TrcvInit shall raise the production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237).

FrTrcv269: The function FrTrcv_TrcvInit shall check whether there has been a wake up due to transceiver activity and report this to the EcuM via EcuM_SetWakeupEvent(event).

Hint:
Every FlexRay conformant transceiver is able to detect wakeup!

FrTrcv362 The driver has to notify ECU State Manager by invoking the EcuM_SetWakeupEvent service once when a wakeup event is detected.

FrTrcv363 The driver has to notify ECU State Manager by invoking the EcuM_SetWakeupEvent service during initialization of the driver if a pending wakeup event is detected during the initialization.

FrTrcv364 The driver shall notify ECU State Manager of wakeup events indirectly via FrTrcv_Cbk_WakeupByTransceiver.

FrTrcv365: Drivers which serve wakeup sources must be re-initialized in the restart block.

FrTrcv366: The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back.

FrTrcv367: The driver shall support a wakeup ISR if supported by hardware.

FrTrcv: The FlexRay Transceiver Driver need not support wakeup validation if this is done by FlexRay Transceiver Hardware.

FrTrcv271: If development error detection of the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is not within the allowed range, the function

FrTrcv_TrcvInit shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX and return without any further action.

8.3.2 FrTrcv_SetTransceiverMode

FrTrcv323:

Service name:	FrTrcv_SetTransceiverMode	
Syntax:	`BusTrcvErrorType FrTrcv_SetTransceiverMode( uint8 FrTrcv_TrcvIdx, FrTrcv_TrcvModeType FrTrcv_TrcvMode )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
	FrTrcv_TrcvMode	Selects the state the transceiver will transit to (transitions to optional states may fail)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	BusTrcvErrorType	BUSTRCV_E_OK: will be returned if the transceiver state has been changed to the requested mode. BUSTRCV_E_ERROR: will be returned if the transceiver state change has failed or the parameter is out of the allowed range. The previous state has not been changed.
Description:	This service returns the transceiver mode.	

FrTrcv252 FrTrcv064:

FrTrcv272: The function FrTrcv_SetTransceiverMode shall switch the internal state of the transceiver identified by FrTrcv_TrcvIdx to the state indicated by FrTrcv_TrcvMode.

FrTrcv273: The function FrTrcv_SetTransceiverMode shall return BUSTRCV_E_ERROR and doesn't change the current state if an illegal transition is requested.

According to [5] every FlexRay Transceiver has to support two mandatory states: FrTRCV_TRCVMODE_STANDBY and FrTRCV_TRCVMODE_NORMAL; all other states are optional.

FrTrcv274: if an optional state (other than NORMAL and STANDBY) is NOT supported by the transceiver and ECU hardware, the function FrTrcv_SetTransceiverMode shall switch to an equivalent state (e.g., FrTRCV_TRCVMODE_SLEEP or FrTRCV_TRCVMODE_RECEIVEONLY not supported -> FrTRCV_TRCVMODE_STANDBY instead of FrTRCV_TRCVMODE_SLEEP; and FrTRCV_TRCVMODE_NORMAL instead of FrTRCV_TRCVMODE_RECEIVEONLY).

FrTrcv278: In case of a fault during transceiver access, the function FrTrcv_SetTransceiverMode shall raise production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237) and return BUSTRCV_E_ERROR.

FrTrcv368: The API function calls to the FlexRay Transceiver Driver shall be synchronuous.

FrTrcv275: If development error detection for the module FrTrcv is enabled: If the parameter FrTrcv_TrcvIdx is not within the allowed range, the function FrTrcv_SetTransceiverMode shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX and return BUSTRCV_E_ERROR.

FrTrcv383: The FlexRay Driver shall avoid illegal state transitions.

FrTrcv276: If development error detection for the module FrTrcv is enabled: If the mode transition fails, the function FrTrcv_SetTransceiverMode shall raise the following development error and return BUSTRCV_E_ERROR:
FRTRCV_E_FR_TRCV_NOT_STANDBY: Transition to FRTRCV_TRCVMODE_STANDBY failed
FRTRCV_E_FR_TRCV_NOT_NORMAL: Transition to FRTRCV_TRCVMODE_NORMAL failed
FRTRCV_E_FR_TRCV_NOT_SLEEP: Transition to FRTRCV_TRCVMODE_SLEEP failed
FRTRCV_E_FR_TRCV_NOT_RECEIVEONLY: Transition to FRTRCV_TRCVMODE_RECEIVEONLY failed

FrTrcv277: If development error detection for the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_SetTransceiverMode shall raise development error FRTRCV_E_FR_UNINIT and return BUSTRCV_E_ERROR.

8.3.3 FrTrcv_GetTransceiverMode

FrTrcv324:

Service name:	FrTrcv_GetTransceiverMode	
Syntax:	`BusTrcvErrorType FrTrcv_GetTransceiverMode( uint8 FrTrcv_TrcvIdx, FrTrcv_TrcvModeType* FrTrcv_TrcvModePtr )`	
Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	FrTrcv_TrcvModePtr	Pointer to structure of current transceiver state; the FlexRay transceiver driver will write the transceiver state information there.
Return value:	BusTrcvErrorType	BUSTRCV_E_OK: will be returned if the transceiver state

		has been provided BUSTRCV_E_ERROR: will be returned if the transceiver state is illegal or the parameter is out of range. Output parameters remain unchanged.
Description:	This function returns the actual state of the transceiver.	

FrTrcv253: The function FrTrcv_GetTransceiverMode shall return the state of the transceiver identified by FrTrcv_TrcvIdx.

FrTrcv281: In case of a fault during transceiver access, the function FrTrcv_GetTransceiverMode shall raise production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237) and return BUSTRCV_E_ERROR.

See FrTrcv_TrcvInit for the provided state after the FlexRay transceiver driver initialization until the first operation mode change request.

If a transceiver supports active star mode, do NOT incorrectly assume it is in node mode.

The number of supported FlexRay transceiver nodes and the transceiver type for each FlexRay transceiver node is statically set in the configuration phase.

FrTrcv279: If development error detection for the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_GetTransceiverMode shall raise the development error FRTRCV_E_FR_INVALID_TRCVIDX and return BUSTRCV_E_ERROR.

FrTrcv280: If development error detection for the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_GetTransceiverMode shall raise the development error FRTRCV_E_FR_UNINIT and return BUSTRCV_E_ERROR.

8.3.4 FrTrcv_GetTransceiverWUReason

FrTrcv325:

Service name:	FrTrcv_GetTransceiverWUReason	
Syntax:	`BusTrcvErrorType FrTrcv_GetTransceiverWUReason( uint8 FrTrcv_TrcvIdx, FrTrcv_TrcvWUReasonType* FrTrcv_TrcvWUReasonPtr )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	FrTrcv_TrcvWUReasonPtr	Pointer to structure of least recent wakeup source, the

		FlexRay transceiver driver will write the transceiver state information there
Return value:	BusTrcvErrorType	BUSTRCV_E_OK: will be returned if the transceiver wake up source has been provided BUSTRCV_E_ERROR: will be returned if the transceiver state is illegal or the parameter is out of range. Output parameters remain unchanged.
Description:	This function returns the wakeup reason.	

FrTrcv232: The function FrTrcv_GetTransceiverWUReason shall return the reason for the wake up that the FlexRay transceiver identified by FrTrcv_TrcvIdx has detected.
The ability to detect and differentiate the possible wake up reasons depends strongly on the FlexRay transceiver hardware.

FrTrcv284: In case of a fault during transceiver access, the function FrTrcv_GetTransceiverWUReason shall raise production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237) and return BUSTRCV_E_ERROR.

Please be aware, that if more than one bus is available each bus may report a different wake up reason. E.g. if an ECU has FlexRay, a wake up by FlexRay may occur and the incoming data may cause an internal wake up for another FlexRay bus.
The FlexRay bus transceiver driver has a “per bus” view and does not vote the more important reason or sequence internally. The same may be true if e.g. one transceiver controls the power supply and the other is just powered or un-powered. Then one may be able to return “FRTRCV_WU_POWER_ON” whereas the other may state e.g. “FRTRCV_WU_RESET”.
It is up to the EcuM and the ComM, to decide what shall happen with that wake up information.

FrTrcv282: If development error detection of the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_GetTransceiverWUReason shall raise the development error FRTRCV_E_FR_INVALID_TRCVIDX and return BUS_TRCV_E_ERROR.

FrTrcv283: If development error detection of the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_GetTransceiverWUReason shall raise the development error FRTRCV_E_FR_UNINIT and return BUS_TRCV_E_ERROR.

8.3.5 FrTrcv_GetVersionInfo

FrTrcv326:

Service name:	FrTrcv_GetVersionInfo
Syntax:	`void FrTrcv_GetVersionInfo(` `  Std_VersionInfoType* versioninfo`

	)	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to structure with version information.
Return value:	None	
Description:	This service returns the version information of this module.	

FrTrcv001: FrTrcv105: FrTrcv109:

FrTrcv285: The function FrTrcv_GetVersionInfo shall return the version information of the FrTrcv module, NOT the version of the FlexRay transceiver hardware.

FrTrcv339: The function FrTrcv_GetVersionInfo shall be pre compile time configurable On/Off by the configuration parameter: FRTRCV_GET_VERSION_INFO

FrTrcv338: If source code for caller and callee of FrTrcv_GetVersionInfo is available, the FrTrcv module should realize FrTrcv_GetVersionInfo as a macro, defined in the module's header file.

8.3.6 FrTrcv_DisableTransceiverWakeup

FrTrcv327:

Service name:	FrTrcv_DisableTransceiverWakeup	
Syntax:	`BusTrcvErrorType FrTrcv_DisableTransceiverWakeup( uint8 FrTrcv_TrcvIdx )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	BusTrcvErrorType	BUSTRCV_E_OK: will be returned if the transceiver wake up has been disabled BUSTRCV_E_ERROR: will be returned if the transceiver state is illegal or the parameter is out of range. Wake up state remains unchanged.
Description:	This function disables the notification for wake up events on the addressed bus.	

FrTrcv246: FrTrcv235:

FrTrcv286: The function FrTrcv_DisableTransceiverWakeup shall disable the notification for wake up events“through the transceiver identified by FrTrcv_TrcvIdx, FrTrcv369 (After the call of FrTrcv_DisableTransceiverWakeup the FrTrcv module shall execute no wake up notifications for the node referenced till FrTrcv_EnableTransceiverWakeup is executed).

In order to keep the transceiver driver simple, this API refers to all kinds of wake up. Further differentiation of wakeup sources requires knowledge available only to higher software layers and is out of scope of the transceiver driver.

FrTrcv287: The module FrTrcv shall not lose wake up events during the disabled period.

In order to realize FrTrcv287 and if necessary by the transceiver device and the underlying communication, the driver has to detect the wake up event during the disabled period and store it internally to raise the event when the wake up notification is enabled again.

FrTrcv290: In case of a fault during transceiver access, the function FrTrcv_DisableTransceiverWakeup shall raise the production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237) and return BUSTRCV_E_ERROR.

FrTrcv288: If development error detection of the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_DisableTransceiverWakeup shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX and return BUSTRCV_E_ERROR.

FrTrcv289: If development error detection of the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_DisableTransceiverWakeup shall raise development error FRTRCV_E_FR_UNINIT and return BUSTRCV_E_ERROR.

8.3.7 FrTrcv_EnableTransceiverWakeup

FrTrcv328:

Service name:	FrTrcv_EnableTransceiverWakeup	
Syntax:	`BusTrcvErrorType FrTrcv_EnableTransceiverWakeup( uint8 FrTrcv_TrcvIdx )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	

Return value:	BusTrcvErrorType	BUSTRCV_E_OK: will be returned if the transceiver wake up has been enabled BUSTRCV_E_ERROR: will be returned if the transceiver state is illegal or the parameter is out of range. Wake up state remains unchanged.
Description:	This function enables the notification for wake up events on the addressed bus.	

FrTrcv245: The function FrTrcv_EnableTransceiverWakeup shall enable the notification for wake up events through the transceiver identified by FrTrcv_TrcvIdx FrTrcv370: After the call of the API the FrTrcv module shall perform a wake up notification to higher layers when a wake up event is detected.

In order to keep the transceiver driver simple, this API refers to all kinds of wake up. Further differentiation of wakeup sources requires knowledge available only to higher software layers and is out of scope of the transceiver driver.

It is very important not to lose wake up events during the disabled period.

FrTrcv300: If the FrTrcv module has a stored wake up event pending, the FrTrcv module shall execute the notifications for the stored wakeup event within the function FrTrcv_EnableTransceiverWakeup or immediately after (depending on the implementation).

The implementation may be e.g. enabling the interrupt source for the wake up. If the interrupt is level triggered a pending interrupt is automatically stored and raised after enabling the notification again.

FrTrcv303: In case of a fault during transceiver access, the function FrTrcv_EnableTransceiverWakeup shall raise the production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237) and return BUSTRCV_E_ERROR.

FrTrcv301: If development error detection for the FrTrcv module is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_EnableTransceiverWakeup shall raise the development error code FRTRCV_E_FR_INVALID_TRCVIDX and return BUS_TRCV_E_ERROR.

FrTrcv302: If development error detection for the FrTrcv module is enabled: if the transceiver has not been initialized, the function FrTrcv_EnableTransceiverWakeup shall raise the development error code FRTRCV_E_FR_UNINIT and return BUSTRCV_E_ERROR.

8.3.8 FrTrcv_ClearTransceiverWakeup

FrTrcv329:

Service name:	FrTrcv_ClearTransceiverWakeup

Syntax:	`BusTrcvErrorType FrTrcv_ClearTransceiverWakeup(`
	`  uint8 FrTrcv_TrcvIdx`
	`)`

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	BusTrcvErrorType	BUSTRCV_E_OK: will be returned if the transceiver wakup source has been cleared BUSTRCV_E_ERROR: will be returned if the transceiver could not clear its wakeup state or the parameter is out of range. Wake up state remains unchanged.
Description:	This function clears a pending wake up event.	

FrTrcv247: The function FrTrcv_ClearTransceiverWakeup shall clear a pending wake up event on the transceiver identified by FrTrcv_TrcvIdx.

FrTrcv371: The API shall clear all pending wake up events under control of the higher layer .
It may be used if the wake up notification is disabled.

In order to keep the transceiver driver simple, this API refers to all kinds of wake up. Further differentiation of wakeup sources requires knowledge available only to higher software layers and is out of scope of the transceiver driver.

FrTrcv306: In case of a fault during transceiver access, the function FrTrcv_ClearTransceiverWakeup shall raise production error FRTRCV_E_FR_NO_TRCV_CONTROL (see also FrTrcv237) and return BUSTRCV_E_ERROR.

FrTrcv304: If development error detection is enabled for the module FrTrcv: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_ClearTransceiverWakeup shall raise the development error code FRTRCV_E_FR_INVALID_TRCVIDX and return BUSTRCV_E_ERROR.

FrTrcv305: If development error detection is enabled for the module FrTrcv: if the transceiver has not been initialized, the function FrTrcv_ClearTransceiverWakeup shall raise the development error code FRTRCV_E_FR_UNINIT and return BUSTRCV_E_ERROR.

8.4 Scheduled functions

This section lists functions that are directly called by Basic Software Scheduler.

8.4.1 FrTrcv_MainFunction

FrTrcv330:

Service name:	FrTrcv_MainFunction

Syntax:	`void FrTrcv_MainFunction(`
	
	`)`

Service ID[hex]:	0x0d
Timing:	FIXED_CYCLIC
Description:	--

FrTrcv020: FrTrcv072: FrTrcv075: **FrTrcv126:**

The FlexRay bus transceiver driver may have cyclic jobs like polling for wake up events (if configured).

FrTrcv340: The function FrTrcv_MainFunction shall scan all busses in STANDBY and SLEEP for wake up events and shall perform these events by calling appropriate call-back functions.

FrTrcv122: The function FrTrcv_MainFunction shall be implemented in such a way that it can run inside a basic task (according to AUTOSAR OS classification).

FrTrcv372: The BSW scheduler shall execute FrTrcv_MainFunction with a period configured by the parameter “FRTRCV_MAIN_FUNCTION_CYCLE_TIME”
FrTrcv373: For a cycle time of 0 this function is never executed and need not be present in compiled code. See chapter 10.2.4 for more details.

FrTrcv309: The function FrTrcv_MainFunction shall raise the production error FRTRCV_E_FR_TRCV_NONE if all transceivers are ok (no transceiver error).

FrTrcv308: If development error detection of the module FrTrcv is enabled: if any of the configured transceivers is not initialized, the function FrTrcv_MainFunction shall raise development error FRTRCV_E_FR_UNINIT.

8.5 Call-back notifications

This is a list of functions provided for lower layer modules.

8.5.1 FrTrcv_Cbk_WakeupByTransceiver

FrTrcv331:

Service name:	FrTrcv_Cbk_WakeupByTransceiver	

Syntax:	`void FrTrcv_Cbk_WakeupByTransceiver(`
	`  uint8 FrTrcv_TrcvIdx`
	`)`	

Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied
Parameters (inout):	None	

Parameters (out):	None
Return value:	None
Description:	--

FrTrcv233: FrTrcv086: FrTrcv262:

FrTrcv310: The Icu driver shall call the callback function FrTrcv_Cbk_WakeupByTransceiver() via the FlexRay Interface (FrIf) in case a wake up interrupt is detected or periodically by a polling process.

FrTrcv311: The function FrTrcv_Cbk_WakeupByTransceiver() shall call the appropriate function of EcuM (EcuM_SetWakeupEvent) with the parameter value ECUM_WKSOURCE_FRTRCV_FR of EcuM_WakeupSourceType only in case a valid wakeup originated from the transceiver identified by FrTrcv_TrcvIdx.

Thus, shared interrupts are easily de-multiplexed: Drivers which did not trigger the interrupt just return doing nothing.

FrTrcv374: The function FrTrcv_Cbk_WakeupByTransceiver() shall clear a pending wake up event on the transceiver identified by FrTrcv_TrcvIdx after the last call of EcuM (EcuM_SetWakeupEvent).

Wake up by bus is always asynchronous to the transition to sleep and standby. In worst case wake up occurs during transition to sleep.

FrTrcv375: The FlexRay Transceiver Driver shall check for wake up events immediately after the API call FrTrcv_SetTransceiverMode

FrTrcv376 The EcuM shall be able to handle the wake up event immediately after requesting the standby or sleep mode.

FrTrcv377 Drivers which serve wakeup sources must be re-initialized in the restart block.

FrTrcv: The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back.

FrTrcv378 If no wake up by bus is used this function need not be present in compiled code.
See configuration parameters FRTRCV_WAKEUP_BY_NODE_USED in chapter 8.6.2 for more details.

FrTrcv379 Calling FrTrcv_Cbk_WakeupByTransceiver in an interrupt context shall be supported. This has to be documented (BSW00333).
FrTrcv380: Calling FrTrcv_Cbk_WakeupByTransceiver by a polling process in sleep mode or by FrTrcv_MainFunction(). Shall be supported

FrTrcv312: If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_Cbk_WakeupByTransceiver shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX.

FrTrcv313: If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_Cbk_WakeupByTransceiver shall raise development error FRTRCV_E_FR_UNINIT.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

FrTrcv332:

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

8.6.2 Optional Interfaces

FrTrcv019: This chapter defines all interfaces that are required to fulfill an optional functionality of the module.

FrTrcv334:

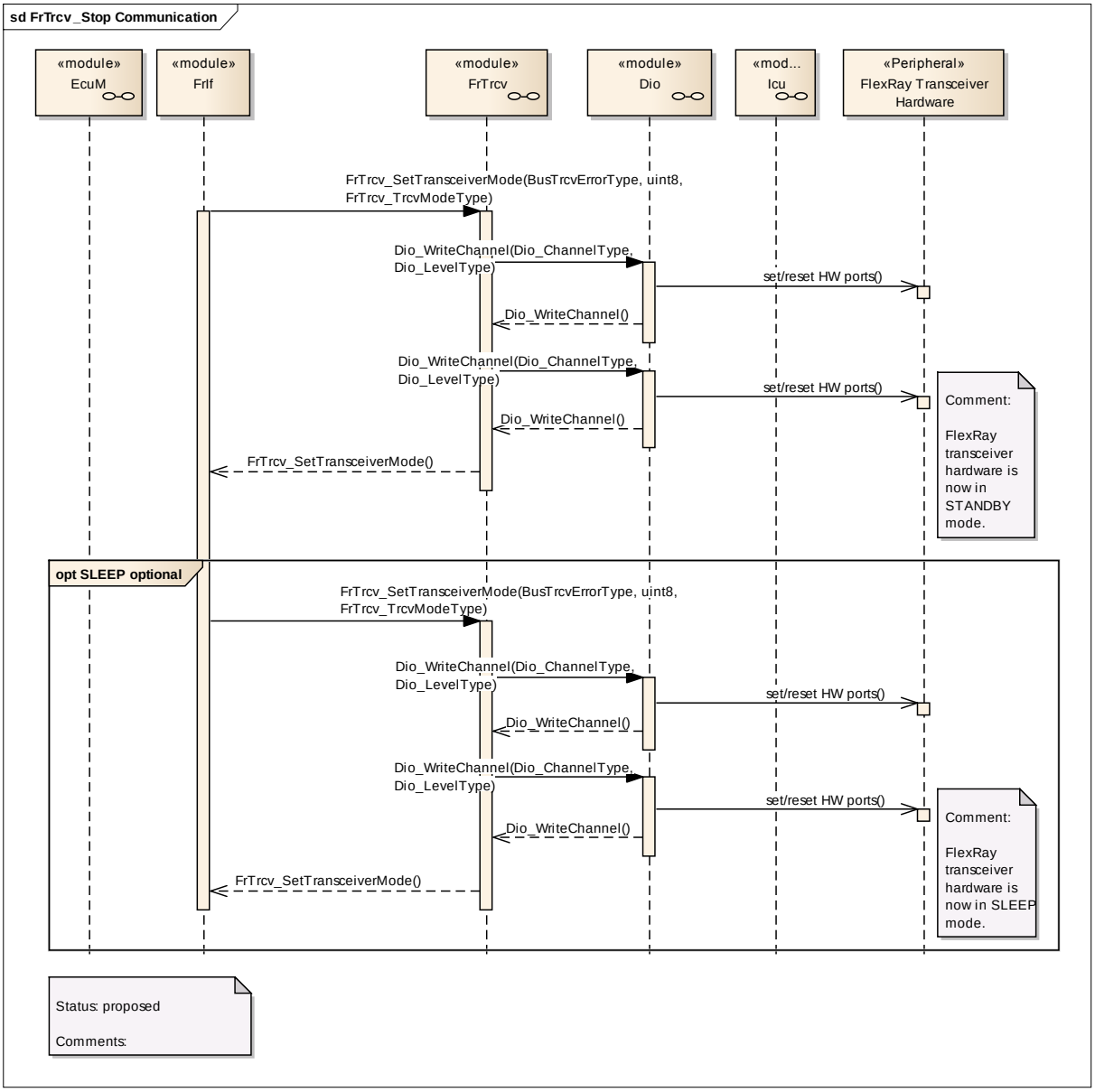
API function	*Description*
Det_ReportError	Service to report development errors.
Dio_ReadChannel	Returns the value of the specified DIO channel.
Dio_ReadChannelGroup	This Service reads a subset of the adjoining bits of a port.
Dio_ReadPort	Returns the level of all channels of that port.
Dio_WriteChannel	Service to set a level of a channel.
Dio_WriteChannelGroup	Service to set a subset of the adjoining bits of a port to a specified level.
Dio_WritePort	Service to set a value of the port.
EcuM_SetWakeupEvent	Sets the wakeup event.
Icu_DisableNotification	This function disables the notification of a channel.
Icu_EnableNotification	This function enables the notification on the given channel.
Spi_GetStatus	Service returns the SPI Handler/Driver software module status.
Spi_ReadIB	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetupEB	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Service to transmit data on the SPI bus
Spi_WriteIB	Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

Interfaces of SPI module are used if there are instances of the container FlexRayTransceiverSPISequences.
Interfaces of DIO module are used if there are instances of the container FlexRayTransceiverDioAccess.
ATTENTION: Either SPI or DIO must be supported depending on FlexRay Transceiver hardware.

8.6.3 Configurable interfaces

There are no configurable interfaces for FlexRay transceiver driver.

9 Sequence diagrams



ATTENTION: Sequence charts are application examples only. They focus on interaction between the FlexRay transceiver driver (FrTrcv), FlexRay Interface (FrIf) and BSW modules ComM, EcuM, Icu and Dio. For details, see [14] and [15]. Depending on FlexRay transceiver hardware, one or more calls to Dio_WriteChannels may be necessary.
For details on FlexRay Transceiver wakeup please refer to chapter 9 of [14].

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.

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

Chapter 10.3 specifies published information of the module FrTrcv.

10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture
- AUTOSAR ECU Configuration Specification
  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

FrTrcv095:
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

FrTrcv087: 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.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapters 7 and Chapter 8.

10.2.1 Variants

FrTrcv314: Variant 1: Only pre compile time parameters.
FrTrcv315: Variant 2: Mix of pre compile- and link time parameters (unused in this driver).
FrTrcv316: Variant 3: Mix of pre compile-, link time and post build time parameters (unused in this driver).

FrTrcv317: The FrTrcv module shall support pre compile time configuration.

FrTrcv318: The FrTrcv module shall not use link time parameters within the FlexRay Transceiver Driver.

FrTrcv319: The FrTrcv module shall not use post build time configuration changes by flashing within the FlexRay Transceiver Driver.

10.2.2 General configuration requirements

All following configuration is provided by a configuration tool. Configuration information is part of files FrTrcv.h and FrTrcv_Cfg.c.

Requirement	*Description*
FrTrcv010:	A configuration tool is used to generate the configuration data and code if any.
FrTrcv011:	[BSW00160] Human-readable configuration data
FrTrcv018:	[BSW00170] Data for reconfiguration of AUTOSAR SW-Components
FrTrcv023:	[BSW00302] Limit exported information
FrTrcv225:	[BSW05131] Configuration Data for FlexRay Transceiver
	[BSW05132] Support for More than One FlexRay Transceiver
FrTrcv016:	The configuration tool has to check the validity of the provided input data and the usability in the project context.
FrTrcv080	Provide configuration dependency information.

FrTrcv088:

10.2.3 FrTrcv

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

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrTrcvGeneral	1	Container gives FlexRay transceiver driver basic information.
FrTrcvNode	1..*	Container gives FlexRay transceiver driver information about a single FlexRay transceiver node. Any FlexRay transceiver driver has such FlexRay transceiver nodes.

10.2.4 FrTrcvGeneral

SWS Item	**FrTrcv055 :**
Container Name	FrTrcvGeneral{FlexRayTransceiverDriverBasic}
Description	Container gives FlexRay transceiver driver basic information.
Configuration Parameters	

SWS Item	**FrTrcv341 :**		
Name	FrTrcvDevErrorDetect {FRTRCV_DEV_ERROR_DETECT}		
Description	Switches development error detection and notification on and off. If switched on, #define FRTRCV_DEV _ERROR_DETECT ON shall be generated. If switched off, #define FRTRCV_DEV_ERROR _DETECT OFF shall be generated. Define shall be part of file FrTrcv_Cfg.h.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv342 :**		
Name	FrTrcvGetVersionInfo {FRTRCV_GET_VERSION_INFO}		
Description	Switches version information API on and off. If switched off, function need not be present in compiled code.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv268 :**		
Name	FrTrcvIndex		
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	**FrTrcv343 :**
Name	FrTrcvMainFunctionCycleTime {FRTRCV_WAKEUP_POLLING}
Description	Cyclic call time for function FrTrcvMainFunction in seconds. A call time of 0ms indicates no calls for this function. In this case function need not be present in compiled code.
Multiplicity	1

Type	FloatParamDef		
Range	-INF .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: FRTRCV_MAIN_FUNCTION_CYCLE_TIME		

SWS Item	**FrTrcv344 :**		
Name	FrTrcvReceiveonlySupport {FRTRCV_RECEIVEONLY_SUPPORT}		
Description	Information if the optional transceiver state RECEIVEONLY is supported by the driver and hardware.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv345 :**		
Name	FrTrcvSleepSupported {FRTRCV_SLEEP_SUPPORTED}		
Description	Information if the optional transceiver state SLEEP is supported by the driver and hardware.		
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

10.2.5 FrTrcvNode

SWS Item	**FrTrcv091 :**
Container Name	FrTrcvNode{FlexRayTransceiverNode}
Description	Container gives FlexRay transceiver driver information about a single FlexRay transceiver node. Any FlexRay transceiver driver has such FlexRay transceiver nodes.
Configuration Parameters	

SWS Item	**FrTrcv346 :**		
Name	FrTrcvControlsPowerSupply {FRTRCV_CONTROLS_POWER_SUPPLY}		
Description	Is ECU power supply controlled by this transceiver?		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv347 :**		
Name	FrTrcvInitState {FRTRCV_INIT_STATE}		
Description	State of FlexRay transceiver after power on. ImplementationType: FrTrcv_TrcvModeType		
Multiplicity	1		
Type	EnumerationParamDef		
Range	FRTRCV_TRCVMODE_NORMAL	Normal mode	
	FRTRCV_TRCVMODE_RECEIVEONLY	Receive only mode	
	FRTRCV_TRCVMODE_SLEEP	Sleep mode	
	FRTRCV_TRCVMODE_STANDBY	Stand by mode	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv348 :**		
Name	FrTrcvMaxBaudrate {FRTRCV_MAX_BAUDRATE}		
Description	Max baudrate for transceiver hardware type. Only used for validation purposes. Value shall be configured by configuration tool based on FRTRCV_HARDWARE_NAME and internal information about ability of this hardware typel.		
Multiplicity	1		
Type	IntegerParamDef		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv349 :**		
Name	FrTrcvNodeId {FRTRCV_NODE_ID}		
Description	Unique node id. It is used by ComM by Icu and internally.		
Multiplicity	1		
Type	IntegerParamDef (Symbolic Name generated for this parameter)		
Range	..		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv350 :**
Name	FrTrcvWakeupByNodeUsed {FRTRCV_WAKEUP_BY_NODE_USED}
Description	Is wake up by node supported? If FlexRay transceiver hardware does not support wake up by node value is always FALSE. If FlexRay transceiver hardware supports wake up by node value is TRUE or FALSE depending whether it is used or not.
Multiplicity	1
Type	BooleanParamDef
Default value	--

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

SWS Item	**FrTrcv351 :**		
Name	FrTrcvWakeupPolling {FRTRCV_WAKEUP_POLLING}		
Description	Information of this entitiy is only relevant if parameter FrTrcvWakeup-ByTransceiverUsed is TRUE. Information whether wake up events will be performed in main function (TRUE) or by a callback function called by an interrupt service (FALSE).		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance dependency: FRTRCV_WAKEUP_BY_NODE_USED, FRTRCV_MAIN_FUNCTION_CYCLE_TIME		

SWS Item	**FrTrcv384 :**		
Name	FrTrcvIcuChannelRef {FRTRCV_ICU_CHANNEL_REF}		
Description	Reference to the IcuChannel to enable/disable the interrupts for wakeups.		
Multiplicity	0..1		
Type	Reference to [IcuChannel]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**FrTrcv269 :**		
Name	FrTrcvWakeupSourceRef {FRTRCV_WAKEUP_SOURCE_REF}		
Description	Reference to a wakeup source in the EcuM configuration. This reference is only needed if FrTrcvWakeupByNodeUsed is true. Implementation Type: reference to EcuM_WakeupSourceType		
Multiplicity	1		
Type	Reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: FrTrcvWakeupByNodeUsed		

No Included Containers

10.3 Published Information

FrTrcv101: Published information contains data defined by the implementer of the SW module that does not change when the module is adapted (i.e. configured) to the actual HW/SW environment. It thus contains version and manufacturer information.

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 3.2 Figure 4.1 and Figure 7.1).

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