Document Title	Specification of FlexRay Transceiver Driver
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	074
Document Classification	Standard
Document Version	1.7.1
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	1.7.1	AUTOSAR Release Management	• Adapted requirement identifier prefixes • Deleted some redundant software specification items
31.10.2013	1.7.0	AUTOSAR Release Management	• Simplified schedule to pre compile fixed cyclic • Reduced run time configuration checks • Editorial changes • Removed chapter(s) on change documentation
26.02.2013	1.6.0	AUTOSAR Administration	• Reworked according to the new SWS_BSWGeneral
21.11.2011	1.5.0	AUTOSAR Administration	• Improved interrupt support by ICU • Improved production error concept
28.10.2010	1.4.0	AUTOSAR Administration	• Support of local wake up • Timing based on OS timer references • Support of error handling by Complex Drivers • Fixed constraints of configuration parameters • Removed APIs FrTrcv_EnableTransceiverWakeup and FrTrcv_DisableTransceiverWakeup

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
30.11.2009	1.3.0	AUTOSAR Administration	• Active star support added: a) Provided three new APIs: FrTrcv_GetTransceiverError(), FrTrcv_DisableTransceiverBranch(), FrTrcv_EnableTransceiverBranch()) b) Active stars can now be accessed as a single FlexRay transceiver by the FlexRay state manager via the FlexRay interface. c) Configuration support of of branches of active stars by FrTrcvBranchIdContainer d) Active stars can now be accessed as a single FlexRay transceiver by the FlexRay state manager via the FlexRay interface: • API FrTrcv_Cbk_WakeupByTransceiver has been renamed to FrTrcv_CheckWakeupByTransceiver • Legal disclaimer revised
30.01.2008	1.2.1	AUTOSAR Administration	• Chapter 9 regenerated from BSW UML Model
23.06.2008	1.2.2	AUTOSAR Administration	• Legal disclaimer revised
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
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

Advice for users

AUTOSAR specifications 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 specifications for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such specifications, 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 6
1.1 Goal of FlexRay transceiver driver 7
1.2 Explicitly uncovered FlexRay Transceiver Functionality 7
1.3 Active Stars 7
2 Acronyms and abbreviations 8
3 Related documentation 10
3.1 Input documents 10
3.2 Related standards and norms 10
3.3 Related specification 11
4 Constraints and assumptions 12
4.1 Limitations 12
4.2 Applicability to car domains 12
5 Dependencies to other modules 13
5.1 File structure 14
5.1.1 Naming convention for transceiver driver implementation 14
5.1.2 Code file structure 14
5.1.3 Header file structure 16
6 Requirements traceability 19
7 Functional specification 32
7.1 AUTOSAR FlexRay Transceiver Operation Mode Model 32
7.2 FlexRay transceiver hardware operation modes 34
7.2.1 Temporary “Go-To-Sleep” Mode 34
7.2.2 “Active Star” Mode 35
7.3 Error classification 35
7.4 Error detection 36
7.5 Error notification 37
7.6 Preconditions for driver initialization 38
7.7 Instance concept 38
7.8 Debug Support 39
7.9 Wake Up Support 39
7.9.1 Power-on: 39
7.9.2 Active wakeup: 39
7.9.3 Passive wakeup: 39
7.9.4 Starting FlexRay Communication without Losing Wake up Events 40
7.9.5 Stopping FlexRay Communication without Losing Wake Up Events 40
7.10 Version checking 40
8 API specification 41
8.1 Imported types 41
8.2 Type definitions 41
8.2.1 FrTrcv_TrcvModeType 42
8.2.2 FrTrcv_TrcvWUReasonType 42

8.3 Function definitions 43
8.3.1 FrTrcv_Init 44
8.3.2 FrTrcv_SetTransceiverMode 46
8.3.3 FrTrcv_GetTransceiverMode 49
8.3.4 FrTrcv_GetTransceiverWUReason 50
8.3.5 FrTrcv_GetVersionInfo 52
8.3.6 FrTrcv_ClearTransceiverWakeup 53
8.3.7 FrTrcv_CheckWakeupByTransceiver 54
8.3.8 FrTrcv_GetTransceiverError 56
8.3.9 FrTrcv_DisableTransceiverBranch 59
8.3.10 FrTrcv_EnableTransceiverBranch 60
8.4 Scheduled functions 61
8.4.1 FrTrcv_MainFunction 61
8.5 Call-back notifications 62
8.6 Expected Interfaces 62
8.7 Mandatory Interfaces 64
8.8 Optional Interfaces 64
8.9 Configurable interfaces 65
9 Sequence diagrams 67
10 Configuration specification 68
10.1 How to read this chapter 68
10.2 Containers and configuration parameters 69
10.2.1 Variants 69
10.2.2 General configuration requirements 69
10.2.3 FrTrcvGeneral 70
10.2.4 FrTrcvChannel 74
10.2.5 FrTrcvChannelDemEventParameterRefs 78
10.2.6 FrTrcvBranchIdContainer 79
10.2.7 FrTrcvAccess 80
10.2.8 FrTrcvDioAccess 80
10.2.9 FrTrcvDioChannelAccess 81
10.2.10 FrTrcvSpiSequence 82
10.3 Published Information 83
11 Not applicable requirements 84

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 a break in the cable harness, 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.

The FlexRay Transceiver Driver has the capability of wake up via bus and the usage is optional.
Some transceivers also support power supply control. Future markets will probably see a lot of different wakeup/sleep and power supply concepts.

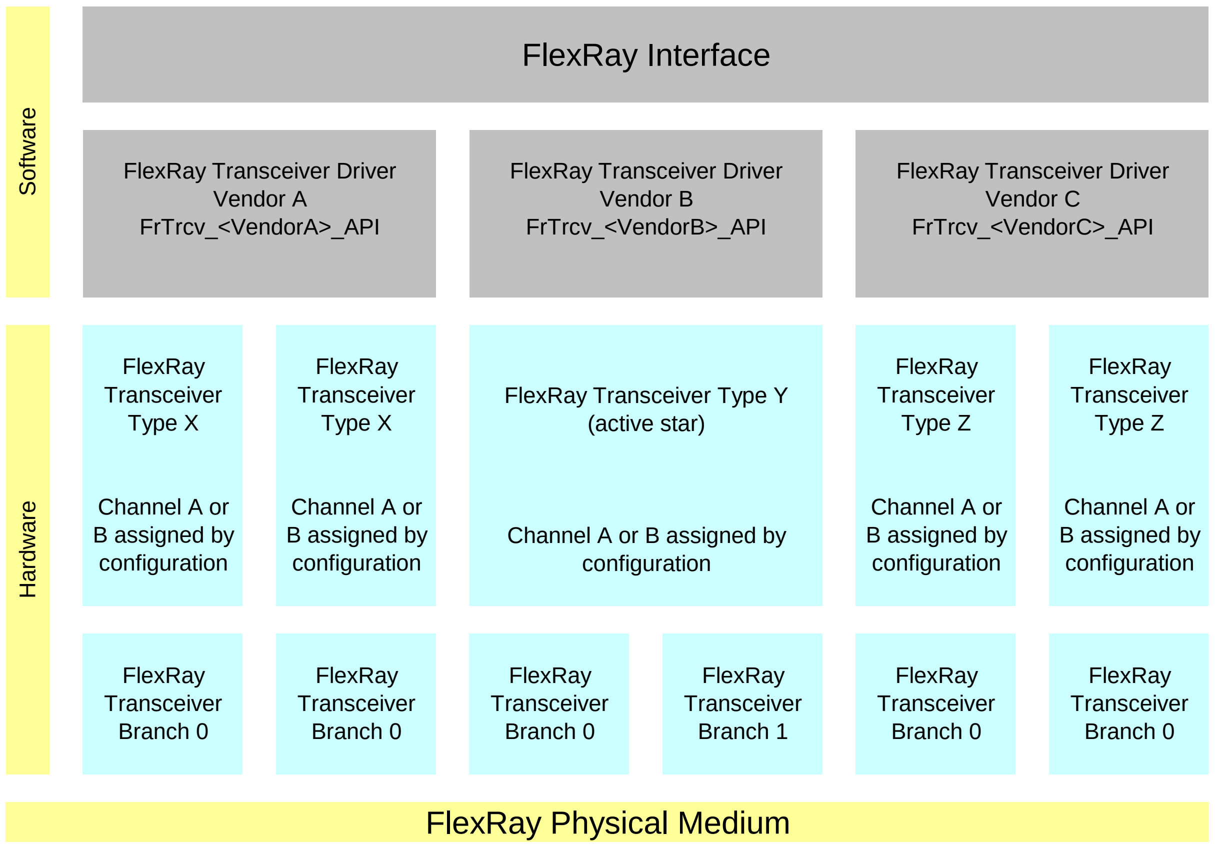

Software
Hardware
FlexRay Interface
FlexRay Transceiver Driver Vendor A FrTrcv_<VendorA>_API
FlexRay Transceiver Driver Vendor B FrTrcv_<VendorB>_API
FlexRay Transceiver Driver Vendor C FrTrcv_<VendorC>_API
FlexRay Transceiver Type X
Channel A or B assigned by configuration
FlexRay Transceiver Type X
Channel A or B assigned by configuration
FlexRay Transceiver Type Y (active star)
Channel A or B assigned by configuration
FlexRay Transceiver Type Z
Channel A or B assigned by configuration
FlexRay Transceiver Type Z
Channel A or B assigned by configuration
FlexRay Transceiver Branch 0
FlexRay Transceiver Branch 0
FlexRay Transceiver Branch 0
FlexRay Transceiver Branch 1
FlexRay Transceiver Branch 0
FlexRay Transceiver Branch 0
FlexRay Physical Medium

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

One FlexRay Interface accesses several FlexRay Transceivers (FlexRay Transceiver 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 A of FlexRay transceiver type Z is addressed by the index 0, FlexRay transceiver B by the index 1 in the example in the Figure above.
A zero based index (FrTrcv_BranchIdx) identifies the branch within the context of the transceiver. .

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 FlexRay Transceiver Driver abstracts from the ECU layout by using the APIs of the MCAL layer to access FlexRay Transceiver hardware.

1.2 Explicitly uncovered FlexRay Transceiver Functionality

The FlexRay Transceiver Driver software specification supports all transceivers conformant to [5].

1.3 Active Stars

The FlexRay Transceiver driver supports active star topologies. The host disables and enables branches of active stars.
Configuration defines the timing of active stars according to [5] and provides topology information of branches.

2 Acronyms and abbreviations

Abbreviation / Acronym:	**Description:**
µC	Microcontroller
Active Star (Network)	Star topology networks consist of one ore more active central nodes which rebroadcast(s) all transmissions received from a branch to all other branches on the network. All peripheral nodes may thus communicate with all others by transmitting to, and receiving from, the central node(s) if they are located on another branch. On detection of the failure of a branch the active star will isolate its peripheral nodes from all other branches resulting in fault confinement
API	Application Programming Interface
AUTOSAR	Automotive Open System Architecture
BD	Bus Driver
Branch	Element of an active star network topology sharing (i.e. electrically connected to) the same transmitter and receiver circuit on the physical layer. The failure of a branch will result in the isolation of its peripheral nodes by the active star from all other branches resulting in fault confinement.
BSW	Basic Software
CC	Communication Controller
ComM	Communication Manager, See [8] for details
DEM/Dem	Diagnostic Event Manager
DET/Det	Development Error Tracer
DIO/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.
ECU	Electronic Control Unit
EcuM	ECU State Manager, see [7] for details
EPL	Electrical Physical Layer
ERRN	ERRor output signal, Negated i.e. active LOW
FlexRay Node	A logical entity connected to the FlexRay Network that is capable of sending and/or receiving frames.
FrIf	FlexRay Interface
FrTrcv	FlexRay Transceiver
GPIO	General Purpose Input Output
HIS	Hersteller Initiative Software
I/O	Input/Output
IB	Internally buffered channel. Buffers containing data to transfer are inside the SPI Handler/Driver.
ID/Id	Identifier
ISR	Interrupt Service Routine
MCAL	Micro controller Abstraction Layer

MCG	Module Configuration Generator
MISRA	Motor Industry Software Reliability Association
n/a	Not applicable
OS	Operating System
Port	Port, one of the SPAL SW modules
RAM	Random Access Memory
RxD	Receive Data
RxEN	Receive Enable
SBC	System Basis Chip; A device, which integrates e.g. CAN and/or FlexRay and/or LIN transceiver, watchdog and power control.
SchM	Schedule Manager
SPAL	Standard Peripheral Abstraction Layer
SPI/Spi	Serial Peripheral Interface.
SPI/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/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/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.
SRS	Software Requirement Specification
SW	Software
SW-C	Software-Component
SWS	Software Specification
XML	eXtended Markup Language

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

[3] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[4] General Requirements on Basic Software
AUTOSAR_SRS_BSWGeneral.pdf

[5] FlexRay_ EPL-Specification_ V2.1_Rev_D2_N010
http://www.flexray.com/
FlexRay_ EPL-Specification_ V2.1_Rev_D2_N010.pdf

[6] FlexRay_ EPL-Application Notes_ V2.1_Rev_D_N009
http://www.flexray.com/
FlexRay_ EPL-Application Notes_ V2.1_Rev_D_N009.pdf

3.2 Related standards and norms

[7] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[8] Specification of Communication Manager
AUTOSAR_SWS_COMManager.pdf

[9] Specification of DIO Driver
AUTOSAR_SWS_DIODriver.pdf

[10] Specification of SPI Handler/Driver
AUTOSAR_SWS_SPIHandlerDriver.pdf

[11] Requirements on FlexRay
AUTOSAR_SRS_FlexRay.pdf

[12] Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

[13] Specification of Basic Software Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

[14] Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

[15] Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[16] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules (SWS BSW General), which is also valid for FlexRay Transceiver Driver.

Thus, the specification SWS BSW General shall be considered as additional and required specification for FlexRay Transceiver Driver.

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.

[SWS_FrTrcv_00231] ⌈The FlexRay Transceiver Driver shall use the APIs of underlying DIO drivers synchronously. ⌋ (SRS_Fr_05138)

[SWS_FrTrcv_00433] ⌈The FlexRay Transceiver Driver should use the APIs of underlying SPI drivers synchronously if possible and asynchronously where required. ⌋ ( )

[SWS_FrTrcv_00441] ⌈The FlexRay transceiver requires a LEVEL 2, Enhanced (Synchronous/Asynchronous) SPI Handler/Driver⌋ ( )

[SWS_FrTrcv_00238] ⌈The FlexRay Transceiver Driver shall handle the transceiver-specific timing requirements internally. ⌋ (SRS_Fr_05152)

The communication between the µC and the transceiver is performed via ports or SPI or both. If ports are used, applying values in a predefined sequence and with a given timing to the ports are used to communicate and change the hardware operation modes. These sequences and timings must be handled within the 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***
FrIf	The FlexRay Interface controls the state of the FlexRay transceivers via the FlexRay Transceiver Driver
Det	The FlexRay Transceiver Driver informs the Development Error Tracer on development errors
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 if supported by hardware.
RTE	The FlexRay Transceiver Driver main function may be scheduled by the by the RTE.
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 specific FlexRay API, knowledge which FlexRay transceiver driver it has to call for a specific communication FlexRay transceiver **is not required**. Only a mapping (=knowledge) generated by configuration is required!

Here is an example:

Upper layer:

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

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

FrTrcv (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 FrTrcv D1 (has Transceiver 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

[SWS_FrTrcv_00059]⌈A FlexRay Transceiver Driver implementation may support different FlexRay Transceiver hardware. ⌋ (SRS_BSW_00347)

[SWS_FrTrcv_00021] ⌈The SRS_BSW_00347 is applied for the naming in a way that no FlexRay transceiver hardware specific naming extensions are used.
The following naming convention shall be used as mentioned in SRS_BSW_00347:
Driver modules shall be named according to the following rules (only for implementation, not for the software specification):
First the module name has to be listed: <Module Abbreviation>
After that the vendor Id defined in the AUTOSAR vendor list has to be given <Vendor Id>
At last a vendor specific name follows <Vendor specific name>
All parts shall be separated by underscores "_"
This naming extension applies to the following externally visible elements of the module:
File names
API names
Published parameters⌋ (SRS_BSW_00300)

5.1.2 Code file structure

The FrTrcv module consists of the following code files:

[SWS_FrTrcv_00033] ⌈FrTrcv.c is the implementation general C file. It does not contain interrupt routines. ⌋ (SRS_BSW_00314)

[SWS_FrTrcv_00057] ⌈Pre-compile-time configuration
All modules of the AUTOSAR Basic Software, operating on Pre--compile--time configuration data (not to be modified after compile time), shall group and export the configuration data to configuration files.
Module specific configuration header file naming convention:

- <Module name>_Cfg.h and possibly
- <Module name>_Cfg.c

Static configuration is decoupled from implementation. Separation of configuration dependent data at compile time furthermore enhances flexibility, readability and reduces version management as no source code is affected. ⌋ (SRS_BSW_00345)

[SWS_FrTrcv_00079] ⌈Separate C-Files for configuration parameters
Configuration parameters being stored in memory shall be placed into separate c--files (effected parameters are those from link--time configuration as well as those from post--build time configuration).

Rationale: Enable the use of different object files. ⌋ (SRS_BSW_00380)

[SWS_FrTrcv_00117] ⌈Separate C-Files for pre-compile time configuration parameters
Configuration parameters being stored in memory shall be placed into separate c--files (effected parameters are those from link--time configuration as well as those from post--build time configuration).

Enable the use of different object files. ⌋ (SRS_BSW_00419)

5.1.3 Header file structure

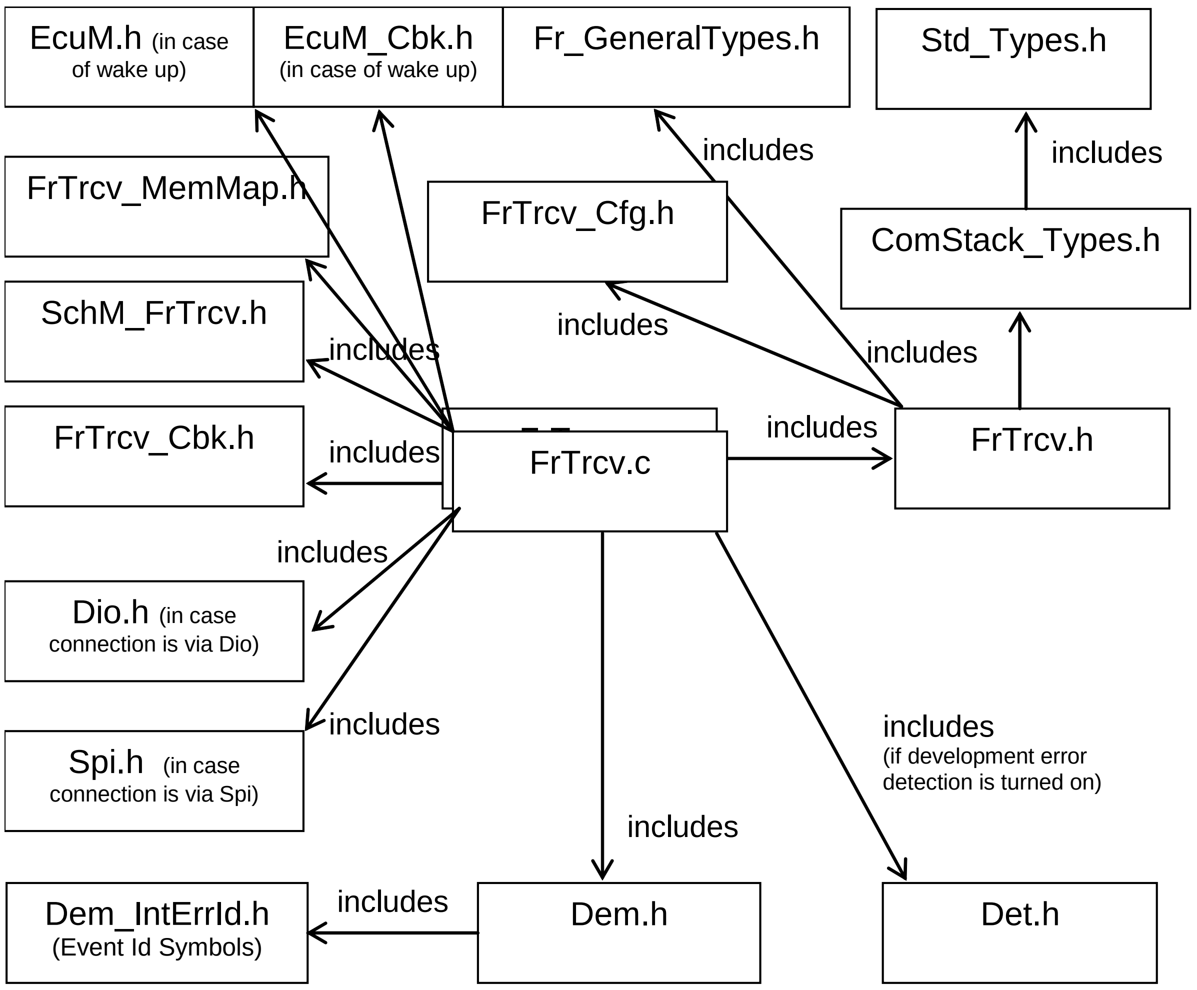

EcuM.h (in case of wake up)
EcuM_Cbk.h (in case of wake up)
Fr_GeneralTypes.h
Std_Types.h
FrTrcv_MemMap.h
FrTrcv_Cfg.h
ComStack_Types.h
SchM_FrTrcv.h
FrTrcv_Cbk.h
FrTrcv.c
FrTrcv.h
Dio.h (in case connection is via Dio)
Spi.h (in case connection is via Spi)
Dem_IntErrId.h (Event Id Symbols)
Dem.h
Det.h
includes
includes (if development error detection is turned on)

Figure 2 FlexRay Transceiver Driver Header File Structure

[SWS_FrTrcv_00022] ⌈All AUTOSAR Basic Software Modules shall only import the necessary information (i.e. header files) that is required to fulfill the modules functional requirements. ⌋ (SRS_BSW_00301)
The header file structure shall include the following FlexRay-specific header files:

[SWS_FrTrcv_00113] ⌈FrTrcv.h -- General header file of the FlexRay Transceiver Driver. It contains only information relevant for other BSW modules (API). Differences in API depending on the configuration are encapsulated. ⌋ (SRS_BSW_00415)

[SWS_FrTrcv_00023] ⌈Limit exported information: All AUTOSAR Basic Software Modules shall export only that kind of information in their correspondent header--files explicitly needed by other modules. ⌋ (SRS_BSW_00302)

[SWS_FrTrcv_00429] ⌈FrTrcv.h shall include FrTrcv_Cfg.h⌋ ( )

[SWS_FrTrcv_00430] ⌈FrTrcv.h shall include ComStack_Types.h⌋ ( )

[SWS_FrTrcv_00431] ⌈FrTrcv.h shall include Fr_GeneralTypes.h
Note: Fr_GeneralTypes.h -- Contains definitions and declarations shared by all AUTOSAR FlexRay BSW modules. ⌋ ( )

[SWS_FrTrcv_00479] ⌈FrTrcv.h shall include Fr_GeneralTypes.h for the include of general FlexRay type declarations.⌋()

[SWS_FrTrcv_00480] ⌈The types specified in SWS_FrTrcv_00434, SWS_FrTrcv_00435 shall be declared in Fr_GeneralTypes.h.⌋()

[SWS_FrTrcv_00266] ⌈SchM_FrTrcv.h -- contains Basic Software Scheduler declarations used by the FlexRay Transceiver Driver is included as specified by [21]
Hint: The Basic Software Scheduler offers concepts and services to integrate Basic Software Modules Hence, the Basic Software Scheduler
embed Basic Software Module implementations into the AUTOSAR OS context
trigger main processing functions of the Basic Software Modules
apply data consistency mechanisms for the Basic Software Modules to communicate modes between Basic Software Modules⌋ (SRS_BSW_00435)

[SWS_FrTrcv_00060] ⌈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. ⌋ (SRS_BSW_00348)

[SWS_FrTrcv_00068] ⌈Compiler.h – the 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. ⌋ (SRS_BSW_00361)

[SWS_FrTrcv_00062] ⌈Platform_Types.h – the platform specific header file. 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. ⌋ (SRS_BSW_00353)

[SWS_FrTrcv_00424] ⌈Det.h -- FrTrcv.c shall include Det.h only if development error detection is turned on. ⌋ ( )

[SWS_FrTrcv_00408] ⌈EcuM_WakeupSourceType shall be imported via EcuM.h in case wake up is configured and supported by hardware. ⌋ ( )

[SWS_FrTrcv_00409] ⌈EcuM_Cbk.h – the transceiver driver indicates the wake up source and mode to the ECU State Manager if supported by hardware⌋ ( )

[SWS_FrTrcv_00476] ⌈FrTrcv_Cbk.h - the transceiver driver encapsulates declarations of the callouts/callbacks configured in ECUC_FrTrcv_00456 in this header file. ⌋ ( )

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	ECUC_FrTrcv_00341
-	-	SWS_FrTrcv_00085
-	-	SWS_FrTrcv_00236
-	-	SWS_FrTrcv_00262
-	-	SWS_FrTrcv_00270
-	-	SWS_FrTrcv_00272
-	-	SWS_FrTrcv_00273
-	-	SWS_FrTrcv_00274
-	-	SWS_FrTrcv_00275
-	-	SWS_FrTrcv_00276
-	-	SWS_FrTrcv_00277
-	-	SWS_FrTrcv_00278
-	-	SWS_FrTrcv_00279
-	-	SWS_FrTrcv_00280
-	-	SWS_FrTrcv_00282
-	-	SWS_FrTrcv_00283
-	-	SWS_FrTrcv_00284
-	-	SWS_FrTrcv_00285
-	-	SWS_FrTrcv_00291
-	-	SWS_FrTrcv_00295
-	-	SWS_FrTrcv_00296
-	-	SWS_FrTrcv_00304
-	-	SWS_FrTrcv_00305
-	-	SWS_FrTrcv_00308
-	-	SWS_FrTrcv_00311
-	-	SWS_FrTrcv_00312
-	-	SWS_FrTrcv_00313
-	-	SWS_FrTrcv_00314
-	-	SWS_FrTrcv_00315
-	-	SWS_FrTrcv_00316
-	-	SWS_FrTrcv_00318
-	-	SWS_FrTrcv_00319
-	-	SWS_FrTrcv_00321
-	-	SWS_FrTrcv_00322
-	-	SWS_FrTrcv_00323
-	-	SWS_FrTrcv_00324

-	-	SWS_FrTrcv_00325
-	-	SWS_FrTrcv_00326
-	-	SWS_FrTrcv_00329
-	-	SWS_FrTrcv_00330
-	-	SWS_FrTrcv_00331
-	-	SWS_FrTrcv_00332
-	-	SWS_FrTrcv_00334
-	-	SWS_FrTrcv_00340
-	-	SWS_FrTrcv_00352
-	-	SWS_FrTrcv_00354
-	-	SWS_FrTrcv_00358
-	-	SWS_FrTrcv_00359
-	-	SWS_FrTrcv_00360
-	-	SWS_FrTrcv_00361
-	-	SWS_FrTrcv_00362
-	-	SWS_FrTrcv_00363
-	-	SWS_FrTrcv_00364
-	-	SWS_FrTrcv_00366
-	-	SWS_FrTrcv_00367
-	-	SWS_FrTrcv_00368
-	-	SWS_FrTrcv_00371
-	-	SWS_FrTrcv_00372
-	-	SWS_FrTrcv_00373
-	-	SWS_FrTrcv_00374
-	-	SWS_FrTrcv_00375
-	-	SWS_FrTrcv_00378
-	-	SWS_FrTrcv_00379
-	-	SWS_FrTrcv_00380
-	-	SWS_FrTrcv_00384
-	-	SWS_FrTrcv_00390
-	-	SWS_FrTrcv_00392
-	-	SWS_FrTrcv_00393
-	-	SWS_FrTrcv_00395
-	-	SWS_FrTrcv_00396
-	-	SWS_FrTrcv_00397
-	-	SWS_FrTrcv_00398
-	-	SWS_FrTrcv_00405
-	-	SWS_FrTrcv_00406
-	-	SWS_FrTrcv_00407

-	-	SWS_FrTrcv_00408
-	-	SWS_FrTrcv_00409
-	-	SWS_FrTrcv_00419
-	-	SWS_FrTrcv_00420
-	-	SWS_FrTrcv_00421
-	-	SWS_FrTrcv_00424
-	-	SWS_FrTrcv_00429
-	-	SWS_FrTrcv_00430
-	-	SWS_FrTrcv_00431
-	-	SWS_FrTrcv_00433
-	-	SWS_FrTrcv_00434
-	-	SWS_FrTrcv_00435
-	-	SWS_FrTrcv_00437
-	-	SWS_FrTrcv_00438
-	-	SWS_FrTrcv_00439
-	-	SWS_FrTrcv_00440
-	-	SWS_FrTrcv_00441
-	-	SWS_FrTrcv_00442
-	-	SWS_FrTrcv_00443
-	-	SWS_FrTrcv_00444
-	-	SWS_FrTrcv_00449
-	-	SWS_FrTrcv_00450
-	-	SWS_FrTrcv_00451
-	-	SWS_FrTrcv_00452
-	-	SWS_FrTrcv_00453
-	-	SWS_FrTrcv_00454
-	-	SWS_FrTrcv_00455
-	-	SWS_FrTrcv_00457
-	-	SWS_FrTrcv_00458
-	-	SWS_FrTrcv_00459
-	-	SWS_FrTrcv_00460
-	-	SWS_FrTrcv_00461
-	-	SWS_FrTrcv_00462
-	-	SWS_FrTrcv_00463
-	-	SWS_FrTrcv_00464
-	-	SWS_FrTrcv_00465
-	-	SWS_FrTrcv_00466
-	-	SWS_FrTrcv_00467
-	-	SWS_FrTrcv_00472

-	-	SWS_FrTrcv_00474
-	-	SWS_FrTrcv_00475
-	-	SWS_FrTrcv_00476
-	-	SWS_FrTrcv_00479
-	-	SWS_FrTrcv_00480
-	-	SWS_FrTrcv_00481
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_FrTrcv_00001
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_FrTrcv_00478
SRS_BSW_00006	The source code of software modules above the æC Abstraction Layer (MCAL) shall not be processor and compiler dependent.	SWS_FrTrcv_00478
SRS_BSW_00007	All Basic SW Modules written in C language shall conform to the MISRA C 2004 Standard.	SWS_FrTrcv_00478
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_FrTrcv_00478
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_FrTrcv_00478
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_FrTrcv_00008
SRS_BSW_00159	All modules of the AUTOSAR Basic Software shall support a tool based configuration	SWS_FrTrcv_00010
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_FrTrcv_00478
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_FrTrcv_00478
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_FrTrcv_00016
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_FrTrcv_00478
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_FrTrcv_00018
SRS_BSW_00171	Optional functionality of a Basic-SW component that is not required in the ECU shall be configurable at pre-compile-time	SWS_FrTrcv_00019
SRS_BSW_00172	The scheduling strategy that is built inside the Basic Software Modules shall be compatible with the strategy used in the system	SWS_FrTrcv_00020
SRS_BSW_00300	All AUTOSAR Basic Software Modules shall be identified by an unambiguous name	SWS_FrTrcv_00021
SRS_BSW_00301	All AUTOSAR Basic Software Modules shall only	SWS_FrTrcv_00022

	import the necessary information	
SRS_BSW_00302	All AUTOSAR Basic Software Modules shall only export information needed by other modules	SWS_FrTrcv_00023
SRS_BSW_00304	-	SWS_FrTrcv_00478
SRS_BSW_00306	AUTOSAR Basic Software Modules shall be compiler and platform independent	SWS_FrTrcv_00478
SRS_BSW_00307	Global variables naming convention	SWS_FrTrcv_00478
SRS_BSW_00308	AUTOSAR Basic Software Modules shall not define global data in their header files, but in the C file	SWS_FrTrcv_00478
SRS_BSW_00309	All AUTOSAR Basic Software Modules shall indicate all global data with read-only purposes by explicitly assigning the const keyword	SWS_FrTrcv_00478
SRS_BSW_00312	Shared code shall be reentrant	SWS_FrTrcv_00478
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_FrTrcv_00033
SRS_BSW_00321	The version numbers of AUTOSAR Basic Software Modules shall be enumerated according specific rules	SWS_FrTrcv_00478
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_FrTrcv_00478
SRS_BSW_00326	-	SWS_FrTrcv_00478
SRS_BSW_00327	Error values naming convention	SWS_FrTrcv_00041
SRS_BSW_00328	All AUTOSAR Basic Software Modules shall avoid the duplication of code	SWS_FrTrcv_00478
SRS_BSW_00329	-	SWS_FrTrcv_00043
SRS_BSW_00330	It shall be allowed to use macros instead of functions where source code is used and runtime is critical	SWS_FrTrcv_00478
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_FrTrcv_00045
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_FrTrcv_00478
SRS_BSW_00334	All Basic Software Modules shall provide an XML file that contains the meta data	SWS_FrTrcv_00047
SRS_BSW_00335	Status values naming convention	SWS_FrTrcv_00048
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_FrTrcv_00478
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_FrTrcv_00478
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_FrTrcv_00478
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_FrTrcv_00478
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_FrTrcv_00057
SRS_BSW_00347	A Naming seperation of different instances of BSW	SWS_FrTrcv_00059

	drivers shall be in place	
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_FrTrcv_00060
SRS_BSW_00350	All AUTOSAR Basic Software Modules shall apply a specific naming rule for enabling/disabling the detection and reporting of development errors	SWS_FrTrcv_00061
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_FrTrcv_00062
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_FrTrcv_00064
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_FrTrcv_00065
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_FrTrcv_00478
SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_FrTrcv_00478
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_FrTrcv_00068
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_FrTrcv_00069
SRS_BSW_00370	-	SWS_FrTrcv_00478
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_FrTrcv_00072
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_FrTrcv_00074
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_FrTrcv_00076
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_FrTrcv_00478
SRS_BSW_00379	All software modules shall provide a module identifier in the header file and in the module XML description file.	SWS_FrTrcv_00078
SRS_BSW_00380	Configuration parameters being stored in memory shall be placed into separate c-files	SWS_FrTrcv_00079
SRS_BSW_00382	-	SWS_FrTrcv_00478
SRS_BSW_00383	The Basic Software Module specifications shall specify which other configuration files from other modules they use at least in the description	SWS_FrTrcv_00478
SRS_BSW_00384	The Basic Software Module specifications shall specify at least in the description which other modules they require	SWS_FrTrcv_00478
SRS_BSW_00385	List possible error notifications	SWS_FrTrcv_00084
SRS_BSW_00386	The BSW shall specify the configuration for	SWS_FrTrcv_00478

	detecting an error	
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_FrTrcv_00086
SRS_BSW_00389	Containers shall have names	SWS_FrTrcv_00088
SRS_BSW_00390	Parameter content shall be unique within the module	SWS_FrTrcv_00089
SRS_BSW_00392	Parameters shall have a type	SWS_FrTrcv_00478
SRS_BSW_00393	Parameters shall have a range	SWS_FrTrcv_00092
SRS_BSW_00394	The Basic Software Module specifications shall specify the scope of the configuration parameters	SWS_FrTrcv_00093
SRS_BSW_00395	The Basic Software Module specifications shall list all configuration parameter dependencies	SWS_FrTrcv_00094
SRS_BSW_00397	The configuration parameters in pre-compile time are fixed before compilation starts	SWS_FrTrcv_00317
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_FrTrcv_00478
SRS_BSW_00399	Parameter-sets shall be located in a separate segment and shall be loaded after the code	SWS_FrTrcv_00478
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_FrTrcv_00478
SRS_BSW_00401	Documentation of multiple instances of configuration parameters shall be available	SWS_FrTrcv_00478
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_FrTrcv_00478
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_FrTrcv_00478
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_FrTrcv_00104
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_FrTrcv_00105
SRS_BSW_00410	Compiler switches shall have defined values	SWS_FrTrcv_00478
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_FrTrcv_00478
SRS_BSW_00414	The init function may have parameters	SWS_FrTrcv_00112
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_FrTrcv_00113
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_FrTrcv_00478
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_FrTrcv_00478
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a	SWS_FrTrcv_00117

	separate c-file	
SRS_BSW_00420	-	SWS_FrTrcv_00478
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_FrTrcv_00478
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_FrTrcv_00478
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_FrTrcv_00122
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_FrTrcv_00123
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_FrTrcv_00478
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_FrTrcv_00478
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_FrTrcv_00126
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_FrTrcv_00478
SRS_BSW_00431	-	SWS_FrTrcv_00478
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_FrTrcv_00478
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_FrTrcv_00478
SRS_BSW_00434	-	SWS_FrTrcv_00478
SRS_BSW_00435	-	SWS_FrTrcv_00266
SRS_BSW_05023	-	SWS_FrTrcv_00478
SRS_BSW_05025	-	SWS_FrTrcv_00478
SRS_BSW_05035	-	SWS_FrTrcv_00478
SRS_BSW_05038	-	SWS_FrTrcv_00478
SRS_BSW_05040	-	SWS_FrTrcv_00478
SRS_BSW_05041	-	SWS_FrTrcv_00478
SRS_BSW_05045	-	SWS_FrTrcv_00478
SRS_BSW_05067	-	SWS_FrTrcv_00478
SRS_BSW_05068	-	SWS_FrTrcv_00478
SRS_BSW_05069	-	SWS_FrTrcv_00478
SRS_BSW_05078	-	SWS_FrTrcv_00478
SRS_BSW_05082	-	SWS_FrTrcv_00478
SRS_BSW_05083	-	SWS_FrTrcv_00478
SRS_BSW_05084	-	SWS_FrTrcv_00478
SRS_BSW_05085	-	SWS_FrTrcv_00478

SRS_BSW_05101	-	SWS_FrTrcv_00478
SRS_BSW_05102	-	SWS_FrTrcv_00478
SRS_BSW_05104	-	SWS_FrTrcv_00478
SRS_BSW_05107	-	SWS_FrTrcv_00478
SRS_BSW_05111	-	SWS_FrTrcv_00478
SRS_BSW_05113	-	SWS_FrTrcv_00478
SRS_BSW_05124	-	SWS_FrTrcv_00478
SRS_BSW_05153	-	SWS_FrTrcv_00478
SRS_BSW_05154	-	SWS_FrTrcv_00478
SRS_BSW_05155	-	SWS_FrTrcv_00478
SRS_BSW_05162	-	SWS_FrTrcv_00478
SRS_BSW_05163	-	SWS_FrTrcv_00478
SRS_BSW_05164	-	SWS_FrTrcv_00478
SRS_BSW_05165	-	SWS_FrTrcv_00478
SRS_BSW_05172	-	SWS_FrTrcv_00478
SRS_BSW_05173	-	SWS_FrTrcv_00478
SRS_BSW_05200	-	SWS_FrTrcv_00478
SRS_BSW_05201	-	SWS_FrTrcv_00478
SRS_BSW_05202	-	SWS_FrTrcv_00478
SRS_BSW_05203	-	SWS_FrTrcv_00436
SRS_BSW_05204	-	SWS_FrTrcv_00478
SRS_BSW_05205	-	SWS_FrTrcv_00478
SRS_BSW_05206	-	SWS_FrTrcv_00478
SRS_BSW_05207	-	SWS_FrTrcv_00478
SRS_BSW_05208	-	SWS_FrTrcv_00478
SRS_BSW_05209	-	SWS_FrTrcv_00478
SRS_BSW_05210	-	SWS_FrTrcv_00478
SRS_BSW_05211	-	SWS_FrTrcv_00478
SRS_BSW_05212	-	SWS_FrTrcv_00412
SRS_BSW_05213	-	SWS_FrTrcv_00415
SRS_BSW_05214	-	SWS_FrTrcv_00414
SRS_BSW_05215	-	SWS_FrTrcv_00478
SRS_Fr_05000	Synchronous SW Modules shall be supported	SWS_FrTrcv_00478
SRS_Fr_05001	Asynchronous SW Modules shall be supported	SWS_FrTrcv_00478
SRS_Fr_05002	FlexRay Interface and FlexRay Driver shall operated synchronized to the global time	SWS_FrTrcv_00478
SRS_Fr_05003	Slot/Cycle Multiplexing shall be supported	SWS_FrTrcv_00478
SRS_Fr_05004	The FlexRay Interface shall provide a PDU-based data API to all upper layers.	SWS_FrTrcv_00478
SRS_Fr_05005	The CC Hardware FIFO Mechanism shall be	SWS_FrTrcv_00478

	supported	
SRS_Fr_05006	Abstraction of FlexRay-Specific Features shall be provided	SWS_FrTrcv_00478
SRS_Fr_05007	The FlexRay Interface shall be able to communicate with at least four FlexRay CCs via the appropriate FlexRay Driver(s)	SWS_FrTrcv_00478
SRS_Fr_05009	The FlexRay Interface shall allocate the needed memory space only once for a PDU sent multiple times in the FlexRay matrix	SWS_FrTrcv_00478
SRS_Fr_05010	Each PDU shall have one PDU-ID	SWS_FrTrcv_00478
SRS_Fr_05011	Initialization of the Low-Level Parameters shall be available	SWS_FrTrcv_00478
SRS_Fr_05012	Initialization of the FlexRay CC Transmit/Receive Buffers shall be available	SWS_FrTrcv_00478
SRS_Fr_05013	The local Memory Space shall be initialized	SWS_FrTrcv_00478
SRS_Fr_05015	The FlexRay Interface shall provide a software interface to start-up a specific FlexRay CC	SWS_FrTrcv_00478
SRS_Fr_05016	A FlexRay CC Communication shall be aborted when wanted	SWS_FrTrcv_00478
SRS_Fr_05018	The FlexRay Interface shall provide a software interface to send a wake-up pattern on a channel or CC	SWS_FrTrcv_00478
SRS_Fr_05019	FlexRay Global Time shall be provided	SWS_FrTrcv_00478
SRS_Fr_05022	FlexRay CC POC Status shall be available	SWS_FrTrcv_00478
SRS_Fr_05024	The software interface of the Driver shall be independent of the CC buffers' configuration	SWS_FrTrcv_00478
SRS_Fr_05027	A PDU shall be transmitted via the FlexRay communication system	SWS_FrTrcv_00478
SRS_Fr_05031	A FlexRay CC shall be initialized and configured	SWS_FrTrcv_00478
SRS_Fr_05033	Tick Conversion shall be provided	SWS_FrTrcv_00478
SRS_Fr_05034	The configuration data shall be modifiable by a Flashing Process	SWS_FrTrcv_00478
SRS_Fr_05039	The Operation Mode of a FlexRay Transceiver shall be set	SWS_FrTrcv_00478
SRS_Fr_05042	The FlexRay Interface shall allow switching from one configuration to another one in Normal Active Mode	SWS_FrTrcv_00478
SRS_Fr_05044	CC's Absolute Timer shall be provided	SWS_FrTrcv_00478
SRS_Fr_05046	Absolute Alarms of a CC shall be enabled	SWS_FrTrcv_00478
SRS_Fr_05047	Absolute Alarms of a CC shall be disabled	SWS_FrTrcv_00478
SRS_Fr_05048	Absolute Alarms of a CC shall be acknowledged	SWS_FrTrcv_00478
SRS_Fr_05049	Relative Alarms of a CC shall be enabled	SWS_FrTrcv_00478
SRS_Fr_05050	Relative Alarms of a CC shall be disabled	SWS_FrTrcv_00478
SRS_Fr_05051	Relative Alarms of a CC shall be acknowledged	SWS_FrTrcv_00478
SRS_Fr_05052	Cycle Length in Macroticks shall be provided	SWS_FrTrcv_00478

SRS_Fr_05053	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific Cluster	SWS_FrTrcv_00478
SRS_Fr_05055	Timer Interrupts during Shutdown shall be avoided	SWS_FrTrcv_00478
SRS_Fr_05056	Configuration of the FlexRay Interface shall be done at System Configuration Time	SWS_FrTrcv_00478
SRS_Fr_05058	The configuration of the FlexRay Driver shall be defined at system configuration time.	SWS_FrTrcv_00478
SRS_Fr_05059	The Driver shall be configure the CC's transmit/receive buffers	SWS_FrTrcv_00478
SRS_Fr_05060	Scheduling of Copy Operation into/from FlexRay CC shall be possible	SWS_FrTrcv_00478
SRS_Fr_05063	A FlexRay CC Communication shall be halted when wanted	SWS_FrTrcv_00478
SRS_Fr_05064	Abstraction of FlexRay CC-specific Implementation shall be provided	SWS_FrTrcv_00478
SRS_Fr_05065	The FlexRay Driver shall be able to communicate with at least four FlexRay CCs of the same type	SWS_FrTrcv_00478
SRS_Fr_05066	L-SDU-Based API shall be available	SWS_FrTrcv_00478
SRS_Fr_05072	The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is Out of Sync	SWS_FrTrcv_00478
SRS_Fr_05073	The FlexRay Transport Layer shall be configured to be compliant with the ISO 10681-2 specification	SWS_FrTrcv_00478
SRS_Fr_05074	The FlexRay Transport Layer software module shall be located between the PDU Router and the FlexRay Interface	SWS_FrTrcv_00478
SRS_Fr_05076	The FlexRay Transport Layer shall support at least 32 logical FlexRay Transport Layer active connections being used concurrently	SWS_FrTrcv_00478
SRS_Fr_05077	Each N-SDU shall have a unique identifier	SWS_FrTrcv_00478
SRS_Fr_05079	Transport Connection Properties "ISO 15765-2"	SWS_FrTrcv_00478
SRS_Fr_05088	FlexRay Transport Layer's variables shall be initialized	SWS_FrTrcv_00478
SRS_Fr_05089	The FlexRay Transport Layer services shall not be operational before initializing the module.	SWS_FrTrcv_00478
SRS_Fr_05090	The FlexRay Transport Layer shall support per connection the ISO 10681-2 / ISO 15765-2 service N_ChangeParameter	SWS_FrTrcv_00478
SRS_Fr_05093	A cancellation service of transmission shal be provided at any time	SWS_FrTrcv_00478
SRS_Fr_05095	The FlexRay Transport Layer shall support the dynamic bandwidth control mechanism	SWS_FrTrcv_00478
SRS_Fr_05096	Communication controllers shall be assigned to FlexRay Driver.	SWS_FrTrcv_00478
SRS_Fr_05097	The FlexRay Interface shall be able to communicate with at least four FlexRay Drivers	SWS_FrTrcv_00478
SRS_Fr_05106	The Buffer of a specific CC in Normal Active Mode	SWS_FrTrcv_00478

	shall be reconfigurable	
SRS_Fr_05109	The FlexRay Driver shall provide a software interface to start-up a specific FlexRay CC	SWS_FrTrcv_00478
SRS_Fr_05114	A FlexRay CC Communication shall be aborted when wanted	SWS_FrTrcv_00478
SRS_Fr_05115	The FlexRay CC Communication shall be halted when wanted	SWS_FrTrcv_00478
SRS_Fr_05116	Initialization of FlexRay CC shall be available	SWS_FrTrcv_00478
SRS_Fr_05117	A Wake-Up Pattern shall be sent on a specific channel of a CC	SWS_FrTrcv_00478
SRS_Fr_05120	FlexRay CC POC Status shall be provided	SWS_FrTrcv_00478
SRS_Fr_05121	FlexRay CC Sync State shall be provided	SWS_FrTrcv_00478
SRS_Fr_05123	The Configuration shall be modifiable by a Flashing Process	SWS_FrTrcv_00478
SRS_Fr_05125	The FlexRay Driver shall provide services to handle interrupts of a FlexRay Communication Controller.	SWS_FrTrcv_00478
SRS_Fr_05126	PDU Update/Valid Information shall be handled	SWS_FrTrcv_00478
SRS_Fr_05130	The FlexRay Interface shall support PDU transmission buffer queues	SWS_FrTrcv_00478
SRS_Fr_05131	The transceiver driver package shall include a description file with the basic information needed to configure the driver for a given bus and the supported notifications.	SWS_FrTrcv_00225
SRS_Fr_05132	The FlexRay Transceiver Driver shall support the configuration for more than one transceiver type as well as for more than one Cluster	SWS_FrTrcv_00226
SRS_Fr_05133	The FlexRay Transceiver Driver shall support the independent configuration of the bus operation mode for each supported Cluster.	SWS_FrTrcv_00227
SRS_Fr_05134	The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack.	SWS_FrTrcv_00228
SRS_Fr_05136	The FlexRay Transceiver Driver shall support the compile time configuration of one notification to a higher layer for change notification for "wake-up by bus" events.	SWS_FrTrcv_00229
SRS_Fr_05137	The FlexRay Transceiver Driver shall provide an API to initialize the driver internally and set then all attached FlexRay Transceivers in their pre-selected operation modes.	SWS_FrTrcv_00230
SRS_Fr_05138	The FlexRay Transceiver Driver API shall be synchronous.	SWS_FrTrcv_00231
SRS_Fr_05144	The FlexRay Transceiver Wake-up Reason shall be provided	SWS_FrTrcv_00232
SRS_Fr_05147	The FlexRay Transceiver Driver shall support a notification to inform higher layers about the wake-up by bus.	SWS_FrTrcv_00233
SRS_Fr_05148	The FlexRay Transceiver Driver shall support situations where a wake-up by bus occurs at the	SWS_FrTrcv_00234

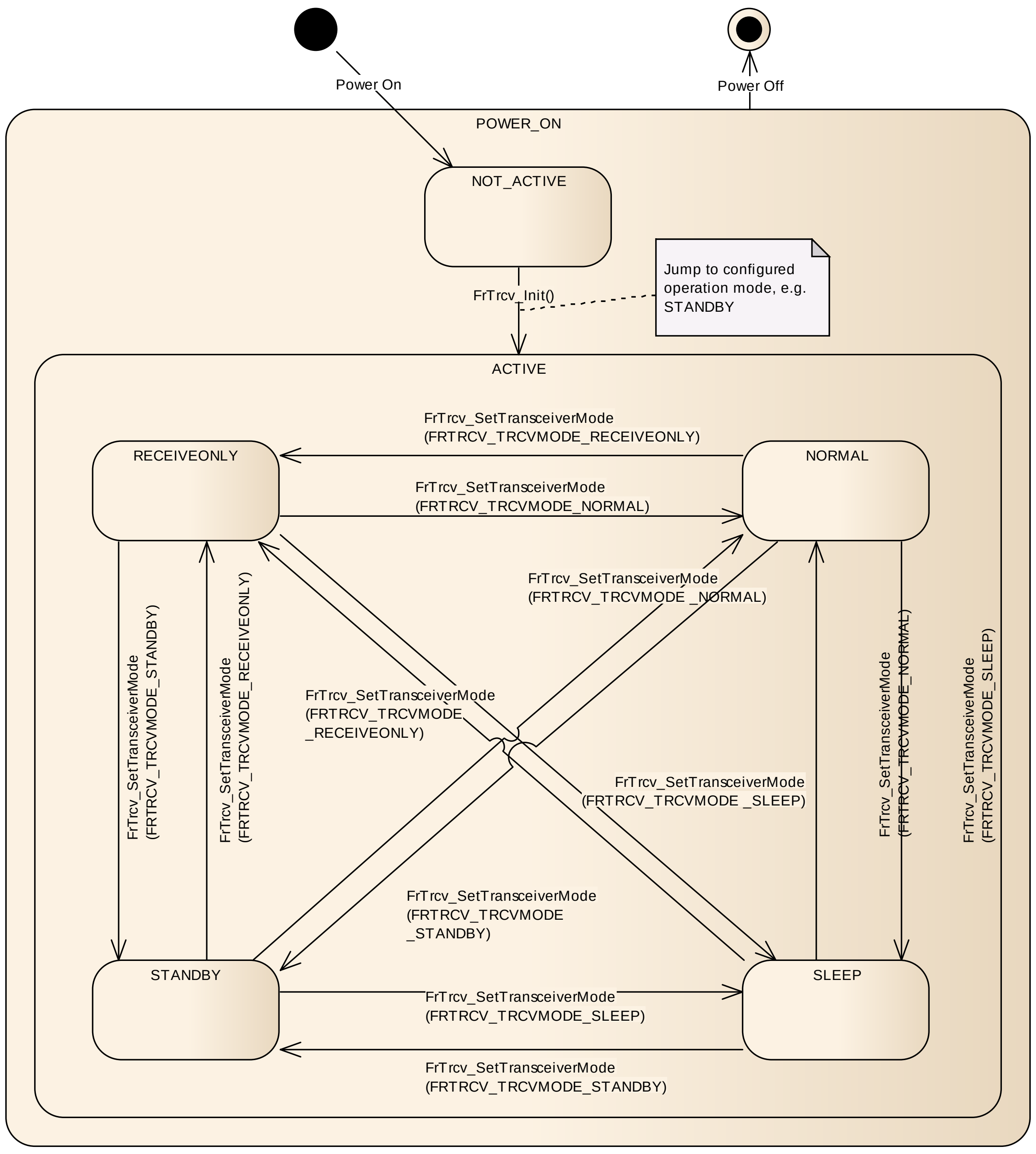
	same moment the transition to standby/sleep is executed by the driver.	
SRS_Fr_05151	The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness.	SWS_FrTrcv_00237, SWS_FrTrcv_00281, SWS_FrTrcv_00306
SRS_Fr_05152	The FlexRay Transceiver Driver shall handle the transceiver-specific timing requirements internally.	SWS_FrTrcv_00238
SRS_Fr_05156	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific CC	SWS_FrTrcv_00478
SRS_Fr_05157	The Operation Mode of a FlexRay Transceiver shall be available	SWS_FrTrcv_00478
SRS_Fr_05158	The wake-up reason of a specific FlexRay Transceiver device shall be available	SWS_FrTrcv_00478
SRS_Fr_05161	Pending Wake-up Events of a Transceiver shall be cleared if necessary	SWS_FrTrcv_00247
SRS_Fr_05166	It shall be possible to set the FlexRay Transceiver Operation Mode	SWS_FrTrcv_00252
SRS_Fr_05167	The FlexRay Transceiver Operation Mode shall be provided	SWS_FrTrcv_00253
SRS_Fr_05168	FlexRay Transceiver Error State shall be indicated (modify according to Monitoring Concept and Concept Reliability)	SWS_FrTrcv_00391
SRS_Fr_05169	Timer Interrupts during Start-up shall be avoided	SWS_FrTrcv_00478
SRS_Fr_05170	PDUs received via the FlexRay communication system shall be retrieved	SWS_FrTrcv_00478
SRS_Fr_05171	A PDU Transmit Confirmation shall be provided	SWS_FrTrcv_00478
SRS_Fr_05174	The FlexRay Interface shall provide services to handle interrupts of a FlexRay Communication Controller.	SWS_FrTrcv_00478
SRS_Fr_05175	The Error Informations shall be provided	SWS_FrTrcv_00478

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 many currently 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.

[SWS_FrTrcv_00227] ⌈The FlexRay Transceiver Driver shall support the independent configuration of the bus operation mode for each supported Cluster. Due to the different startup requirements on a multiple-FlexRay-Cluster-ECU, the FlexRay Transceiver Driver support the independent pre-selection of the bus operation mode to which each FlexRay Transceiver device is set during the driver initialization. ⌋ (SRS_Fr_05133)


Power On
Power Off
POWER_ON
NOT_ACTIVE
FrTrcv_Init()
Jump to configured operation mode, e.g. STANDBY
ACTIVE
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_RECEIVEONLY)
RECEIVEONLY
NORMAL
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_NORMAL)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE _NORMAL)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_STANDBY)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_RECEIVEONLY)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE _RECEIVEONLY)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE _SLEEP)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_NORMAL)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_SLEEP)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE _STANDBY)
STANDBY
SLEEP
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_SLEEP)
FrTrcv_SetTransceiverMode (FRTRCV_TRCVMODE_STANDBY)

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_Init() 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 if supported by hardware.
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 if supported by hardware.
RECEIVEONLY	Similar to NORMAL, but only reception is possible.

[SWS_FrTrcv_00291] ⌈On initialization, the FrTrcv module shall switch all covered FlexRay transceivers into the state ACTIVE. This is observable, see SWS_FrTrcv_00277
In state ACTIVE each FlexRay transceiver may be in a different sub state.
Only the states NORMAL and STANDBY are mandatory for FlexRay transceivers; all other states are optional.
If a state is optional according to [5] 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 state actually supported by the transceiver hardware 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.

[SWS_FrTrcv_00352] ⌈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

[SWS_FrTrcv_00451] ⌈If a transceiver supports active star mode, do NOT assume it is in node mode. ⌋ ( )

7.3 Error classification

[SWS_FrTrcv_00084] ⌈Production code errors and development errors of FlexRay Transceiver Driver are provided in the table below. This list must be mapped into the code (i.e. the respective function calls to the error notifications must be in the code). ⌋ (SRS_BSW_00385)

[SWS_FrTrcv_00085] ⌈Development and Production Errors used by the FlexRay Transceiver Driver:

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 called with wrong parameter	Development	FRTRCV_E_FR_INVALID_BRANCHIDX FlexRay transceiver branch index out of range	0x02
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
API service called passing a NULL pointer as a parameter	Development	FRTRCV_E_FR_TRCV_NULL_PTR	0x15
Error Status of Class A (GPIO) transceiver	Production	FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>	* (Assigned by DEM)
Error Status of Class B (SPI) transceiver bus errors where TrcvIdx is the transceiver index	Production	FRTRCV_E_FR_BUSERROR_TRCV<TrcvIdx>	* (Assigned by DEM)

No/incorrect communication to transceiver	Development	FRTRCV_E_FR_NO_CONTROL_TRCV	0x16

⌋ ( )

Note: The DEM module is configured to include these symbols, and the MCG of the DEM provides an header file with the symbols, which are then available to the FrTrcv module by inclusion of Dem.h.

[SWS_FrTrcv_00041] ⌈Error values naming convention
All AUTOSAR Basic Software Modules shall apply the following naming rules for all error values:
Error values shall have only CAPITAL LETTERS
Naming convention: FRTRCV_E_<ERRORNAME>
If <ERRORNAME> consists of several words, they shall be separated by underscores
The error shows to which module it belongs. ⌋ (SRS_BSW_00327)

7.4 Error detection

[SWS_FrTrcv_00237] ⌈The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness if supported by hardware. ⌋ (SRS_Fr_05151)

[SWS_FrTrcv_00354] ⌈In case of faults of the transceiver hardware, the FlexRay Transceiver Driver shall raise a development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled. ⌋ ( )

Example: 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.

[SWS_FrTrcv_00295] ⌈The FrTrcv module shall check for bus errors and report them to DEM executing Dem_ReportErrorStatus(FRTRCV_E_FR_BUSERROR_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREFAILED). ⌋ ( )

[SWS_FrTrcv_00472] ⌈If a no bus error is detected, the module shall execute Dem_ReportErrorStatus(FRTRCV_E_FR_BUSERROR_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREPASSED). ⌋ ( )

In the above descriptions, <TrcvIdx> represents the transceiver index.

Note on Host Software / ECU control (derived from [6])
The application controller (host) has to ensure that the BD enters NORMAL (or RECEIVEONLY) mode, before the CC enters one of its states where the CC starts to listen to the channel (e.g. POC:startup, POC:normal active, POC:normal passive).
In case the BD cannot enter NORMAL (or RECEIVEONLY) due to low voltage conditions, this low voltage will be signaled as error to the host. In this case the host shall force the CC to step back to a non listening state (e.g. POC:default config, POC:config, POC:ready, POC:halt).
The reason for this is that as long as the BD is not in NORMAL (or RECEIVEONLY) mode, no information about the status of the channel is available via the signals RxD and RxEN

When shutting down the ECU, the host shall command the BD into a low power mode before commanding the CC into a state, where the CC does not evaluate the RxD signal. This is to ensure that the CC does not miss any communication element on the channel. Mind that the BD does not necessarily react on traffic when in a low power mode. For more information see [PS08], especially those sections that deal with wakeup and startup.

7.5 Error notification

[SWS_FrTrcv_00391] ⌈If the configuration parameter FrTrcvErrorCheckDuringCommunication is set to true, the function FrTrcv_MainFunction shall report periodically the error state of the FlexRay transceiver to the Diagnostic Event Manager. ⌋ (SRS_Fr_05168)

[SWS_FrTrcv_00384] ⌈If an error (e.g. the state of the ERRN pin is active low) is detected the module shall execute Dem_ReportErrorStatus(FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREFAILED).
In the above description, <TrcvIdx> represents the transceiver index. ⌋ ( )

[SWS_FrTrcv_00395] ⌈If an error is not detected (e.g. the state of the ERRN pin is passive high) the module shall execute Dem_ReportErrorStatus(FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREPASSED).
In the above descriptions, <TrcvIdx> represents the transceiver index. ⌋ ( )

Note: It is possible that ERRN status is active only for a short time. There is a possibility that ERRN status has already vanished when the MainFunction is executed. In this case, ERRN could be connected to an interrupt pin in the actual hardware. This way the transceiver driver would detect any active transitions of the ERRN status.

7.6 Preconditions for driver initialization

[SWS_FrTrcv_00296] ⌈The FrTrcv module shall use drivers for SPI and Dio to control the FlexRay bus transceiver hardware. ⌋ ( )

Note: The environment of the FrTrcv module ensures that all necessary BSW drivers (used by the FrTrcv module) have been initialized and are usable before FrTrcv_Init is called.
Thus, these drivers are assumed available and ready to operate before the FlexRay bus transceiver driver is initialized.

[SWS_FrTrcv_00358] ⌈The FlexRay bus transceiver driver shall fulfill the FlexRay Transceiver hardware timing requirements also on initialization. ⌋ ( )

[SWS_FrTrcv_00359] ⌈The FlexRay transceiver driver initialization shall schedule before other BSW modules (e.g. the FlexRay State manager) access its software services. ⌋ ( )

[SWS_FrTrcv_00360] ⌈The runtime of the underlying services used shall be short enough and synchronous in order to fulfill the requirements defined by the FlexRay EPL [5] and the timing requirements of the hardware device used. (SWS_FrTrcv_00231). ⌋ ( )

[SWS_FrTrcv_00361] ⌈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. ⌋ ( )

7.7 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.

[SWS_FrTrcv_00226] ⌈Multiple FlexRay transceivers of the same type are handled by a single FlexRay transceiver driver. ⌋ (SRS_Fr_05132)

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)!

7.8 Debug Support

[SWS_FrTrcv_00405] ⌈The mode (FrTrcv_TrcvModeType) of the FlexRay Transceiver shall be available for debugging. ⌋ ( )

[SWS_FrTrcv_00406] ⌈The wake up reason (FrTrcv_TrcvWUReasonType) of the FlexRay Transceiver shall be available for debugging if supported by hardware. ⌋ ( )

[SWS_FrTrcv_00407] ⌈The wake up state of the FlexRay Transceiver shall be available for debugging if supported by hardware. ⌋ ( )

[SWS_FrTrcv_00450] ⌈The branch state of the FlexRay Transceiver active stars shall be available for debugging if supported by hardware. ⌋ ( )

7.9 Wake Up Support

From the EcuM point of view, the FrTrcv only needs to detect and report passive wakeups if supported by hardware. An active wakeup or power-on is handled by the EcuM/ComM anyway and there is no need to ask FrTrcv.

7.9.1 Power-on:

EcuM is started and no wakeup source reports a passive wakeup. So EcuM does a full startup. Applications are started and if they request communication an active wakeup of the corresponding busses is performed by ComM.

7.9.2 Active wakeup:

EcuM wakes up and checks the wakeup sources. If it was a wakeup, the wakeup source reports the wakeup event. Since the wakeup source (here a port pin or similar) is _not_ a communication network, EcuM will not inform ComM. Instead, applications are started and if they request communication a startup of the corresponding networks is performed by ComM.

7.9.3 Passive wakeup:

EcuM wakes up and checks the wakeup sources. If it was a wakeup, the wakeup source reports the wakeup event. Since the wakeup source (this time bus transceivers and/or controllers) is a communication network, EcuM will inform ComM. ComM will perform a startup of this network.

So, EcuM only needs a wakeup event from FrTrcv in case of a passive wakeup. Allother cases shall not be reported to EcuM.

7.9.4 Starting FlexRay Communication without Losing Wake up Events

While the CC is in state HALT until it is in state READY, the FlexRay transceiver shall remain in STANDBY in order to detect wake up events if supported by hardware.
Caveat: Wake up events may get lost during power supply failure (this is detected and reported to DEM).
The FlexRay transceiver driver will store the wake up information (including a time stamp) internally if supported by hardware.
Wake up events will be cleared with the API FrTrcv_ClearTransceiverWakeup (intended use: internal only)
As soon as the CC is in state READY, NORMAL is requested. Until the requested state has been achieved (polling SPI or DIO), the CC shall not leave READY.

7.9.5 Stopping FlexRay Communication without Losing Wake Up Events

Deferred READY is requested from the CC.
As soon as READY is reached, STANDBY is requested. Until the requested state has been achieved (polling SPI or DIO), the CC shall not leave READY.

7.10 Version checking

For details refer to the chapter 5.1.8 “Version Check” in *SWS_BSWGeneral.*

8 API specification

8.1 Imported types

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

[SWS_FrTrcv_00321] ⌈

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

⌋ ( )

8.2 Type definitions

[SWS_FrTrcv_00045] ⌈Separation of error and status values
All Basic Software Modules shall strictly separate error and status information. This requirement applies to return values and also to internal variables. ⌋
(SRS_BSW_00331)

[SWS_FrTrcv_00069] ⌈Do not return development error codes via API
All AUTOSAR Basic Software Modules shall not return specific development error codes via the API. In case of a detected development error the error shall only be reported to the DET. If the API-- function which detected the error has a return type it shall return E_NOT_OK. ⌋ (SRS_BSW_00369)

[SWS_FrTrcv_00076] ⌈Module specific API return types
If a Basic Software Module needs module specific return types, it shall use one of the following possibilities:

1. Use uint8 as return value, take the standard E_OK value from Std_Types.h and define additional return values using #define.

2. Define a module specific return value with typedef enum.
   Hint: Within this enum, E_OK cannot be used (because E_OK is already #defined in Std_Types.h and OSEK OS)

⌋ (SRS_BSW_00377)

8.2.1 FrTrcv_TrcvModeType

[SWS_FrTrcv_00481]⌈

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.	

⌋()

[SWS_FrTrcv_00048] ⌈Status values naming convention:
The following naming rules apply for status values that are visible outside of the module: - Status values shall have only CAPITAL LETTERS - Naming convention: FRTRCV_<STATUSNAME>
If <STATUSNAME> consists of several words, they shall be separated by underscores. ⌋ (SRS_BSW_00335)

[SWS_FrTrcv_00434] ⌈The type definition FrTrcv_TrcvModeType 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⌋ ( )

According to [5], at least these operation modes are defined:
• NORMAL
• STANDBY
• RECEIVEONLY (if supported by hardware)
• SLEEP (if supported by hardware)
Note: According to [5] every FlexRay Transceiver has to support two mandatory states: FrTRCV_TRCVMODE_STANDBY and FrTRCV_TRCVMODE_NORMAL; all other states are optional.

8.2.2 FrTrcv_TrcvWUReasonType

[SWS_FrTrcv_00435] ⌈The type definition 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. ⌋ ( )

[SWS_FrTrcv_00074] ⌈

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_BY_PIN`	The transceiver has detected a wake-up event at one of the transceiver's pins (not at the FlexRay bus).
	`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.	

⌋ (SRS_BSW_00375)

8.3 Function definitions

[SWS_FrTrcv_00043] ⌈Avoidance of generic interfaces
All Basic Software Modules shall not use generic interfaces. A ‘generic interface’ is an interface without a defined scope and content. ⌋ (SRS_BSW_00329)

[SWS_FrTrcv_00089] ⌈Parameter content shall be unique within the module
The same intention, logical contents or semantic shall be placed in one parameter only (There must not be several parameters with the same intention, logical contents or semantic). ⌋ (SRS_BSW_00390)

[SWS_FrTrcv_00092] ⌈Parameters shall have a range
Each parameter shall have a list of valid values or the minimum as well as maximum values shall be specified. ⌋ (SRS_BSW_00393)

[SWS_FrTrcv_00093] ⌈Specify the scope of the parameters
A parameter may only be applicable for the module it is defined in. In this case, the parameter is marked as “local”. Alternatively, the parameter may be shared with other modules (i.e. exported). In that case, the scope shall list the names of the other modules sharing this parameter. Each parameter shall only be defined once in one module. All other modules sharing the parameter must not define the parameter again. Instead, the parameter is to be imported. This is applicable for c--code as well as for .XML configuration. ⌋ (SRS_BSW_00394)

[SWS_FrTrcv_00094] ⌈List the required parameters (per parameter)
The Basic Software Module specifications must list configuration parameters of this or other modules this parameter relies on. A dependency is for example: the value of another parameter influences or invalidates the setting of this parameter. ⌋ (SRS_BSW_00395)

[SWS_FrTrcv_00104] ⌈Check module initialization
A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called. The initialization function of the BSW modules shall set the static status variable to a value not equal to 0. Exception shall be the "<Module name>_GetVersionInfo" function. It shall be possible to call this function at any time. ⌋ (SRS_BSW_00406)

8.3.1 FrTrcv_Init

[SWS_FrTrcv_00322] ⌈

Service name:	FrTrcv_Init

Syntax:	`void FrTrcv_Init(`
	`    void`
	`)`

Service ID[hex]:	0x00
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This service initializes the FrTrcv.

⌋ ( )

[SWS_FrTrcv_00008] ⌈Initialization interface
The FlexRay transceiver driver initializes variables and hardware resources in a separate initialization function. This function shall be named FrTrcv_Init().
Note: According to SWS_EcuM_02562: 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). ⌋ (SRS_BSW_00101)

[SWS_FrTrcv_00228] ⌈Initialization Sequence for FlexRay Transceiver Driver
The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack.
To start the ECU from power-up or reset, a fixed sequence of driver and manager initialization is necessary to reach the required startup times and to set the FlexRay stack into working state. The sequence itself depends on many requirements, partly dependent on the FlexRay controller and the power supply concept⌋ (SRS_Fr_05134)

[SWS_FrTrcv_00230] ⌈Initialize the FlexRay Transceiver Driver
The FlexRay Transceiver Driver shall provide an API to initialize the driver internally and set then all attached FlexRay Transceivers in their preselected operation modes.

- The FlexRay Transceiver Driver must be initialized during the power-up/reset sequence of the ECU.
- Depending on the used drivers to control the transceivers (e.g. DIO, SPI), they must be already available and working when the FlexRay Transceiver Driver is initialized.
- The wake-up reason has to be detected and stored during the execution of the driver initialization, too. ⌋ (SRS_Fr_05137)

[SWS_FrTrcv_00270] ⌈The function FrTrcv_Init shall set all transceivers in the state defined by the configuration parameter FRTRCV_INIT_STATE, i.e. in any state defined by SWS_FrTrcv_00434. ⌋ ( )

Note that in the time span between power up and the call FrTrcv_Init 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.

Note: Before calling FrTrcv_Init the environment insures that all SPAL drivers used by the FrTrcv module to access the transceiver hardware are initialized and usable.

[SWS_FrTrcv_00437] ⌈In case of a fault during transceiver access, the initialization process shall be restarted from the beginning. It shall be retried until the retry counter exceeds the number specified by FrTrcvRetryCountInInit. If the process doesn’t succeed, the function FrTrcv_Init shall raise a development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also SWS_FrTrcv_00237). ⌋ ( )

[SWS_FrTrcv_00390] ⌈If the configuration parameter FrTrcvErrorCheckInInit is set true, the function FrTrcv_Init shall check state of ERRN to detect hardware failure. If an error is detected, FrTrcv_Init shall raise a production error FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>.⌋ ( )

[SWS_FrTrcv_00438] ⌈The function FrTrcv_Init shall check whether there has been a wake up due to transceiver activity if supported by hardware and report this to the EcuM via EcuM_SetWakeupEvent(event). ⌋ ( )

[SWS_FrTrcv_00362] ⌈The driver has to notify ECU State Manager by invoking the EcuM_SetWakeupEvent service once only if a wakeup event is detected. ⌋ ( )

[SWS_FrTrcv_00363] ⌈The driver has to detect a pending wakeup event during the initialization. ⌋ ( )

[SWS_FrTrcv_00112] ⌈The init function in general shall have no parameter. ⌋ (SRS_BSW_00414)

[SWS_FrTrcv_00065] ⌈The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void. ⌋ (SRS_BSW_00358)

[SWS_FrTrcv_00366] ⌈The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back i. e. wake up events are enabled at driver initialization if configured accordingly and supported by hardware. ⌋ ( )

[SWS_FrTrcv_00367] ⌈The FlexRay Transceiver Driver driver shall support a wakeup ISR if supported by hardware. ⌋ ( )

[SWS_FrTrcv_00414] ⌈Hardware Resetting Function on Bus Driver
The FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality
Hint: When trouble occurs in the hardware level, it is likely to fix the cause by resetting the hardware. This function shall be executed when a configurable amount of errors are detected in by the FlexRay modules and have been reported to DEM. ⌋ (SRS_BSW_05214)

[SWS_FrTrcv_00455] ⌈If a transceiver is in active star mode and one or more branches have been disabled, the FlexRay Transceiver Driver shall re-enable all branches on initialization. ⌋ ( )

8.3.2 FrTrcv_SetTransceiverMode

[SWS_FrTrcv_00323] ⌈

Service name:	FrTrcv_SetTransceiverMode	

Syntax:	`Std_ReturnType 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:	Std_ReturnType	E_OK: will be returned if the transceiver state has been changed to the requested mode. E_NOT_OK: will be returned if the transceiver state change has failed. The previous state has not been changed.
Description:	This service sets the transceiver mode.	

⌋ ( )

[SWS_FrTrcv_00252] ⌈Set FlexRay Transceiver Operation Mode
The FlexRay Transceiver Driver shall provide a software interface to set the operation mode of a specific FlexRay Transceiver device. ⌋ (SRS_Fr_05166)

[SWS_FrTrcv_00392] ⌈Whenever FrTrcv_SetTransceiverMode changes the state to STANDBY, it shall clear error history in transceiver as long as the hardware supports such a function. This modification has the same effect as introducing a new API FrTrcv_ClearErrorHistory() and adding a call of the function in FrSm's sequence. ⌋ ( )

[SWS_FrTrcv_00393] ⌈After setting FlexRay Trcv to STANDBY the FrTrcv shall call a configurable callout function (see ECUC_FrTrcv_00456), in which the integrator can enable the interrupt pin. If optional states are used, this requirement applies, whenever a transition from a state not capable to a state capable of detecting and/or latching wake up occurs (e.g.: a transition from RECEIVEONLY to SLEEP) and if wake up is supported by hardware. ⌋ ( )

[SWS_FrTrcv_00086] ⌈The callout function of SWS_FrTrcv_00393 shall be pre compile configurable, its name is provided in ECUC_FrTrcv_00456. ⌋ (SRS_BSW_00387)

[SWS_FrTrcv_00474] ⌈A mode request of the current mode is allowed and shall not lead to an error even if DET is enabled. ⌋ ( )

[SWS_FrTrcv_00064] ⌈Standard API return type:
The Std_ReturnType shall normally be used with value E_OK or E_NOT_OK. If those return values are not sufficient user specific values can be defined by using the 6 least specific bits. ⌋ (SRS_BSW_00357)

[SWS_FrTrcv_00272] ⌈The function FrTrcv_SetTransceiverMode shall switch the internal state of the transceiver identified by FrTrcv_TrcvIdx to the state indicated by FrTrcv_TrcvMode. ⌋ ( )

[SWS_FrTrcv_00273] ⌈The function FrTrcv_SetTransceiverMode shall return E_NOT_OK and doesn't change the current state if a transition not defined in FrTrcv_TrcvModeType is requested. ⌋ ( )

[SWS_FrTrcv_00274] ⌈If an optional state is NOT supported by the transceiver and ECU hardware, the function FrTrcv_SetTransceiverMode shall switch to an equivalent state. ⌋ ( )

[SWS_FrTrcv_00440] ⌈If the FlexRay transceiver and ECU hardware does not support a receive only state, FrTRCV_TRCVMODE_NORMAL shall be used. ⌋ ( )

[SWS_FrTrcv_00236] ⌈If the FlexRay transceiver and ECU hardware does not support a sleep state, FrTRCV_TRCVMODE_STANDBY shall be used.
(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.) ⌋ ( )

[SWS_FrTrcv_00278] ⌈In case of a fault during transceiver access, the function FrTrcv_SetTransceiverMode shall raise development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also SWS_FrTrcv_00237) and return E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00368] ⌈The API function calls to the FlexRay Transceiver Driver shall be synchronuous. ⌋ ( )

[SWS_FrTrcv_00275] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00276] ⌈If development error detection for the module FrTrcv is enabled: If the mode transition fails (SWS_FrTrcv_00452), the function FrTrcv_SetTransceiverMode shall raise the following development error and return E_NOT_OK:
- 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⌋ ( )

[SWS_FrTrcv_00452] ⌈A mode transition fails, if the mode returned by the API service FrTrcv_GetTransceiverMode() would mismatch the mode requested by the API service FrTrcv_SetTransceiverMode().⌋ ( )

[SWS_FrTrcv_00277] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00415] ⌈Hardware Checking Function on Bus Driver
The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly.
This functionality ensures that the hardware is working as expected.
Improvement of hardware reliability. ⌋ (SRS_BSW_05213)

[SWS_FrTrcv_00454] ⌈If a transceiver is in active star mode and one or more branches are disabled, the FlexRay Transceiver Driver shall avoid side effects of the API service FrTrcv_SetTransceiverMode() which re-enable any branches. ⌋ ( )

8.3.3 FrTrcv_GetTransceiverMode

[SWS_FrTrcv_00324] ⌈

Service name:	FrTrcv_GetTransceiverMode	

Syntax:	`Std_ReturnType 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:	Std_ReturnType	E_OK: will be returned if the transceiver state has been provided E_NOT_OK: will be returned if the parameter is out of range. Output parameters remain unchanged.
Description:	This function returns the actual state of the transceiver.	

⌋ ( )

[SWS_FrTrcv_00253] ⌈The function FrTrcv_GetTransceiverMode shall return the state of the transceiver identified by FrTrcv_TrcvIdx. ⌋ (SRS_Fr_05167)

[SWS_FrTrcv_00281] ⌈In case of a fault during transceiver access, the function FrTrcv_GetTransceiverMode shall raise the development error

FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also SWS_FrTrcv_00237) and return E_NOT_OK. ⌋ (SRS_Fr_05151)

See FrTrcv_Init (SWS_FrTrcv_00270) for the provided state after the FlexRay transceiver driver initialization until the first operation mode change request.

The number of supported FlexRay transceivers and their type is statically set in the configuration phase.

[SWS_FrTrcv_00279] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00280] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00397] ⌈When the caller provides a NULL pointer as a parameter value to the API
FrTrcv_GetTransceiverMode, the return value shall be E_NOT_OK and the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DET if development error detection is enabled. ⌋ ( )

8.3.4 FrTrcv_GetTransceiverWUReason

[SWS_FrTrcv_00325] ⌈

Service name:	FrTrcv_GetTransceiverWUReason	

Syntax:	`Std_ReturnType 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 wakeup reason information.
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver wake up source has been provided E_NOT_OK: will be returned if the transceiver wakeup

		reason is not defined in FrTrcv_TrcvWUReasonType. Output parameters remain unchanged.
Description:	This function returns the wakeup reason.	

⌋ ( )

[SWS_FrTrcv_00232] ⌈The function FrTrcv_GetTransceiverWUReason shall return the reason for the wake up that the FlexRay transceiver identified by FrTrcv_TrcvIdx has detected if supported by hardware.
The ability to detect and differentiate the possible wake up reasons depends strongly on the FlexRay transceiver hardware. ⌋ (SRS_Fr_05144)

[SWS_FrTrcv_00284] ⌈In case of a fault during transceiver access, the function FrTrcv_GetTransceiverWUReason shall raise development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also SWS_FrTrcv_00237) and return E_NOT_OK. ⌋ ( )

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.

[SWS_FrTrcv_00453] ⌈The FlexRay Transceiver Driver shall report the wake up reason in the order defined by SWS_FrTrcv_00074.Thus, FRTRCV_WU_BY_BUS is reported first in case of multiple concurrent events. ⌋ ( )

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.

[SWS_FrTrcv_00282] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00283] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00398] ⌈When the caller provides a NULL pointer as a parameter value to the API FrTrcv_GetTransceiverWUReason, the development error

FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DET if development error detection is enabled. ⌋ ( )

8.3.5 FrTrcv_GetVersionInfo

[SWS_FrTrcv_00326] ⌈

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.	

⌋ ( )

[SWS_FrTrcv_00001] ⌈Version identification: The FlexRay transceiver driver shall support a version information API. ⌋ (SRS_BSW_00003)

[SWS_FrTrcv_00105] ⌈Function to read out published parameters: The function FrTrcv_GetVersionInfo shall return the version information of the FrTrcv module.

The version information consists of three parts:
• Two bytes for the vendor ID
• One byte for the module ID
• Three bytes version number.
The numbering shall be vendor specific; it consists of:
The major, the minor and the patch version number of the module.
The AUTOSAR specification version number shall not be included.
It shall be possible to call this function at any time
(e.g. before the init function is called). ⌋ (SRS_BSW_00407)

[SWS_FrTrcv_00078] ⌈Module identification
All software modules shall provide a module identifier in the header file and in the module XML description file.
The value shall be taken from the Basic Software Module List. Naming convention:
FRTRCV_MODULE_ID
The module ID shall be represented in uint8 (8 bit). ⌋ (SRS_BSW_00379)

[SWS_FrTrcv_00285] ⌈The function FrTrcv_GetVersionInfo shall return the version information of the FrTrcv module, NOT the version of the FlexRay transceiver hardware. ⌋ ( )

[SWS_FrTrcv_00396] ⌈When a NULL pointer is passed as a parameter value of FrTrcv_GetVersionInfo, the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DETshall be reported to DET if development error detection is enabled. ⌋ ( )

8.3.6 FrTrcv_ClearTransceiverWakeup

[SWS_FrTrcv_00329] ⌈

Service name:	FrTrcv_ClearTransceiverWakeup	

Syntax:	`Std_ReturnType 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:	Std_ReturnType	E_OK: will be returned if the transceiver wake up source has been cleared E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx is out of range. Wake up state remains unchanged.
Description:	This function clears a pending wake up event.	

⌋ ( )

[SWS_FrTrcv_00247] ⌈The function FrTrcv_ClearTransceiverWakeup shall clear a pending wake up event on the transceiver identified by FrTrcv_TrcvIdx. ⌋ (SRS_Fr_05161)

[SWS_FrTrcv_00371] ⌈The API shall clear all pending wake up events under control of the higher layer .

It may even 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.

[SWS_FrTrcv_00306] ⌈In case of a fault during transceiver access, the function FrTrcv_ClearTransceiverWakeup shall raise the development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also SWS_FrTrcv_00237) and return E_NOT_OK. ⌋ (SRS_Fr_05151)

[SWS_FrTrcv_00304] ⌈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 E_NOT_OK. ⌋ ( )

[SWS_FrTrcv_00305] ⌈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 E_NOT_OK. ⌋ ( )

8.3.7 FrTrcv_CheckWakeupByTransceiver

[SWS_FrTrcv_00331] ⌈

Service name:	FrTrcv_CheckWakeupByTransceiver	

Syntax:	`void FrTrcv_CheckWakeupByTransceiver(`
	`    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:	--	

⌋ ( )

[SWS_FrTrcv_00364] ⌈The driver shall notify ECU State Manager of wakeup events if triggered by the API call FrTrcv_CheckWakeupByTransceiver. ⌋ ( )

[SWS_FrTrcv_00233] ⌈Notification for Wake-up by Bus
The FlexRay Transceiver Driver shall support a notification to inform higher layers about the wake-up by bus. It must be possible to support more than one bus within the ECU with this notification.
The FlexRay Transceiver Driver shall call this notification when the transceiver detects a wake-up by bus.
The FlexRay Transceiver Driver is notified by a notification from the underlying SPI or DIO driver in that case. The notification is executed in the context of the caller (may be interrupt context!). Because the delay from wake-up detection until the start of the necessary actions has a large influence on the startup time of an ECU, this event shall be processed internally and transferred immediately via this notification to the next layer.

The call context and the reaction time depend on the call context of the lower layer DIO or SPI. In case of interrupt it is very fast but data consistency issues must be covered in all layers. In case of polling data consistency issues are reduced but reaction time may be too slow.
Rationale: Support wake-up by FlexRay Transceiver devices.
Use Case: The FlexRay Transceiver detects a wake-up condition on the bus and shows this to the µC via e.g. a port pin.
Further handling depends on current ECU state. Assuming the ECU is halted, the change on the port may terminate the "HALT" statement and let the processor continue its work. The assigned port interrupt will be executed and this handler is called. Now, the FlexRay Transceiver Driver will store the wake-up reason and give the call via this notification to e.g. the NM to let the NM decide how to handle the event. ⌋ (SRS_Fr_05147)

[SWS_FrTrcv_00262] ⌈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 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. ⌋ ( )

[SWS_FrTrcv_00311] ⌈The function FrTrcv_CheckWakeupByTransceiver() shall call the API service EcuM_SetWakeupEvent with the parameter value ECUM_WKSOURCE_FRTRCV_FR of EcuM_WakeupSourceType <u>only</u> in case a valid wakeup originated from the transceiver identified by FrTrcv_TrcvIdx. Thus, shared interrupts are easily de-multiplexed: Drivers just return doing nothing if they did not trigger the interrupt. ⌋ ( )

[SWS_FrTrcv_00374] ⌈The function FrTrcv_CheckWakeupByTransceiver() 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. ⌋ ( )

[SWS_FrTrcv_00375] ⌈The FlexRay Transceiver Driver shall check for wake up events immediately after the API call FrTrcv_SetTransceiverMode if supported by hardware. ⌋ ( )

[SWS_FrTrcv_00378] ⌈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. ⌋ ( )

[SWS_FrTrcv_00379] ⌈Calling FrTrcv_CheckWakeupByTransceiver in an interrupt context shall be supported. Hint: This has to be documented according to (SRS_BSW_00333).

Hint: While the ECU is in SLEEP, the main function () is not scheduled.yet, but the wake up reason has to be identified by the FlexRay Transceiver Driver in the context of the wake up interrupt. ⌋ ( )

[SWS_FrTrcv_00380] ⌈Calling FrTrcv_CheckWakeupByTransceiver by a polling process in sleep mode shall be supported. ⌋ ( )

[SWS_FrTrcv_00312] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_CheckWakeupByTransceiver shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX. ⌋ ( )

[SWS_FrTrcv_00313] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_CheckWakeupByTransceiver shall raise development error FRTRCV_E_FR_UNINIT. ⌋ ( )

[SWS_FrTrcv_00229] ⌈Configuration "Notification for Wake-up by Bus"
The FlexRay Transceiver Driver shall support the compile time configuration of one notification to a higher layer for change notification for "wake-up by bus" events.
One wake-up by bus event notification shall be supported to one higher layer.
If a transceiver device does not support "wake-up by bus", this notification is never called for this bus.

Efficient coupling between FlexRay Transceiver Driver and higher layer. ⌋
(SRS_Fr_05136)

[SWS_FrTrcv_00234] ⌈Support for Wake-up During Sleep Transition
The FlexRay Transceiver Driver shall support situations where a wake-up by bus occurs at the same moment the transition to standby/sleep is executed by the driver.
Wake-up by bus is always asynchronous to the internal transition to sleep.
In worst case, the wake-up occurs during the transition to sleep.
This situation must be covered by the design and explicitly tested for each ECU.
The driver shall create a wake-up notification by bus immediately after the API to enter the standby/sleep mode has finished.
The calling/controlling component (NM or ECU state manager) must be capable to handle the wake-up immediately after requesting the standby/sleep.
Safe wake-up and sleep handling.

All busses with a wake-up by bus are affected. ⌋ (SRS_Fr_05148)

8.3.8 FrTrcv_GetTransceiverError

[SWS_FrTrcv_00419] ⌈

Service name:	FrTrcv_GetTransceiverError

Syntax:	`Std_ReturnType FrTrcv_GetTransceiverError(`
	`    uint8 FrTrcv_TrcvIdx,`

	`uint8 FrTrcv_BranchIdx,`
	`uint32* FrTrcv_BusErrorState`
	`)`	

Service ID[hex]:	0x08	
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_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	FrTrcv_BusErrorState	Pointer to structure of detailed transceiver error state; - Parameter is reference to variable: The transceiver driver will write the current transceiver error state information according to FrTrcv420 there, if the transceiver supports this information.
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver error state has been provided E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx or FrTrcv_BranchIdx is out of range or the transceiver error state is not available. Output parameters remain unchanged.
Description:	All mandatory errors defined by the FlexRay EPL [5] which are supported by the FlexRay transceiver hardware can be accessed via this API:In addition to errors on the physical layer and local to the ECU hardware, a global error flag is provided.	

⌋ ( )

[SWS_FrTrcv_00412] ⌈Detect Error Information in Bus Driver
The FlexRay Transceiver Driver shall provide an API that detects errors in the bus driver and notifies the application level.

The FlexRay modules should provide information that only the modules can detect. ⌋ (SRS_BSW_05212)

[SWS_FrTrcv_00420] ⌈The FlexRay Transceiver Driver shall support all mandatory errors defined by the FlexRay EPL [5] if supported by hardware:

Availability	Topology	Description	Bit
Mandatory	Global error	Any of the mandatory errors defined in this table, please see SWS_FrTrcv_00457, SWS_FrTrcv_00458	0
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between bus lines according to [5]	1
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between positive bus line and ground according to [5]	2
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between positive bus line and power supply according to [5]	3

Mandatory	on the bus (i. e. external to the ECU):	Short circuit between negative bus line and power supply according to [5]	4
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between negative bus line and ground according to [5]	5
Mandatory	on the bus (i. e. external to the ECU):	Any bus fault according to [5], which cannot be resolved according to the description of bit 1...5	6
Mandatory	Local error	Under voltage of transceiver power supply according to [5]	7
Mandatory	Local error	FlexRay transceiver is permanently enabled according to [5]	8
Mandatory	Local error	Over temperature of transceiver according to [5]	9

⌋ ( )

[SWS_FrTrcv_00421] ⌈Additional transceiver errors, which are supported by hardware shall be appended to the table in SWS_FrTrcv_00420. ⌋ ( )

[SWS_FrTrcv_00439] ⌈When the caller provides a NULL pointer as a parameter value to the API FrTrcv_GetTransceiverError the return value shall be E_NOT_OK and the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DET if development error detection is enabled. ⌋ ( )

[SWS_FrTrcv_00444] ⌈The FlexRay Transceiver Driver shall identify bus faults per branch on active star transceivers. ⌋ ( )

[SWS_FrTrcv_00449] ⌈The FlexRay Transceiver Driver shall ignore the parameter FrTrcv_BranchIdx in case the transceiver is not an active star device. ⌋ ( )

[SWS_FrTrcv_00457] ⌈The FlexRay Transceiver Driver shall set bit 0 of FrTrcv_BusErrorState if the state of ERRN is active low for transceivers according to class A of [5] ⌋ ( )

[SWS_FrTrcv_00458] ⌈The FlexRay Transceiver Driver shall set bit 0 of FrTrcv_BusErrorState if any of bit 1...9 is set for transceivers according to class B of [5] ⌋ ( )

[SWS_FrTrcv_00459] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_GetTransceiverError shall raise development error FRTRCV_E_FR_UNINIT. ⌋ ( )

[SWS_FrTrcv_00460] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function

FrTrcv_GetTransceiverError shall raise development error

FRTRCV_E_FR_INVALID_TRCVIDX. ⌋ ( )

[SWS_FrTrcv_00461] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_BranchIdx is out of range, the function FrTrcv_GetTransceiverError shall raise development error

FRTRCV_E_FR_INVALID_BRANCHIDX. ⌋ ( )

8.3.9 FrTrcv_DisableTransceiverBranch

[SWS_FrTrcv_00442] ⌈The FlexRay Transceiver Driver shall disable the faulty branches of active stars

Service name:	FrTrcv_DisableTransceiverBranch	
Syntax:	`Std_ReturnType FrTrcv_DisableTransceiverBranch( uint8 FrTrcv_TrcvIdx, uint8 FrTrcv_BranchIdx )`	
Service ID[hex]:	0x0f	
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_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver branch has been disabled E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx or FrTrcv_BranchIdx is out of range. Branch state remains unchanged.
Description:	This function disables the specified branch on the addressed (active star) transceiver.	

⌋ ( )

[SWS_FrTrcv_00462] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_DisableTransceiverBranch shall raise development error

FRTRCV_E_FR_UNINIT. ⌋ ( )

[SWS_FrTrcv_00463] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_DisableTransceiverBranch shall raise development error

FRTRCV_E_FR_INVALID_TRCVIDX. ⌋ ( )

[SWS_FrTrcv_00464] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_BranchIdx is out of range, the function

FrTrcv_DisableTransceiverBranch shall raise development error

FRTRCV_E_FR_INVALID_BRANCHIDX. ⌋ ( )

8.3.10 FrTrcv_EnableTransceiverBranch

[SWS_FrTrcv_00443] ⌈The FlexRay Transceiver Driver shall enable the branches of active stars synchronously to the FlexRay bus schedule

Service name:	FrTrcv_EnableTransceiverBranch	
Syntax:	`Std_ReturnType FrTrcv_EnableTransceiverBranch( uint8 FrTrcv_TrcvIdx, uint8 FrTrcv_BranchIdx )`	
Service ID[hex]:	0x10	
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_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver branch has been enabled E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx or FrTrcv_BranchIdx is out of range. Branch state remains unchanged.
Description:	This function enables the specified branch on the addressed (active star) transceiver.	

⌋ ( )

[SWS_FrTrcv_00465] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_EnableTransceiverBranch shall raise development error

FRTRCV_E_FR_UNINIT. ⌋ ( )

[SWS_FrTrcv_00466] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_EnableTransceiverBranch shall raise development error

FRTRCV_E_FR_INVALID_TRCVIDX. ⌋ ( )

[SWS_FrTrcv_00467] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_BranchIdx is out of range, the function FrTrcv_EnableTransceiverBranch shall raise development error

FRTRCV_E_FR_INVALID_BRANCHIDX. ⌋ ( )

8.4 Scheduled functions

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

8.4.1 FrTrcv_MainFunction

[SWS_FrTrcv_00330] ⌈

Service name:	FrTrcv_MainFunction

Syntax:	`void FrTrcv_MainFunction(`
	`    void`
	`)`

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

⌋ ( )

[SWS_FrTrcv_00020] ⌈Compatibility and documentation of scheduling strategy: The FlexRay bus transceiver driver may have cyclic jobs like polling for wake up events (if configured). The period of the main function is defined by configuration. ⌋ (SRS_BSW_00172)

[SWS_FrTrcv_00072] ⌈Main processing function naming convention: The main function of the FlexRay transceiver driver shall be named FrTrcv_MainFunction. ⌋ (SRS_BSW_00373)

[SWS_FrTrcv_00126] ⌈Execution order dependencies of main processing functions: The main processing function of the FlexRay transceiver driver shall be independent of the FlexRay bus schedule i. e. it may be scheduled either synchronous to the FlexRay bus schedule as well as asynchronous to the FlexRay bus schedule. ⌋ (SRS_BSW_00428)

[SWS_FrTrcv_00340] ⌈The function FrTrcv_MainFunction shall scan all busses in STANDBY and SLEEP for wake up events and store them internally.
Note: EcuM will invoke EcuM_CheckWakeup. This results in the invocation of FrTrcv_CheckWakeupByTransceiver. So, in case of POLLING, the API FrTrcv_CheckWakeupByTransceiver shall invoke the EcuM_SetWakeupEvent. ⌋ ( )

[SWS_FrTrcv_00122] ⌈The function FrTrcv_MainFunction shall be implemented in such a way that it can run inside a basic task (scheduled by the AUTOSAR RTE). ⌋ (SRS_BSW_00424)

[SWS_FrTrcv_00372] ⌈The Basic Software Scheduler shall execute FrTrcv_MainFunction with a period configured by the parameter FRTRCV_MAIN_FUNCTION_CYCLE_TIME. See ECUC_FrTrcv_00343 for more details. ⌋ ( )

[SWS_FrTrcv_00373] ⌈If a cycle time of 0 is configured in FrTrcvMainFunctionCycleTime this function is not executed by the Basic Software Scheduler and need not be present in compiled code. See ECUC_FrTrcv_00343 for more details. ⌋ ( )

[SWS_FrTrcv_00308] ⌈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. ⌋ ( )

[SWS_FrTrcv_00123] ⌈Trigger conditions for schedulable objects
The BSW module description template shall provide means to model the following trigger conditions of schedulable objects:

- Cyclic timings (fixed and selectable during runtime)
- Sporadic events

⌋ (SRS_BSW_00425)

[SWS_FrTrcv_00436] ⌈Error Information in Bus Driver
The FrTrcv_MainFunction shall call API of SRS_BSW_05212 (SWS_FrTrcv_00412) FrTrcv_GetTransceiverError () periodically to detect error information in BD. ⌋ (SRS_BSW_05203)

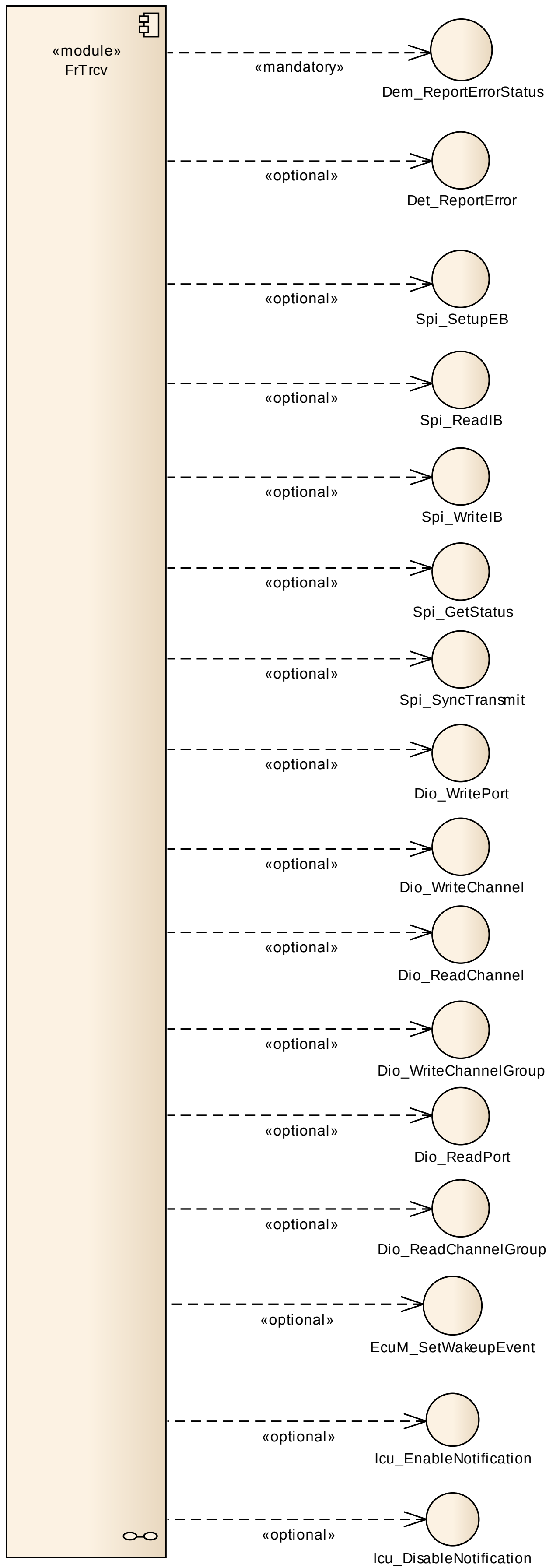
[ECUC_FrTrcv_00341] ⌈A pre-compile configuration parameter FrTrcvDevErrorDetect shall determine whether this functionality SWS_FrTrcv_00436 is activated. ⌋ ( )
Note: The FlexRay modules should provide information that only the modules can detect.
Applications could take actions to recover the failure cause like resetting the modules when they receive this error information.

8.5 Call-back notifications

This is a list of functions provided for lower layer modules.
E.g. the SPI driver might provide a call back whenever an transfer is finished.
(There are none).
Please see SWS_FrTrcv_00393 and ECUC_FrTrcv_00456.

8.6 Expected Interfaces

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


«module»
FrTrcv
«mandatory»
Dem_ReportErrorStatus
«optional»
Det_ReportError
«optional»
Spi_SetupEB
«optional»
Spi_ReadIB
«optional»
Spi_WriteIB
«optional»
Spi_GetStatus
«optional»
Spi_SyncTransmit
«optional»
Dio_WritePort
«optional»
Dio_WriteChannel
«optional»
Dio_ReadChannel
«optional»
Dio_WriteChannelGroup
«optional»
Dio_ReadPort
«optional»
Dio_ReadChannelGroup
«optional»
EcuM_SetWakeupEvent
«optional»
Icu_EnableNotification
«optional»
Icu_DisableNotification

8.7 Mandatory Interfaces

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

[SWS_FrTrcv_00332] ⌈

API function	***Description***
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.

⌋ ( )

8.8 Optional Interfaces

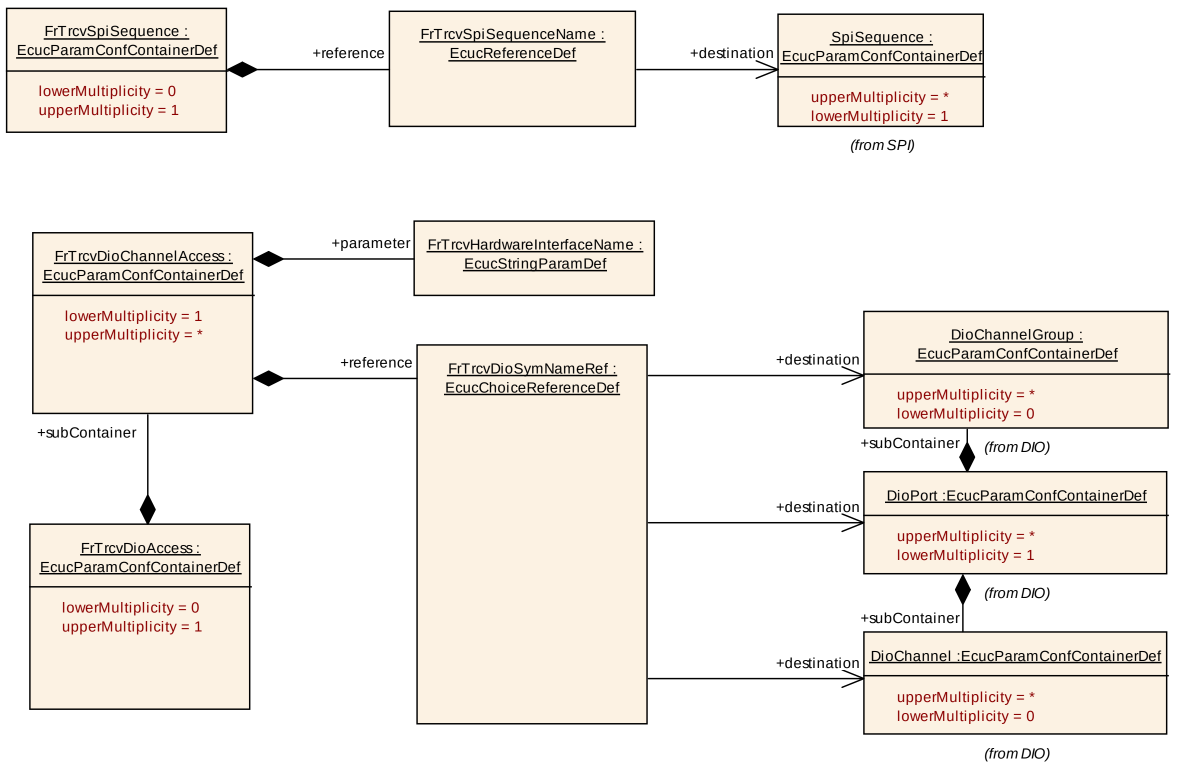
[SWS_FrTrcv_00334] ⌈The FlexRay Transceiver Driver uses these optional Interfaces:

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.

⌋ ( )
Configuration of the container FlexRayTransceiverDioAccess ECUC_FrTrcv_00145 enables the FlexRay Transceiver Driver to use the API of the DIO module.

Configuration of the container FlexRayTransceiverSPISequences FrTrcv427 enables the FlexRay Transceiver Driver to use the API of the SPI module.

ATTENTION: Either SPI or DIO must be supported depending on FlexRay Transceiver hardware


FrTrcvSpiSequence : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
FrTrcvSpiSequenceName : EcucReferenceDef
+destination
SpiSequence : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from SPI)
FrTrcvDioChannelAccess : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
FrTrcvHardwareInterfaceName : EcucStringParamDef
+reference
FrTrcvDioSymNameRef : EcucChoiceReferenceDef
+destination
DioChannelGroup : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
(from DIO)
+subContainer
DioPort :EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from DIO)
DioChannel :EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 0
(from DIO)
FrTrcvDioAccess : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1

[SWS_FrTrcv_00061] ⌈If FRTRCV_DEV_ERROR_DETECT is configured, the FlexRay Transceiver Driver uses the API of the DET module. ⌋ (SRS_BSW_00350)

8.9 Configurable interfaces

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

[SWS_FrTrcv_00475] ⌈If the optional configuration parameter FrTrcvDemReportErrorStatusConfiguration is provided in the global FlexRay Transceiver Driver configuration, the function defined by this configuration parameter shall be called instead of Dem_ReportErrorStatus with the same signature. ⌋ ( )

E.g. FrTrcv_ReportErrorStatus() could be configured and would be called instead of Dem_ReportErrorStatus()

[SWS_FrTrcv_00019] ⌈Configurability of optional functionality. Optional functionality of a Basic--SW component that is not required in the ECU shall be configurable at pre--compile--time (on/off).

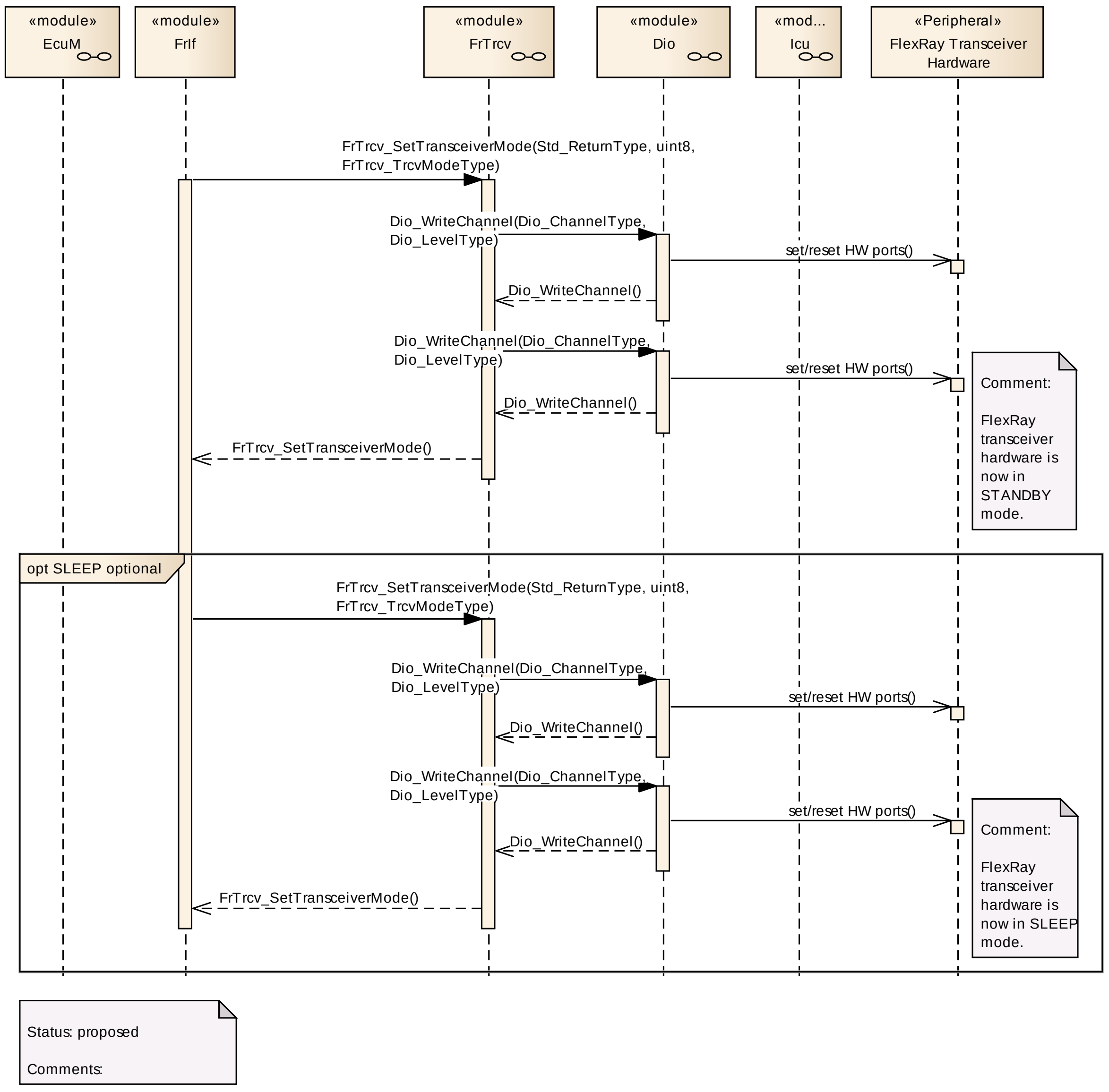
If branches of active stars using ECUC_FrTrcv_00357 are configured, these additional APIs shall be available:

- The API to enable branches SWS_FrTrcv_00443

- The API to disable branches SWS_FrTrcv_00442
- The API to detect errors of branches SWS_FrTrcv_00419

Please see SWS_FrTrcv_00393 and ECUC_FrTrcv_00456.⌋ (SRS_BSW_00171)

9 Sequence diagrams


«module» EcuM
«module» FrIf
«module» FrTrcv
«module» Dio
«mod... Icu
«Peripheral» FlexRay Transceiver Hardware
FrTrcv_SetTransceiverMode(Std_ReturnType, uint8, FrTrcv_TrcvModeType)
Dio_WriteChannel(Dio_ChannelType, Dio_LevelType)
set/reset HW ports()
Dio_WriteChannel()
Dio_WriteChannel(Dio_ChannelType, Dio_LevelType)
set/reset HW ports()
Dio_WriteChannel()
FrTrcv_SetTransceiverMode()
Comment: FlexRay transceiver hardware is now in STANDBY mode.
opt SLEEP optional
FrTrcv_SetTransceiverMode(Std_ReturnType, uint8, FrTrcv_TrcvModeType)
Dio_WriteChannel(Dio_ChannelType, Dio_LevelType)
set/reset HW ports()
Dio_WriteChannel()
Dio_WriteChannel(Dio_ChannelType, Dio_LevelType)
set/reset HW ports()
Dio_WriteChannel()
FrTrcv_SetTransceiverMode()
Comment: FlexRay transceiver hardware is now in SLEEP mode.
Status: proposed
Comments:

ATTENTION: These sequence charts are application examples only. They focus on interaction between the FlexRay transceiver driver (FrTrcv), FlexRay Interface (FrIf) and BSW module Dio. For details see [7] and [14]. 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 [13].

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

For details refer to the chapter 10.1 “Introduction to configuration specification” in *SWS_BSWGeneral.*

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

[SWS_FrTrcv_00314] ⌈VARIANT-PRE-COMPILE: Only parameters with "Pre-compile time" configuration are allowed in this variant. ⌋ ( )

[SWS_FrTrcv_00315] ⌈VARIANT-LINK-TIME: Only parameters with "Pre-compile time" and "Link time" are allowed in this variant. ⌋ ( )

[SWS_FrTrcv_00316] ⌈VARIANT-POST-BUILD: Parameters with "Pre-compile time", "Link time" and "Post-build time" are allowed in this variant. ⌋ ( )

[SWS_FrTrcv_00317] ⌈The FrTrcv module shall support pre compile time configuration. ⌋ (SRS_BSW_00397)

[SWS_FrTrcv_00318] ⌈The FrTrcv module shall not use link time parameters within the FlexRay Transceiver Driver. ⌋ ( )

[SWS_FrTrcv_00319] ⌈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.

[SWS_FrTrcv_00010] ⌈A configuration tool is used to generate the configuration data and code if any. ⌋ (SRS_BSW_00159)

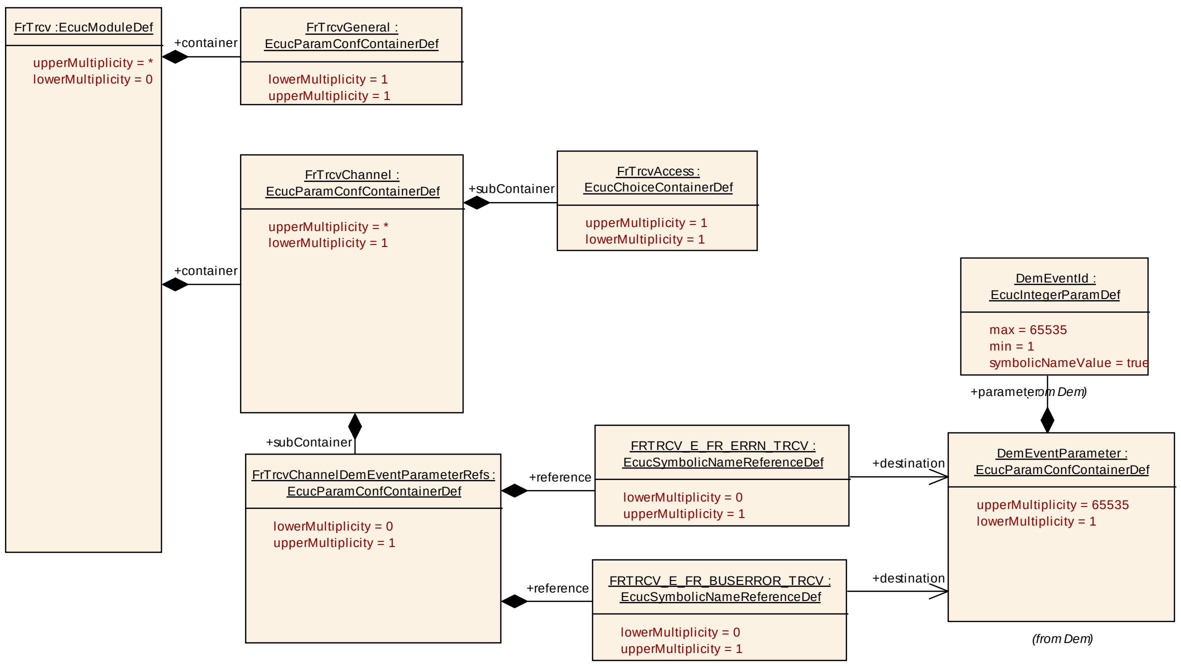
[SWS_FrTrcv_00018] ⌈Data for reconfiguration of AUTOSAR SW-Components⌋ (SRS_BSW_00170)

[SWS_FrTrcv_00047] ⌈All Basic Software Modules shall provide an XML file that contains the meta data which is required for the SW configuration and integration process. ⌋ (SRS_BSW_00334)

[SWS_FrTrcv_00225] ⌈Configuration Data for FlexRay Transceiver⌋ (SRS_Fr_05131)

[SWS_FrTrcv_00016] ⌈The configuration tool has to check the validity of the provided input data and the usability in the project context. ⌋ (SRS_BSW_00167)

[SWS_FrTrcv_00088] ⌈Containers shall have names.
The configuration of the transceiver is assembled in a container⌋ (SRS_BSW_00389)



10.2.3 FrTrcvGeneral

SWS Item	**ECUC_FrTrcv_00055 :**
Container Name	FrTrcvGeneral{FlexRayTransceiverDriverBasic}
Description	Container gives FlexRay transceiver driver basic information.
Configuration Parameters	

SWS Item	**ECUC_FrTrcv_00455 :**		
Name	FrTrcvDemReportErrorStatusConfiguration {FRTRCV_DEM_REPORT_ERROR_STATUS_CONFIGURATION}		
Description	Name of a C function which substitutes Dem_ReportErrorStatus.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00341 :**		
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00447 :**		
Name	FrTrcvErrorCheckDuringCommunication {FRTRCV_ERROR_CHECK_DURING_COMMUNICATION}		
Description	Enable a functionality to check transceiver's state during communication.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00446 :**		
Name	FrTrcvErrorCheckInInit {FRTRCV_ERROR_CHECK_IN_INIT}		
Description	Enable a functionality to check transceiver's state while initialization process of FrTrcv.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00342 :**
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	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00268 :**		
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	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

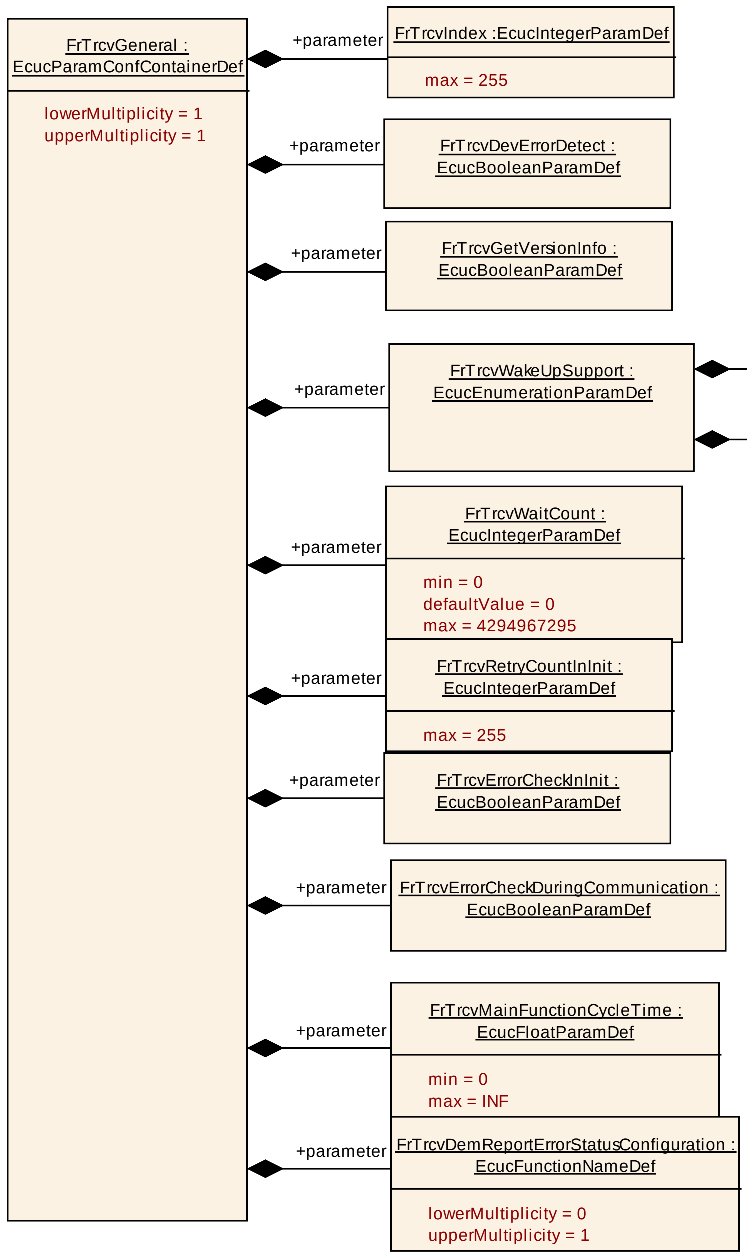
SWS Item	**ECUC_FrTrcv_00343 :**		
Name	FrTrcvMainFunctionCycleTime {FRTRCV_MAIN_FUNCTION_CYCLE_TIME}		
Description	Cyclic call time for function FrTrcvMainFunction in seconds. A call time of 0ms indicates no calls for this function. In this case function need not be present in compiled code.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00445 :**		
Name	FrTrcvRetryCountInInit {FRTRCV_RETRY_COUNT_IN_INIT}		
Description	Specifies the number of retry count when error occurs while initialization process of FrTrcv.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00353 :**		
Name	FrTrcvWaitCount		
Description	Wait count for transceiver state changes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	0		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00352 :**		
Name	FrTrcvWakeUpSupport {FRTRCV_GENERAL_WAKE_UP_SUPPORT}		
Description	Informs whether wake up is supported by polling or whether it is not supported. In case no wake up is supported by FlexRay transceiver hardware setting has to be always NO. Only in case wake up is supported by polling main function FlexRayTrcv_main has to be present in source code. In case of support for wake up either by polling wake up ability may be switched on or off for each channel of one FlexRay transceiver channel independently by FrTrcvWakeupByBusUsed.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRTRCV_WAKEUP_BY_POLLING	Wake up by polling	
	FRTRCV_WAKEUP_NOT_SUPPORTED	Wake up is not supported	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: FrTrcvWakeupByBusUsed		

No Included Containers


FrTrcvGeneral : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 1
+parameter
FrTrcvIndex :EcucIntegerParamDef
max = 255
FrTrcvDevErrorDetect : EcucBooleanParamDef
FrTrcvGetVersionInfo : EcucBooleanParamDef
FrTrcvWakeUpSupport : EcucEnumerationParamDef
+literal
FRTRCV_WAKEUP_BY_POLLING : EcucEnumerationLiteralDef
FRTRCV_WAKEUP_NOT_SUPPORTED : EcucEnumerationLiteralDef
FrTrcvWaitCount : EcucIntegerParamDef
min = 0
defaultValue = 0
max = 4294967295
FrTrcvRetryCountInInit : EcucIntegerParamDef
max = 255
FrTrcvErrorCheckInInit : EcucBooleanParamDef
FrTrcvErrorCheckDuringCommunication : EcucBooleanParamDef
FrTrcvMainFunctionCycleTime : EcucFloatParamDef
min = 0
max = INF
FrTrcvDemReportErrorStatusConfiguration : EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1

10.2.4 FrTrcvChannel

SWS Item	**ECUC_FrTrcv_00091 :**
Container Name	FrTrcvChannel{FlexRayTransceiverNode}
Description	Container gives FlexRay transceiver driver information about a single FlexRay transceiver channel. Any FlexRay transceiver driver has such FlexRay transceiver channels.
Configuration Parameters	

SWS Item	**ECUC_FrTrcv_00349 :**
Name	FrTrcvChannelId {FRTRCV_NODE_ID}
Description	Unique identifier of the FlexRay Transceiver Channel.

Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00355 :**		
Name	FrTrcvChannelUsed {FRTRCV_CHANNEL_USED}		
Description	Shall the related FlexRay transceiver channel be used?		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00346 :**		
Name	FrTrcvControlsPowerSupply {FRTRCV_CONTROLS_POWER_SUPPLY}		
Description	Is ECU power supply controlled by this transceiver?		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00456 :**		
Name	FrTrcvEnableInterruptCallout {FRTRCV_ENABLE_INTERRUPT_CALLOUT}		
Description	This parameter defines the existence and the name of a callout function that enables the interrupt pin for the wakeup. If this parameter is omitted no callout shall take place.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

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

SWS Item	**ECUC_FrTrcv_00347 :**		
Name	FrTrcvInitState {FRTRCV_INIT_STATE}		
Description	State of FlexRay transceiver after power on. ImplementationType: FrTrcv_TrcvModeType		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRTRCV_TRCVMODE_NORMAL	Normal operation mode	
	FRTRCV_TRCVMODE_RECEIVEONLY	Receive only mode	
	FRTRCV_TRCVMODE_SLEEP	Sleep operation mode	
	FRTRCV_TRCVMODE_STANDBY	Standby operation mode	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00348 :**		
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	EcucEnumerationParamDef		
Range	FR_10M	10.0 MBaud	
	FR_2M5	2.5 MBaud	
	FR_5M0	5.0 MBaud	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00350 :**
Name	FrTrcvWakeupByBusUsed {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	EcucBooleanParamDef

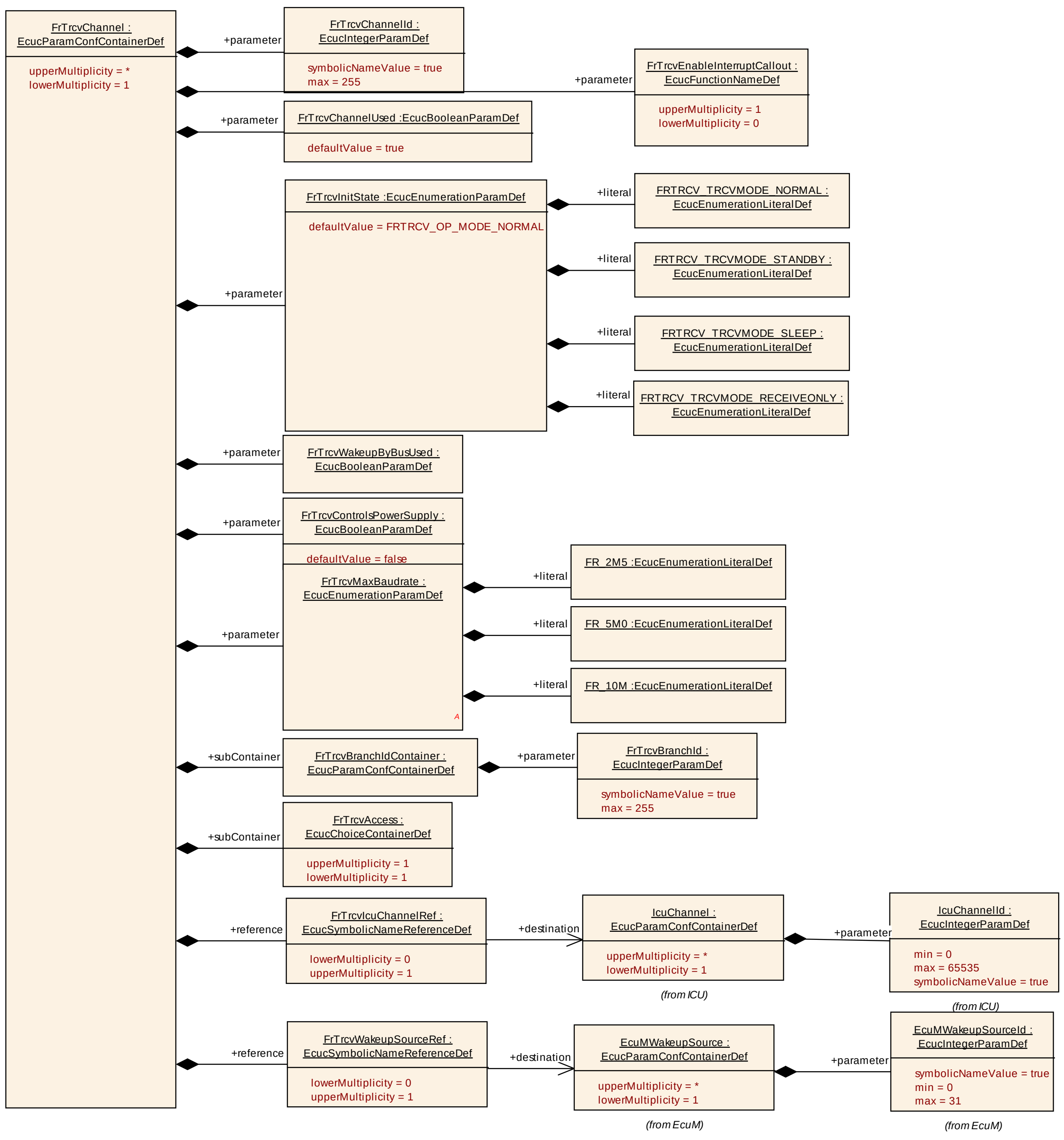
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: FRTRCV_WAKEUP_POLLING		

SWS Item	**ECUC_FrTrcv_00384 :**		
Name	FrTrcvIcuChannelRef {FRTRCV_ICU_CHANNEL_REF}		
Description	Reference to the IcuChannel to enable/disable the interrupts for wakeups.		
Multiplicity	0..1		
Type	Symbolic name reference to [IcuChannel]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_FrTrcv_00269 :**		
Name	FrTrcvWakeupSourceRef {FRTRCV_WAKEUP_SOURCE_REF}		
Description	Reference to a wakeup source in the EcuM configuration. If FrTrcvWakeUpSupport is configured as FRTRCV_WAKEUP_NOT_SUPPORTED the FrTrcvWakeupSourceRef is not needed. Implementation Type: reference to EcuM_WakeupSourceType		
Multiplicity	0..1		
Type	Symbolic name reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: FrTrcvWakeUpSupport		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrTrcvAccess	1	--
FrTrcvBranchIdContainer	1	Only one SymbolicNameValue can be defined per container. Therefore this container is necessary.
FrTrcvChannelDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced

		DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.


FrTrcvChannel : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
FrTrcvChannelId : EcucIntegerParamDef
symbolicNameValue = true
max = 255
+parameter
FrTrcvEnableInterruptCallout : EcucFunctionNameDef
upperMultiplicity = 1
lowerMultiplicity = 0
+parameter
FrTrcvChannelUsed :EcucBooleanParamDef
defaultValue = true
FrTrcvInitState :EcucEnumerationParamDef
defaultValue = FRTRCV_OP_MODE_NORMAL
+parameter
+literal
FRTRCV_TRCVMODE_NORMAL : EcucEnumerationLiteralDef
+literal
FRTRCV_TRCVMODE_STANDBY : EcucEnumerationLiteralDef
+literal
FRTRCV_TRCVMODE_SLEEP : EcucEnumerationLiteralDef
+literal
FRTRCV_TRCVMODE_RECEIVEONLY : EcucEnumerationLiteralDef
+parameter
FrTrcvWakeupByBusUsed : EcucBooleanParamDef
+parameter
FrTrcvControlsPowerSupply : EcucBooleanParamDef
defaultValue = false
FrTrcvMaxBaudrate : EcucEnumerationParamDef
+parameter
+literal
FR_2M5 :EcucEnumerationLiteralDef
+literal
FR_5M0 :EcucEnumerationLiteralDef
+literal
FR_10M :EcucEnumerationLiteralDef
+subContainer
FrTrcvBranchIdContainer : EcucParamConfContainerDef
+parameter
FrTrcvBranchId : EcucIntegerParamDef
symbolicNameValue = true
max = 255
+subContainer
FrTrcvAccess : EcucChoiceContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+reference
FrTrcvIcuChannelRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
IcuChannel : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from ICU)
+parameter
IcuChannelId : EcucIntegerParamDef
min = 0
max = 65535
symbolicNameValue = true
(from ICU)
+reference
FrTrcvWakeupSourceRef : EcucSymbolicNameReferenceDef
lowerMultiplicity = 0
upperMultiplicity = 1
+destination
EcuMWakeupSource : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
(from EcuM)
+parameter
EcuMWakeupSourceId : EcucIntegerParamDef
symbolicNameValue = true
min = 0
max = 31
(from EcuM)

10.2.5 FrTrcvChannelDemEventParameterRefs

SWS Item	**ECUC_FrTrcv_00450 :**
Container Name	FrTrcvChannelDemEventParameterRefs

Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**ECUC_FrTrcv_00453 :**		
Name	FRTRCV_E_FR_BUSERROR_TRCV {FRTRCV_E_FR_BUSERROR_TRCV}		
Description	Reference to configured DEM event to report "Error Status of Class B (SPI) transceiver bus errors where TrcvIdx is the transceiver index"		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Dem		

SWS Item	**ECUC_FrTrcv_00452 :**		
Name	FRTRCV_E_FR_ERRN_TRCV {FRTRCV_E_FR_ERRN_TRCV}		
Description	Reference to configured DEM event to report "Error Status of Class A (GPIO) transceiver"		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Dem		

No Included Containers

10.2.6 FrTrcvBranchIdContainer

SWS Item	**ECUC_FrTrcv_00357 :**
Container Name	FrTrcvBranchIdContainer
Description	Only one SymbolicNameValue can be defined per container. Therefore this container is necessary.
Configuration Parameters	

SWS Item	**ECUC_FrTrcv_00356 :**		
Name	FrTrcvBranchId {FRTRCV_BRANCH_ID}		
Description	Unique branch id. It is used by CDDs and internally.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.7 FrTrcvAccess

SWS Item	**ECUC_FrTrcv_00454 :**
Choice container Name	FrTrcvAccess
Description	--

Container Choices		
Container Name	*Multiplicity*	*Scope / Dependency*
FrTrcvDioAccess	0..1	Container gives FR transceiver driver information about accessing ports and port pins. If a FR transceiver hardware has no Dio interface, there is no instance of this container.
FrTrcvSpiSequence	0..1	Container gives FlexRay transceiver driver information about one SPI sequence. One SPI sequence used by FlexRay transceiver driver is in exclusive use for it. No other driver is allowed to access this sequence. FlexRay transceiver driver may use one sequence to access n FlexRay transceiver hardwares chips of the same type or n sequences are used to access one single FlexRay transceiver hardware chip. If a FlexRay transceiver hardware has no SPI interface, there is no instance of this container.

10.2.8 FrTrcvDioAccess

SWS Item	**ECUC_FrTrcv_00145 :**

Container Name	FrTrcvDioAccess{FrTransceiverDioAccess}
Description	Container gives FR transceiver driver information about accessing ports and port pins. If a FR transceiver hardware has no Dio interface, there is no instance of this container.
Configuration Parameters	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrTrcvDioChannelAccess	1..*	In this Container the relation between FR transceiver hardware pin names and Dio port access information is given.

10.2.9 FrTrcvDioChannelAccess

SWS Item	**ECUC_FrTrcv_00471 :**
Container Name	FrTrcvDioChannelAccess{FrTrcvDioChannelAccess}
Description	In this Container the relation between FR transceiver hardware pin names and Dio port access information is given.
Configuration Parameters	

SWS Item	**ECUC_FrTrcv_00150 :**		
Name	FrTrcvHardwareInterfaceName {FRTRCV_HARDWARE_INTERFACE_NAME}		
Description	FR transceiver hardware interface name. It is typically the name of a pin. From a Dio point of view it is either a port, a single channel or a channel group. Depending on this fact either FRTRCV_DIO_PORT_SYMBOLIC_NAME or FRTRCV_DIO_CHANNEL_SYMBOLIC_NAME or FRTRCV_DIO_CHANNEL_GROUP_SYMBOLIC_NAME shall reference a Dio configuration. The FR transceiver driver implementation description shall list up this name for the appropriate FR transceiver hardware.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrTrcv_00149 :**
Name	FrTrcvDioSymNameRef
Description	Choice Reference to a DIO Port, DIO Channel or DIO Channel Group. This reference replaces the

	FRTRCV_DIO_PORT_SYM_NAME, FRTRCV_DIO_CHANNEL_SYM_NAME and FRTRCV_DIO_GROUP_SYM_NAME references in the Fr Trcv SWS.		
Multiplicity	1		
Type	Choice reference to [DioChannel , DioChannelGroup , DioPort]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

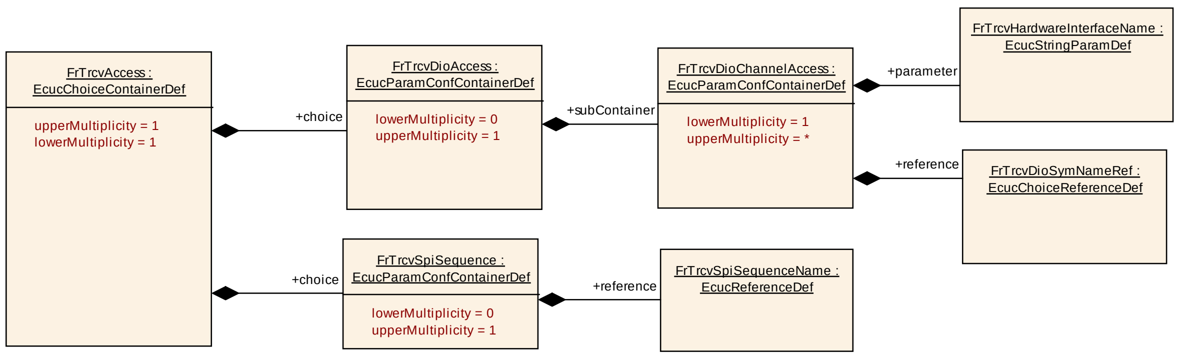
No Included Containers

10.2.10 FrTrcvSpiSequence

SWS Item	**ECUC_FrTrcv_00144 :**		
Container Name	FrTrcvSpiSequence{FlexRayTransceiverSPISequences}		
Description	Container gives FlexRay transceiver driver information about one SPI sequence. One SPI sequence used by FlexRay transceiver driver is in exclusive use for it. No other driver is allowed to access this sequence. FlexRay transceiver driver may use one sequence to access n FlexRay transceiver hardwares chips of the same type or n sequences are used to access one single FlexRay transceiver hardware chip. If a FlexRay transceiver hardware has no SPI interface, there is no instance of this container.		
Configuration Parameters			

SWS Item	**ECUC_FrTrcv_00151 :**		
Name	FrTrcvSpiSequenceName {FRTRCV_SPI_SEQUENCE_NAME}		
Description	Reference to a Spi sequence configuration container.		
Multiplicity	1		
Type	Reference to [SpiSequence]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: SpiSequence		

No Included Containers


FrTrcvAccess : EcucChoiceContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+choice
FrTrcvDioAccess : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+subContainer
FrTrcvDioChannelAccess : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
+parameter
FrTrcvHardwareInterfaceName : EcucStringParamDef
+reference
FrTrcvDioSymNameRef : EcucChoiceReferenceDef
+choice
FrTrcvSpiSequence : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
FrTrcvSpiSequenceName : EcucReferenceDef

10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS_BSWGeneral.*

11 Not applicable requirements

[SWS_FrTrcv_00478] ⌈These requirements are not applicable to this specification.⌋ (SRS_BSW_00005, SRS_BSW_00006, SRS_BSW_00007, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00161, SRS_BSW_00164, SRS_BSW_00168, SRS_BSW_00304, SRS_BSW_00306, SRS_BSW_00307, SRS_BSW_00308, SRS_BSW_00309, SRS_BSW_00312, SRS_BSW_00321, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00328, SRS_BSW_00330, SRS_BSW_00333, SRS_BSW_00336, SRS_BSW_00341, SRS_BSW_00342, SRS_BSW_00344, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00370, SRS_BSW_00378, SRS_BSW_00382, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00386, SRS_BSW_00392, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00400, SRS_BSW_00401, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00410, SRS_BSW_00413, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00420, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00429, SRS_BSW_00431, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00434, SRS_Fr_05000, SRS_Fr_05001, SRS_Fr_05002, SRS_Fr_05003, SRS_Fr_05004, SRS_Fr_05005, SRS_Fr_05006, SRS_Fr_05007, SRS_Fr_05009, SRS_Fr_05010, SRS_Fr_05011, SRS_Fr_05012, SRS_Fr_05013, SRS_Fr_05015, SRS_Fr_05016, SRS_Fr_05018, SRS_Fr_05019, SRS_Fr_05022, SRS_BSW_05023, SRS_Fr_05024, SRS_BSW_05025, SRS_Fr_05027, SRS_Fr_05031, SRS_Fr_05033, SRS_Fr_05034, SRS_BSW_05035, SRS_BSW_05038, SRS_Fr_05039, SRS_BSW_05040, SRS_BSW_05041, SRS_Fr_05042, SRS_Fr_05044, SRS_BSW_05045, SRS_Fr_05046, SRS_Fr_05047, SRS_Fr_05048, SRS_Fr_05049, SRS_Fr_05050, SRS_Fr_05051, SRS_Fr_05052, SRS_Fr_05053, SRS_Fr_05055, SRS_Fr_05056, SRS_Fr_05058, SRS_Fr_05059, SRS_Fr_05060, SRS_Fr_05063, SRS_Fr_05064, SRS_Fr_05065, SRS_Fr_05066, SRS_BSW_05067, SRS_BSW_05068, SRS_BSW_05069, SRS_Fr_05072, SRS_Fr_05073, SRS_Fr_05074, SRS_Fr_05076, SRS_Fr_05077, SRS_BSW_05078, SRS_Fr_05079, SRS_BSW_05082, SRS_BSW_05083, SRS_BSW_05084, SRS_BSW_05085, SRS_Fr_05088, SRS_Fr_05089, SRS_Fr_05090, SRS_Fr_05093, SRS_Fr_05095, SRS_Fr_05096, SRS_Fr_05097, SRS_BSW_05101, SRS_BSW_05102, SRS_BSW_05104, SRS_Fr_05106, SRS_BSW_05107, SRS_Fr_05109, SRS_BSW_05111, SRS_BSW_05113, SRS_Fr_05114, SRS_Fr_05115, SRS_Fr_05116, SRS_Fr_05117, SRS_Fr_05120, SRS_Fr_05121, SRS_Fr_05123, SRS_BSW_05124, SRS_Fr_05125, SRS_Fr_05126, SRS_Fr_05130, SRS_BSW_05153, SRS_BSW_05154, SRS_BSW_05155, SRS_Fr_05156, SRS_Fr_05157, SRS_Fr_05158, SRS_BSW_05162, SRS_BSW_05163, SRS_BSW_05164, SRS_BSW_05165, SRS_Fr_05169, SRS_Fr_05170, SRS_Fr_05171, SRS_BSW_05172, SRS_BSW_05173, SRS_Fr_05174, SRS_Fr_05175, SRS_BSW_05200, SRS_BSW_05201, SRS_BSW_05202, SRS_BSW_05204, SRS_BSW_05205, SRS_BSW_05206, SRS_BSW_05207, SRS_BSW_05208, SRS_BSW_05209, SRS_BSW_05210, SRS_BSW_05211, SRS_BSW_05215)