Document Title	Specification of FlexRay Transceiver Driver
Document Category	Software specification (SWS)
Document UID	74
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• VersionInfo API removed • Extended Prod. Errors reclassified
2025-11-26	R25-11	AUTOSAR Release Management	• Removed requirement
2024-11-27	R24-11	AUTOSAR Release Management	• Clarification of multicore support. • Editorial updates
2023-11-23	R23-11	AUTOSAR Release Management	• editorial cleanup. • withAuto tag set to true for Index and ID configuration parameters. • Editorial update of main function definition table
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• Header file cleanup
2020-11-30	R20-11	AUTOSAR Release Management	• Modeling of Development Errors, Runtime Errors, and Transient Faults • Improve the structure of the 'error sections' of the SWS documents



△

2019-11-28	R19-11	AUTOSAR Release Management	• Incorporation of validation results for CONC_639 • Fix inconsistent renaming of general types headers • Bus-independent solution regarding channel states upon initialization introduced • Periodic Error Detection in Bus Driver added • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Incorporation of concept 639 MCALMulticoreDistribution (Draft) • Renaming of initialization error to FRTRCV_E_UNINIT
2017-12-08	4.3.1	AUTOSAR Release Management	• Introduction of Default Error Tracer • Introduction of runtime errors
2016-11-30	4.3.0	AUTOSAR Release Management	• Icu APIs are used to activate/de-activate the ISR that indicates a wakeup • Clarification in configuration of SPI sequence • Correction of mainfunction period
2015-07-31	4.2.2	AUTOSAR Release Management	• Redesigned extended production error chapter, updated to default error tracer • Added a (dummy) configuration parameter to the initialization interface • Debugging support marked as obsolete • Removed chapter(s) on change documentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Reworked development and production errors according to the new SWS_BSWGeneral • Supports multiple branch ids per transceiver • Supports new busy wait time service
2014-03-31	4.1.3	AUTOSAR Release Management	• Adapted requirement identifier prefixes • Deleted some redundant software specification items

▽

△

2013-10-31	4.1.2	AUTOSAR Release Management	• Simplified schedule to pre compile fixed cyclic • Reduced run time configuration checks • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• Reworked according to the new SWS_BSWGeneral
2011-12-22	4.0.3	AUTOSAR Administration	• Improved interrupt support by ICU • Improved production error concept
2010-09-30	3.1.5	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
2010-02-02	3.1.4	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. • Renamed API FrTrcv_Cbk_WakeupByTransceiver to FrTrcv_CheckWakeupByTransceiver • Legal disclaimer revised

▽

△

2008-08-13	3.1.1	AUTOSAR Administration	• Chapter 9 regenerated from BSW UML Model
2008-02-01	3.0.2	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	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
2007-01-24	2.1.15	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
2006-05-16	2.0	AUTOSAR Administration	• Initial release

Disclaimer

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

The material contained in this work is protected by copyright and other types of intellectual property rights. The commercial exploitation of the material contained in this work requires a license to such intellectual property rights.

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

The work has been developed for automotive applications only. It has neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 9
1.1 Goal of FlexRay transceiver driver . . . 10
1.2 Explicitly uncovered FlexRay Transceiver Functionality . . . 10
1.3 Active Stars . . . 10
2 Acronyms and Abbreviations 11
3 Related documentation 13
3.1 Input documents & related standards and norms . . . 13
3.2 Related specification . . . 13
4 Constraints and assumptions 14
4.1 Limitations . . . 14
4.2 Applicability to car domains . . . 14
5 Dependencies to other modules 15
5.1 File structure . . . 16
5.1.1 Naming convention for transceiver driver implementation . . . 16
5.1.2 Code file structure . . . 17
5.1.3 Header file structure . . . 17
6 Requirements Tracing 19
7 Functional specification 22
7.1 AUTOSAR FlexRay Transceiver Operation Mode Model . . . 22
7.2 FlexRay transceiver hardware operation modes . . . 24
7.2.1 Temporary "Go-To-Sleep" Mode . . . 24
7.2.2 "Active Star" Mode . . . 24
7.3 Enabling/Disabling wakeup notification . . . 25
7.4 Error Classification . . . 25
7.4.1 Development Errors . . . 25
7.4.2 Runtime Errors . . . 26
7.4.3 Production Errors . . . 26
7.4.3.1 FRTRCV_E_FR_ERRN_TRCV_<TrcvIdx> . . . 26
7.4.3.2 FRTRCV_E_FR_BUSERROR_TRCV_<TrcvIdx> . . . 26
7.5 Preconditions for driver initialization . . . 29
7.6 Instance concept . . . 30
7.7 Wake Up Support . . . 30
7.7.1 Power-on: . . . 30
7.7.2 Active wakeup: . . . 30
7.7.3 Passive wakeup: . . . 31
7.8 Security Events . . . 31

8 API specification 32
8.1 Imported types . . . 32
8.2 Type definitions . . . 32
8.2.1 FrTrcv_ConfigType . . . 33
8.2.2 FrTrcv_TrcvModeType . . . 33
8.2.3 FrTrcv_TrcvWUReasonType . . . 34
8.3 Function definitions . . . 35
8.3.1 FrTrcv_Init . . . 36
8.3.2 FrTrcv_SetTransceiverMode . . . 38
8.3.3 FrTrcv_GetTransceiverMode . . . 41
8.3.4 FrTrcv_GetTransceiverWUReason . . . 42
8.3.5 FrTrcv_ClearTransceiverWakeup . . . 43
8.3.6 FrTrcv_CheckWakeupByTransceiver . . . 45
8.3.7 FrTrcv_GetTransceiverError . . . 47
8.3.8 FrTrcv_DisableTransceiverBranch . . . 49
8.3.9 FrTrcv_EnableTransceiverBranch . . . 50
8.4 Scheduled functions . . . 51
8.4.1 FrTrcv_MainFunction . . . 52
8.5 Expected interfaces . . . 53
8.5.1 Mandatory interfaces . . . 53
8.5.2 Optional interfaces . . . 54
8.5.3 Configurable interfaces . . . 55
8.6 Service Interfaces . . . 56
9 Sequence diagrams 57
10 Configuration specification 58
10.1 How to read this chapter . . . 58
10.2 Containers and configuration parameters . . . 58
10.2.1 General configuration requirements . . . 58
10.2.2 FrTrcv . . . 59
10.2.3 FrTrcvGeneral . . . 60
10.2.4 FrTrcvChannel . . . 65
10.2.5 FrTrcvChannelDemEventParameterRefs . . . 71
10.2.6 FrTrcvBranchIdContainer . . . 72
10.2.7 FrTrcvAccess . . . 72
10.2.8 FrTrcvDioAccess . . . 73
10.2.9 FrTrcvDioChannelAccess . . . 74
10.2.10 FrTrcvSpiSequence . . . 75
10.3 Published Information . . . 76
A Not applicable requirements 77

B Change history of AUTOSAR traceable items 78
B.1 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 78
B.1.1 Added Specification Items in R26-11 . . . 78
B.1.2 Changed Specification Items in R26-11 . . . 78
B.1.3 Deleted Specification Items in R26-11 . . . 78
B.2 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 78
B.2.1 Added Specification Items in R25-11 . . . 78
B.2.2 Changed Specification Items in R25-11 . . . 79
B.2.3 Deleted Specification Items in R25-11 . . . 79
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 79
B.3.1 Added Specification Items in R24-11 . . . 79
B.3.2 Changed Specification Items in R24-11 . . . 79
B.3.3 Deleted Specification Items in R24-11 . . . 80
B.4 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 80
B.4.1 Added Specification Items in R23-11 . . . 80
B.4.2 Changed Specification Items in R23-11 . . . 84
B.4.3 Deleted Specification Items in R23-11 . . . 84
B.5 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 85
B.5.1 Added Specification Items in R22-11 . . . 85
B.5.2 Changed Specification Items in R22-11 . . . 85
B.5.3 Deleted Specification Items in R22-11 . . . 86
B.5.4 Added Constraints in R22-11 . . . 86
B.5.5 Changed Constraints in R22-11 . . . 86
B.5.6 Deleted Constraints in R22-11 . . . 86

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module FlexRay Transceiver Driver.

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.

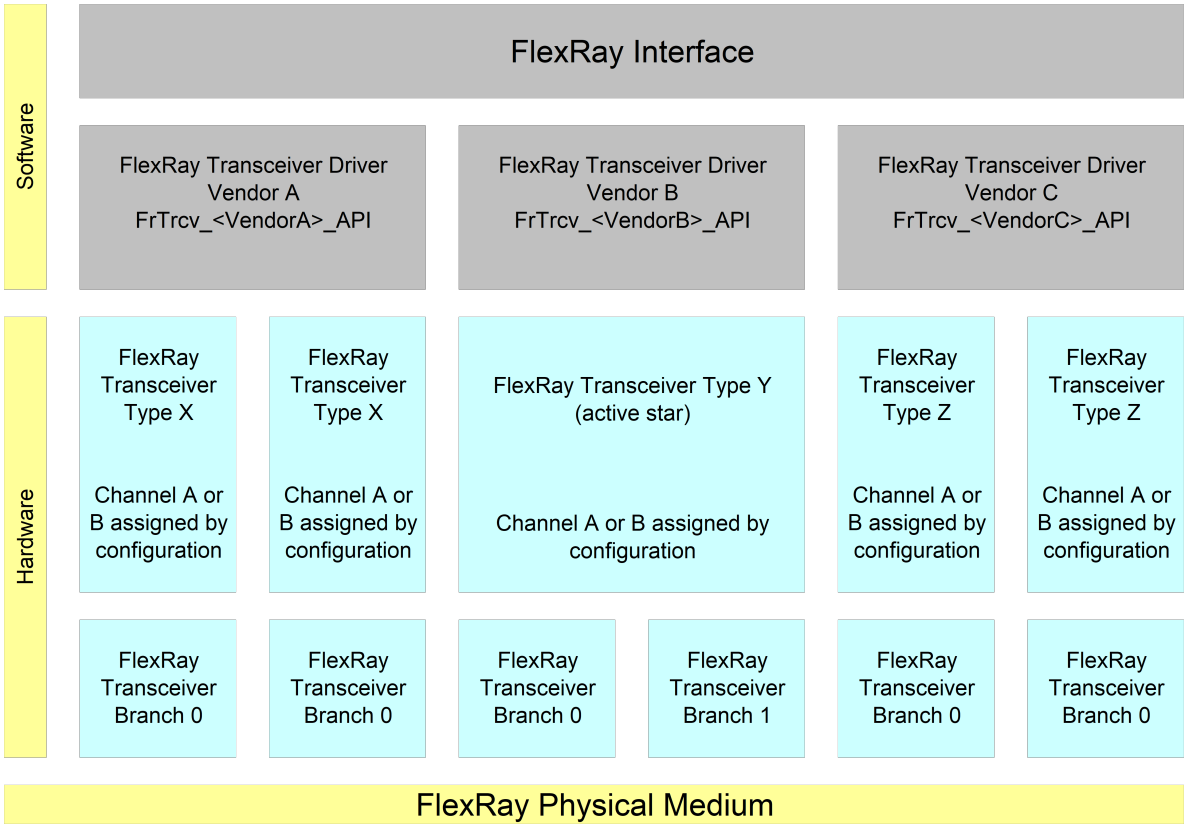

Figure 1.1: Description of the basic structure of the FlexRay Stack

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 [1, FlexRay EPL].

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 [1, FlexRay EPL] and provides topology information of branches.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the FlexRay Transceiver Driver module that are not included in the [2, AUTOSAR glossary].

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 [3] for details
DEM/Dem	Diagnostic Event Manager
DET/Det	Default 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 [4] 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
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

Table 2.1: Acronyms and abbreviations used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] FlexRay Communications System Protocol Specification V2.1
http://www.flexray.com/

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] Specification of Communication Manager
AUTOSAR_CP_SWS_COMManager

[4] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[5] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[6] Requirements on FlexRay
AUTOSAR_CP_RS_FlexRay

[7] System Template
AUTOSAR_CP_TPS_SystemTemplate

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [5, 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]

Upstream requirements: SRS_Fr_05138

⌈The FlexRay Transceiver Driver shall use the APIs of underlying DIO drivers synchronously.⌋

[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]

Upstream requirements: SRS_Fr_05152

⌈The FlexRay Transceiver Driver shall handle the transceiver-specific timing requirements internally.⌋

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 Default Error Tracer on 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]

Upstream requirements: SRS_BSW_00347

⌈A FlexRay Transceiver Driver implementation may support different FlexRay Transceiver hardware.⌋

[SWS_FrTrcv_00021]

Upstream requirements: SRS_BSW_00300

⌈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⌋

5.1.2 Code file structure

The FrTrcv module consists of the following code files:

[SWS_FrTrcv_00033]

Upstream requirements: SRS_BSW_00314

⌈FrTrcv.c is the implementation general C file. It does not contain interrupt routines.⌋

[SWS_FrTrcv_00057] Pre-compile-time configuration

Upstream requirements: SRS_BSW_00345

⌈

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.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.⌋

[SWS_FrTrcv_00117] Separate C-Files for pre-compile time configuration parameters

Upstream requirements: SRS_BSW_00419

⌈

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.⌋

5.1.3 Header file structure

[SWS_FrTrcv_00022]

Upstream requirements: SRS_BSW_00301

⌈All AUTOSAR Basic Software Modules shall only import the necessary information (i.e. header files) that is required to fulfill the modules functional requirements.⌋

[SWS_FrTrcv_00023]

Upstream requirements: SRS_BSW_00302

⌈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.⌋

6 Requirements Tracing

Requirement	Description	Satisfied by
[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_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_FrTrcv_00016]
[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]	Unique Name of Basic Software Modules	[SWS_FrTrcv_00021]
[SRS_BSW_00301]	All AUTOSAR Basic Software Modules shall only import the necessary information	[SWS_FrTrcv_00022]
[SRS_BSW_00302]	All AUTOSAR Basic Software Modules shall only export information needed by other modules	[SWS_FrTrcv_00023]
[SRS_BSW_00314]	All internal driver modules shall separate the interrupt frame definition from the service routine	[SWS_FrTrcv_00033]
[SRS_BSW_00327]	Error values naming convention	[SWS_FrTrcv_00041]
[SRS_BSW_00331]	All Basic Software Modules shall strictly separate error and status information	[SWS_FrTrcv_00045]
[SRS_BSW_00334]	Machine readable module description	[SWS_FrTrcv_00047]
[SRS_BSW_00335]	Status values naming convention	[SWS_FrTrcv_00048]
[SRS_BSW_00345]	BSW Modules shall support pre-compile configuration	[SWS_FrTrcv_00057]
[SRS_BSW_00347]	A Naming seperation of different instances of BSW drivers shall be in place	[SWS_FrTrcv_00059]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_FrTrcv_00061]
[SRS_BSW_00357]	For success/failure of an API call a standard return type shall be defined	[SWS_FrTrcv_00064]



△

Requirement	Description	Satisfied by
[SRS_BSW_00358]	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	[SWS_FrTrcv_00065]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_FrTrcv_00069]
[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_00385]	List possible error notifications	[SWS_FrTrcv_00084]
[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_00393]	Parameters shall have a range	[SWS_FrTrcv_00092]
[SRS_BSW_00395]	The Basic Software Module specifications shall list all configuration parameter dependencies	[SWS_FrTrcv_00094]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_FrTrcv_00487]
[SRS_BSW_00419]	If a pre-compile time configuration parameter is implemented as `const` it should be placed into a separate c-file	[SWS_FrTrcv_00117]
[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_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_00458]	Classification of production errors	[SWS_FrTrcv_00488] [SWS_FrTrcv_00489]
[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_05134]	The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack.	[SWS_FrTrcv_00228]

▽

△

Requirement	Description	Satisfied by
[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.	[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 same moment the transition to standby/sleep is executed by the driver.	[SWS_FrTrcv_00234]
[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_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_05203]	The Error Information in Bus Driver shall be available	[SWS_FrTrcv_00436]
[SRS_Fr_05212]	The Errors in Bus Driver shall be detected and notified	[SWS_FrTrcv_00412]
[SRS_Fr_05213]	The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly	[SWS_FrTrcv_00415]
[SRS_Fr_05214]	FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality	[SWS_FrTrcv_00414]

Table 6.1: Requirements Tracing

7 Functional specification

[SWS_FrTrcv_00485] ⌈The FlexRay Tranceiver Driver shall use the Time service Tm_BusyWait1us16bit to realize the wait time for transceiver state changes.⌋

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 set each FlexRay Transceiver to SLEEP or STANDBY mode during the driver initialization, depending on the configuration of FrTrcvInitState.⌋

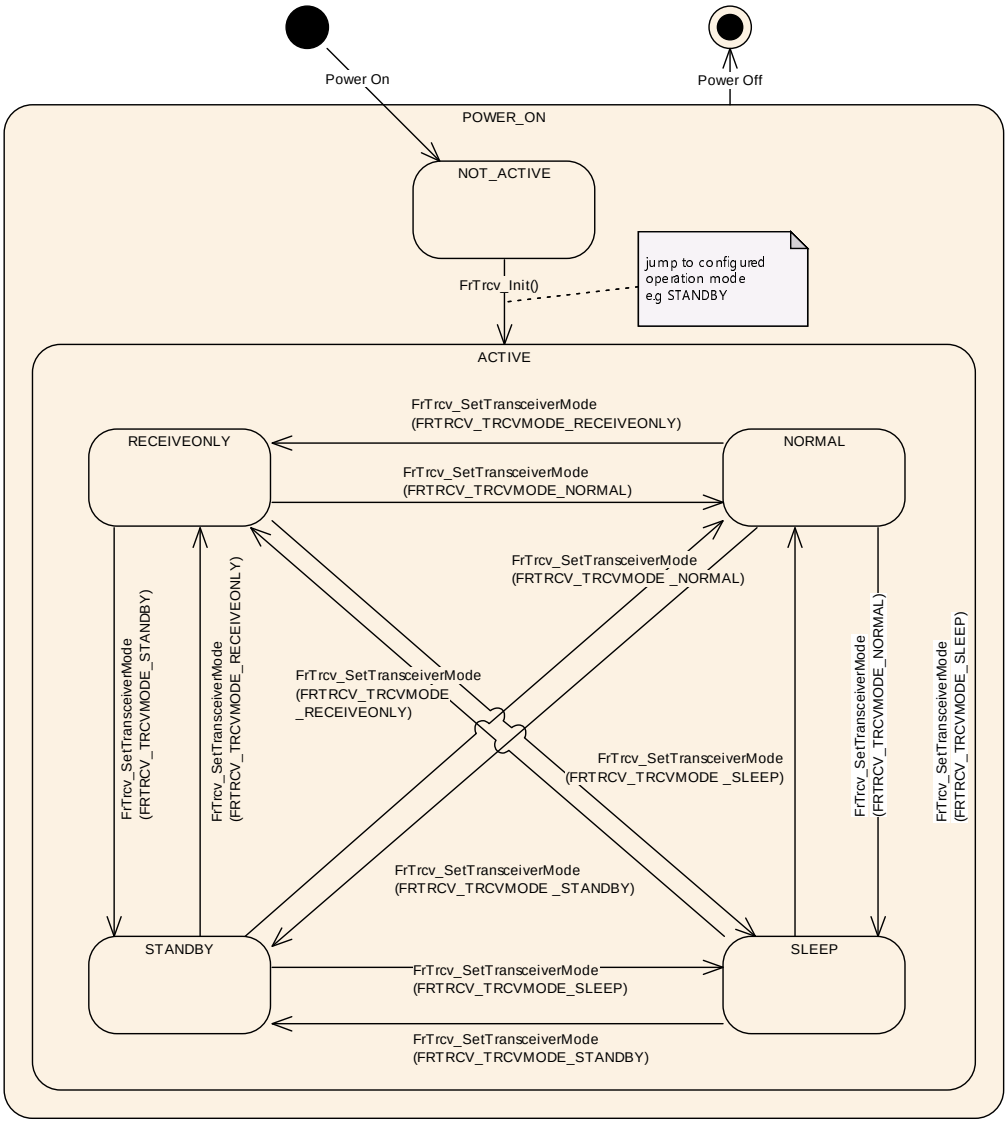

Figure 7.1: FlexRay Transceiver operation modes

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 [1, FlexRay EPL] 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 Enabling/Disabling wakeup notification

[SWS_FrTrcv_00490] ⌈FrTrcv driver shall use the following APIs provided by ICU driver, to enable and disable the wakeup event notification:

- Icu_EnableNotification
- Icu_DisableNotification

FrTrcv driver shall enable/disable ICU channels only if reference is configured for the parameter FrTrcvIcuChannelRef.⌋

FrTrcv driver shall ensure the following to avoid the loss of wakeup events:

[SWS_FrTrcv_00491] ⌈It shall enable the ICU channels when the transceiver transitions to the Standby mode (FRTRCV_STANDBY).⌋

[SWS_FrTrcv_00492] ⌈It shall disable the ICU channels when the transceiver transitions to the Normal mode (FRTRCV_NORMAL).⌋

7.4 Error Classification

Chapter [5, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

Based on this foundation, the following section specifies particular errors arranged in the respective subsections below.

7.4.1 Development Errors

[SWS_FrTrcv_00085] Definition of development errors in module FrTrcv ⌈

Type of error	*Related error code*	*Error value*
API service called with wrong parameter (FlexRay transceiver index out of range)	FRTRCV_E_FR_INVALID_TRCVIDX	0x01
API service called with wrong parameter (FlexRay transceiver branch index out of range)	FRTRCV_E_FR_INVALID_BRANCHIDX	0x02
API Service used without initialization	FRTRCV_E_UNINIT	0x10
API service called passing a NULL pointer as a parameter	FRTRCV_E_PARAM_POINTER	0x15
Invalid configuration set selection	FRTRCV_E_INIT_FAILED	0x17

⌋

7.4.2 Runtime Errors

[SWS_FrTrcv_91001] Definition of runtime errors in module FrTrcv ⌈

Type of error	*Related error code*	*Error value*
API service called in wrong transceiver operation mode	FRTRCV_E_FR_TRCV_NOT_STANDBY	0x11
API service called in wrong transceiver operation mode	FRTRCV_E_FR_TRCV_NOT_NORMAL	0x12
API service called in wrong transceiver operation mode	FRTRCV_E_FR_TRCV_NOT_SLEEP	0x13
API service called in wrong transceiver operation mode	FRTRCV_E_FR_TRCV_NOT_RECEIVEONLY	0x14
No/incorrect communication to transceiver	FRTRCV_E_FR_NO_CONTROL_TRCV	0x16

⌋

7.4.3 Production Errors

7.4.3.1 FRTRCV_E_FR_ERRN_TRCV_<TrcvIdx>

[SWS_FrTrcv_00489] Error Status of Class A (GPIO) transceiver where TrcvIdx is the transceiver index.

Upstream requirements: SRS_BSW_00458

⌈

Diagnostic Event (Error Name)	FRTRCV_E_FR_ERRN_TRCV_TrcvIdx
Description	Error Status of Class A general purpose input ouput port controlled transceiver where TrcvIdx is the transceiver index.
Failed condition	ERRN pin is active low.
Passed condition	ERRN pin is inactive high.

⌋

7.4.3.2 FRTRCV_E_FR_BUSERROR_TRCV_<TrcvIdx>

[SWS_FrTrcv_00488] Error Status of Class B serial peripheral interface controlled transceiver.

Upstream requirements: SRS_BSW_00458

⌈

Diagnostic Event (Error Name)	FRTRCV_E_FR_BUSERROR_TRCV_TrcvIdx
Description	Error Status of Class B serial peripheral interface controlled transceiver where TrcvIdx is the transceiver index.
Failed condition	Bus error, i.e. if any of bit 1...9 is set for transceivers according to class B of [1, Flex Ray EPL], see also SWS_FrTrcv_00458.

▽

△

Passed condition	No Bus error, i.e. all of bit 1...9 are cleared for transceivers according to class B of [1, FlexRay EPL], see also SWS_FrTrcv_00458.

⌋

[SWS_FrTrcv_00084]

Upstream requirements: SRS_BSW_00385

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

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]

Upstream requirements: SRS_BSW_00327

⌈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.⌋

Error detection

[SWS_FrTrcv_00237]

Upstream requirements: SRS_Fr_05151

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

[SWS_FrTrcv_00354] ⌈In case of faults of the transceiver hardware, the FlexRay Transceiver Driver shall raise a runtime error FRTRCV_E_FR_NO_CONTROL_TRCV.⌋

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_SetEventStatus(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_SetEventStatus(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, SRS FlexRay])

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.

Error notification

[SWS_FrTrcv_00391]

Upstream requirements: SRS_Fr_05168

⌈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.⌋

[SWS_FrTrcv_00384] ⌈If an error (e.g. the state of the ERRN pin is active low) is detected the module shall execute Dem_SetEventStatus(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_SetEventStatus(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.

7.5 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 [1, FlexRay EPL] 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.⌋

7.6 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]

Upstream requirements: SRS_Fr_05132

⌈Multiple FlexRay transceivers of the same type are handled by a single FlexRay transceiver driver.⌋

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

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

7.7 Wake Up Support

From the EcuM point of view, the FrTrcv only needs to detect and report passive wake-ups 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.

7.7.1 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.

7.7.2 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.

7.7.3 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.

7.8 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

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

[SWS_FrTrcv_00321] Definition of imported datatypes of module FrTrcv ⌈

Module	*Header File*	*Imported Type*
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Dio	Dio.h	Dio_ChannelGroupType
	Dio.h	Dio_ChannelType
	Dio.h	Dio_LevelType
	Dio.h	Dio_PortLevelType
	Dio.h	Dio_PortType
EcuM	EcuM.h	EcuM_WakeupSourceType
Icu	Icu.h	Icu_ChannelType
Spi	Spi.h	Spi_ChannelType
	Spi.h	Spi_DataBufferType
	Spi.h	Spi_NumberOfDataType
	Spi.h	Spi_SequenceType
	Spi.h	Spi_StatusType
Std	Std_Types.h	Std_ReturnType

⌋

8.2 Type definitions

[SWS_FrTrcv_00045] Separation of error and status values

Upstream requirements: SRS_BSW_00331

⌈All Basic Software Modules shall strictly separate error and status information. This requirement applies to return values and also to internal variables.⌋

[SWS_FrTrcv_00069] Do not return development error codes via API

Upstream requirements: SRS_BSW_00369

⌈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.⌋

[SWS_FrTrcv_00076] Module specific API return types

Upstream requirements: SRS_BSW_00377

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

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

⌋

8.2.1 FrTrcv_ConfigType

[SWS_FrTrcv_00487] Definition of datatype FrTrcv_ConfigType

Upstream requirements: SRS_BSW_00414

⌈

Name	FrTrcv_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	–
Description	Configuration data structure of the FrTrcv module.	
Available via	FrTrcv.h	

⌋

8.2.2 FrTrcv_TrcvModeType

[SWS_FrTrcv_00481] Definition of datatype FrTrcv_TrcvModeType ⌈

Name	FrTrcv_TrcvModeType		
Kind	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.		
Available via	Fr_GeneralTypes.h		

⌋

[SWS_FrTrcv_00048] Status values naming convention

Upstream requirements: SRS_BSW_00335

⌈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.⌋

[SWS_FrTrcv_00434] ⌈The type definition FrTrcv_TrcvModeType shall 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 [1, FlexRay EPL], at least these operation modes are defined:

- NORMAL
- STANDBY
- RECEIVEONLY (if supported by hardware)
- SLEEP (if supported by hardware)

Note: According to [1, FlexRay EPL] every FlexRay Transceiver has to support two mandatory states: FrTRCV_TRCVMODE_STANDBY and FrTRCV_TRCVMODE_NORMAL; all other states are optional.

8.2.3 FrTrcv_TrcvWUReasonType

[SWS_FrTrcv_00435] ⌈The type definition FrTrcv_TrcvWUReasonType shall 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] Definition of datatype FrTrcv_TrcvWUReasonType

Upstream requirements: SRS_BSW_00375

⌈

Name	FrTrcv_TrcvWUReasonType		
Kind	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_Set TransceiverMode() 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.		
Available via	Fr_GeneralTypes.h		

⌋

8.3 Function definitions

This chapter contains all APIs provided by the module.

[SWS_FrTrcv_00089] Parameter content shall be unique within the module

Upstream requirements: SRS_BSW_00390

⌈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).⌋

[SWS_FrTrcv_00092] Parameters shall have a range

Upstream requirements: SRS_BSW_00393

⌈Each parameter shall have a list of valid values or the minimum as well as maximum values shall be specified.⌋

[SWS_FrTrcv_00094] List the required parameters (per parameter)

Upstream requirements: SRS_BSW_00395

⌈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.⌋

8.3.1 FrTrcv_Init

[SWS_FrTrcv_00322] Definition of API function FrTrcv_Init ⌈

Service Name	FrTrcv_Init	
Syntax	`void FrTrcv_Init ( const FrTrcv_ConfigType* ConfigPtr )`	
Service ID [hex]	0x00	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	ConfigPtr	Pointer to the selected configuration set.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This service initializes the FrTrcv.	
Available via	FrTrcv.h	

⌋

[SWS_FrTrcv_00008] Initialization interface

Upstream requirements: SRS_BSW_00101

⌈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.⌋

[SWS_FrTrcv_00228] Initialization Sequence for FlexRay Transceiver Driver

Upstream requirements: SRS_Fr_05134

⌈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⌋

[SWS_FrTrcv_00230] Initialize the FlexRay Transceiver Driver

Upstream requirements: 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 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.

⌋

[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 runtime error FRTRCV_E_FR_NO_CONTROL_TRCV (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_00065]

Upstream requirements: SRS_BSW_00358

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

[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

Upstream requirements: SRS_Fr_05214

⌈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.

[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] Definition of API function FrTrcv_SetTransceiverMode ⌈

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.	
Available via	FrTrcv.h	

⌋

[SWS_FrTrcv_00252] Set FlexRay Transceiver Operation Mode

Upstream requirements: SRS_Fr_05166

⌈The FlexRay Transceiver Driver shall provide a software interface to set the operation mode of a specific FlexRay Transceiver device.⌋

[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_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

Upstream requirements: SRS_BSW_00357

⌈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.⌋

[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 runtime error FRTRCV_E_FR_NO_CONTROL_TRCV (see also [SWS_FrTrcv_00237]).⌋

[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.⌋

[SWS_FrTrcv_00276] ⌈If the mode transition fails [SWS_FrTrcv_00452], the function FrTrcv_SetTransceiverMode shall raise the following runtime error:

- FRTRCV_E_FR_TRCV_NOT_STANDBY:

  Transition to FRTRCV_TRCVMODE_STANDBY failed

- FRTRCV_E_FR_TRCV_NOT_NORMAL:

  Transition to FRTRCV_TRCVMODE_NORMAL failed

- FRTRCV_E_FR_TRCV_NOT_SLEEP:

  Transition to FRTRCV_TRCVMODE_SLEEP failed

- FRTRCV_E_FR_TRCV_NOT_RECEIVEONLY:

  Transition to FRTRCV_TRCVMODE_RECEIVEONLY failed

⌋

[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_UNINIT.⌋

[SWS_FrTrcv_00415] Hardware Checking Function on Bus Driver

Upstream requirements: SRS_Fr_05213

⌈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.⌋

[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] Definition of API function FrTrcv_GetTransceiverMode ⌈

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.	
Available via	FrTrcv.h	

⌋

[SWS_FrTrcv_00253]

Upstream requirements: SRS_Fr_05167

⌈The function FrTrcv_GetTransceiverMode shall return the state of the transceiver identified by FrTrcv_TrcvIdx.⌋

[SWS_FrTrcv_00281]

Upstream requirements: SRS_Fr_05151

⌈In case of a fault during transceiver access, the function FrTrcv_GetTransceiverMode shall raise the runtime error FRTRCV_E_FR_NO_CONTROL_TRCV (see also [SWS_FrTrcv_00237]).⌋

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.⌋

[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_UNINIT.⌋

[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_PARAM_POINTER shall be reported to DET if development error detection is enabled.⌋

8.3.4 FrTrcv_GetTransceiverWUReason

[SWS_FrTrcv_00325] Definition of API function FrTrcv_GetTransceiverWUReason ⌈

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.	
Available via	FrTrcv.h	

⌋

[SWS_FrTrcv_00232]

Upstream requirements: SRS_Fr_05144

⌈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.⌋

[SWS_FrTrcv_00284] ⌈In case of a fault during transceiver access, the function FrTrcv_GetTransceiverWUReason shall raise runtime error FRTRCV_E_FR_NO_CONTROL_TRCV (see also [SWS_FrTrcv_00237]).⌋

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.⌋

[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_UNINIT.⌋

[SWS_FrTrcv_00398] ⌈When the caller provides a NULL pointer as a parameter value to the API FrTrcv_GetTransceiverWUReason, the development error FRTRCV_E_PARAM_POINTER shall be reported to DET if development error detection is enabled.⌋

8.3.5 FrTrcv_ClearTransceiverWakeup

[SWS_FrTrcv_00329] Definition of API function FrTrcv_ClearTransceiverWakeup ⌈

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.	
Available via	FrTrcv.h	

⌋

[SWS_FrTrcv_00247]

Upstream requirements: SRS_Fr_05161

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

[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]

Upstream requirements: SRS_Fr_05151

⌈In case of a fault during transceiver access, the function FrTrcv_ClearTransceiverWakeup shall raise the runtime error FRTRCV_E_FR_NO_CONTROL_TRCV (see also [SWS_FrTrcv_00237]).⌋

[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.⌋

[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_UNINIT.⌋

8.3.6 FrTrcv_CheckWakeupByTransceiver

[SWS_FrTrcv_00331] Definition of callback function FrTrcv_CheckWakeupBy Transceiver ⌈

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	--	
Available via	FrTrcv.h	

⌋

[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

Upstream requirements: SRS_Fr_05147

⌈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.⌋

[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 only 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 FrTrcvWakeupByBusUsed [ECUC_FrTrcv_00350] 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_UNINIT.⌋

[SWS_FrTrcv_00229] Configuration "Notification for Wake-up by Bus"

Upstream requirements: 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.

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.⌋

[SWS_FrTrcv_00234] Support for Wake-up During Sleep Transition

Upstream requirements: SRS_Fr_05148

⌈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.⌋

8.3.7 FrTrcv_GetTransceiverError

[SWS_FrTrcv_00419] Definition of API function FrTrcv_GetTransceiverError ⌈

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 Fr Trcv_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.	
Available via	FrTrcv.h	

⌋

[SWS_FrTrcv_00412] Detect Error Information in Bus Driver

Upstream requirements: SRS_Fr_05212

⌈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.⌋

[SWS_FrTrcv_00420] FlexRay Transceiver Driver mandatory errors ⌈

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 [1, FlexRay EPL]	1
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between positive bus line and ground according to [1, FlexRay EPL]	2
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between positive bus line and power supply according to [1, FlexRay EPL]	3
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between negative bus line and power supply according to [1, FlexRay EPL]	4
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between negative bus line and ground according to [1, FlexRay EPL]	5
Mandatory	on the bus (i. e. external to the ECU):	Any bus fault according to [1, FlexRay EPL], which cannot be resolved according to the description of bit 1...5	6
Mandatory	Local error	Under voltage of transceiver power supply according to [1, FlexRay EPL]	7

▽

△

Mandatory	Local error	FlexRay transceiver is permanently enabled according to [1, FlexRay EPL]	8
Mandatory	Local error	Over temperature of transceiver according to [1, FlexRay EPL]	9

The FlexRay Transceiver Driver shall support all mandatory errors defined by the [1, FlexRay EPL], if supported by hardware.

⌋

[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_PARAM_POINTER 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 [1, FlexRay EPL]⌋

[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 [1, FlexRay EPL]⌋

[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_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 ([ECUC_FrTrcv_00357]), the function FrTrcv_GetTransceiverError shall raise development error FRTRCV_E_FR_INVALID_BRANCHIDX.⌋

8.3.8 FrTrcv_DisableTransceiverBranch

The FlexRay Transceiver Driver shall disable the faulty branches of active stars

[SWS_FrTrcv_00442] Definition of API function FrTrcv_DisableTransceiverBranch ⌈

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 Fr Trcv_BranchIdx is out of range. Branch state remains unchanged.
Description	This function disables the specified branch on the addressed (active star) transceiver.	
Available via	FrTrcv.h	

⌋

[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_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 ([ECUC_FrTrcv_00357]), the function FrTrcv_DisableTransceiverBranch shall raise development error FRTRCV_E_FR_INVALID_BRANCHIDX.⌋

8.3.9 FrTrcv_EnableTransceiverBranch

The FlexRay Transceiver Driver shall enable the branches of active stars synchronously to the FlexRay bus schedule

[SWS_FrTrcv_00443] Definition of API function FrTrcv_EnableTransceiverBranch
⌈

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 Fr Trcv_BranchIdx is out of range. Branch state remains unchanged.
Description	This function enables the specified branch on the addressed (active star) transceiver.	
Available via	FrTrcv.h	

⌋

[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_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 ([ECUC_FrTrcv_00357]), the function FrTrcv_EnableTransceiverBranch shall raise development error FRTRCV_E_FR_INVALID_BRANCHIDX.⌋

8.4 Scheduled functions

These functions are directly called by Basic Software Scheduler. The following functions shall have no return value and no parameter. All functions shall be non reentrant.

8.4.1 FrTrcv_MainFunction

[SWS_FrTrcv_00330] Definition of scheduled function FrTrcv_MainFunction ⌈

Service Name	FrTrcv_MainFunction
Syntax	`void FrTrcv_MainFunction ( void )`
Service ID [hex]	0x0d
Description	--
Available via	SchM_FrTrcv.h

⌋

[SWS_FrTrcv_00020]

Upstream requirements: SRS_BSW_00172

⌈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.⌋

[SWS_FrTrcv_00072]

Upstream requirements: SRS_BSW_00373

⌈Main processing function naming convention: The main function of the FlexRay transceiver driver shall be named FrTrcv_MainFunction.⌋

[SWS_FrTrcv_00126]

Upstream requirements: SRS_BSW_00428

⌈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.⌋

[SWS_FrTrcv_00340] ⌈The function FrTrcv_MainFunction shall scan all busses in STANDBY and SLEEP for wake up events and store them internally.⌋

[SWS_FrTrcv_00122]

Upstream requirements: SRS_BSW_00424

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

[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 no cycle time is configured for the FrTrcv_MainFunction (multiplicity of FrTrcvMainFunctionCycleTime is 0) 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_00123] Trigger conditions for schedulable objects

Upstream requirements: SRS_BSW_00425

⌈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

⌋

[SWS_FrTrcv_00436]

Upstream requirements: SRS_Fr_05203

⌈The FrTrcv_MainFunction shall call FrTrcv_GetTransceiverError () periodically to detect error information in BD if FrTrcvErrorCheckDuringCommunication is enabled.⌋

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 Expected interfaces

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

8.5.1 Mandatory interfaces

Note: This section defines all interfaces, which are required to fulfill the core functionality of the module.

[SWS_FrTrcv_00493] Definition of mandatory interfaces required by module Fr Trcv ⌈

API Function	*Header File*	*Description*
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

⌋

8.5.2 Optional interfaces

This section defines all interfaces, which are required to fulfill an optional functionality of the module.

The FlexRay Transceiver Driver uses these optional Interfaces:

[SWS_FrTrcv_00334] Definition of optional interfaces requested by module Fr Trcv ⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
Dio_ReadChannel	Dio.h	Returns the value of the specified DIO channel.
Dio_ReadChannelGroup	Dio.h	This Service reads a subset of the adjoining bits of a port.
Dio_ReadPort	Dio.h	Returns the level of all channels of that port.
Dio_WriteChannel	Dio.h	Service to set a level of a channel.
Dio_WriteChannelGroup	Dio.h	Service to set a subset of the adjoining bits of a port to a specified level.
Dio_WritePort	Dio.h	Service to set a value of the port.
EcuM_SetWakeupEvent	EcuM.h	Sets the wakeup event.
Icu_DisableNotification	Icu.h	This function disables the notification of a channel.
Icu_EnableNotification	Icu.h	This function enables the notification on the given channel.
Spi_GetStatus	Spi.h	Service returns the SPI Handler/Driver software module status.
Spi_ReadIB	Spi.h	Service for reading synchronously one or more data from an IB SPI Handler/Driver Channel specified by parameter.
Spi_SetupEB	Spi.h	Service to setup the buffers and the length of data for the EB SPI Handler/Driver Channel specified.
Spi_SyncTransmit	Spi.h	Service to transmit data on the SPI bus
Spi_WriteIB	Spi.h	Service for writing one or more data to an IB SPI Handler/Driver Channel specified by parameter.

⌋

Configuration of the container FrTrcvChannelDemEventParameterRefs [ECUC_FrTrcv_00450] enables production error report to the DEM module.

Configuration of the container FrTrcvDioAccess [ECUC_FrTrcv_00145] enables the FlexRay Transceiver Driver to use the API of the DIO module.

Configuration of the container FrTrcvSpiSequence [ECUC_FrTrcv_00144] 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

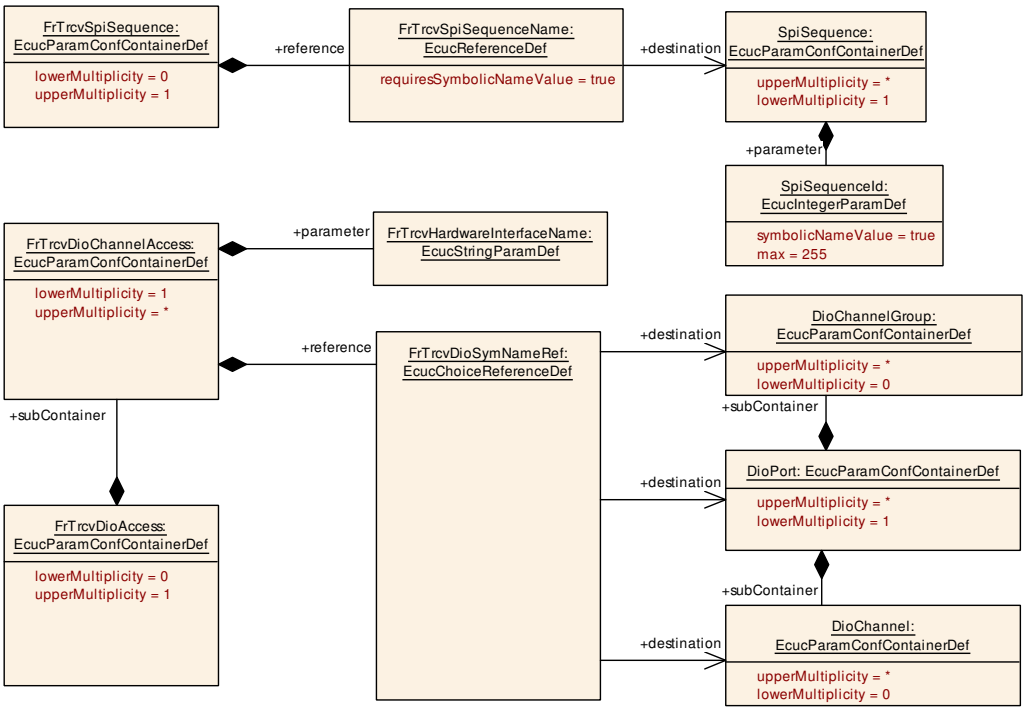

Figure 8.1: Dio/SPI configration of FlexRay Transceiver

[SWS_FrTrcv_00061]

Upstream requirements: SRS_BSW_00350

⌈If FRTRCV_DEV_ERROR_DETECT is configured, the FlexRay Transceiver Driver uses the API of the DET module.⌋

8.5.3 Configurable interfaces

In this section, all interfaces are listed where the target function could be configured. The target function is usually a callback function. The names of this kind of interfaces are not fixed because they are configurable.

[SWS_FrTrcv_91002] Definition of configurable interface <FrTrcvEnableInterrupt Callout> ⌈

Service Name	<FrTrcvEnableInterruptCallout>
Syntax	`void <FrTrcvEnableInterruptCallout> ( void )`
Service ID [hex]	0x15
Sync/Async	Asynchronous
Reentrancy	Non Reentrant
Parameters (in)	None
Parameters (inout)	None

▽

△

Parameters (out)	None
Return value	None
Description	–
Available via	FrTrcv_Externals.h

⌋

[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_SetEventStatus with the same signature.⌋

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

[SWS_FrTrcv_00019]

Upstream requirements: SRS_BSW_00171

⌈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]

⌋

8.6 Service Interfaces

No service interfaces provided.

9 Sequence diagrams

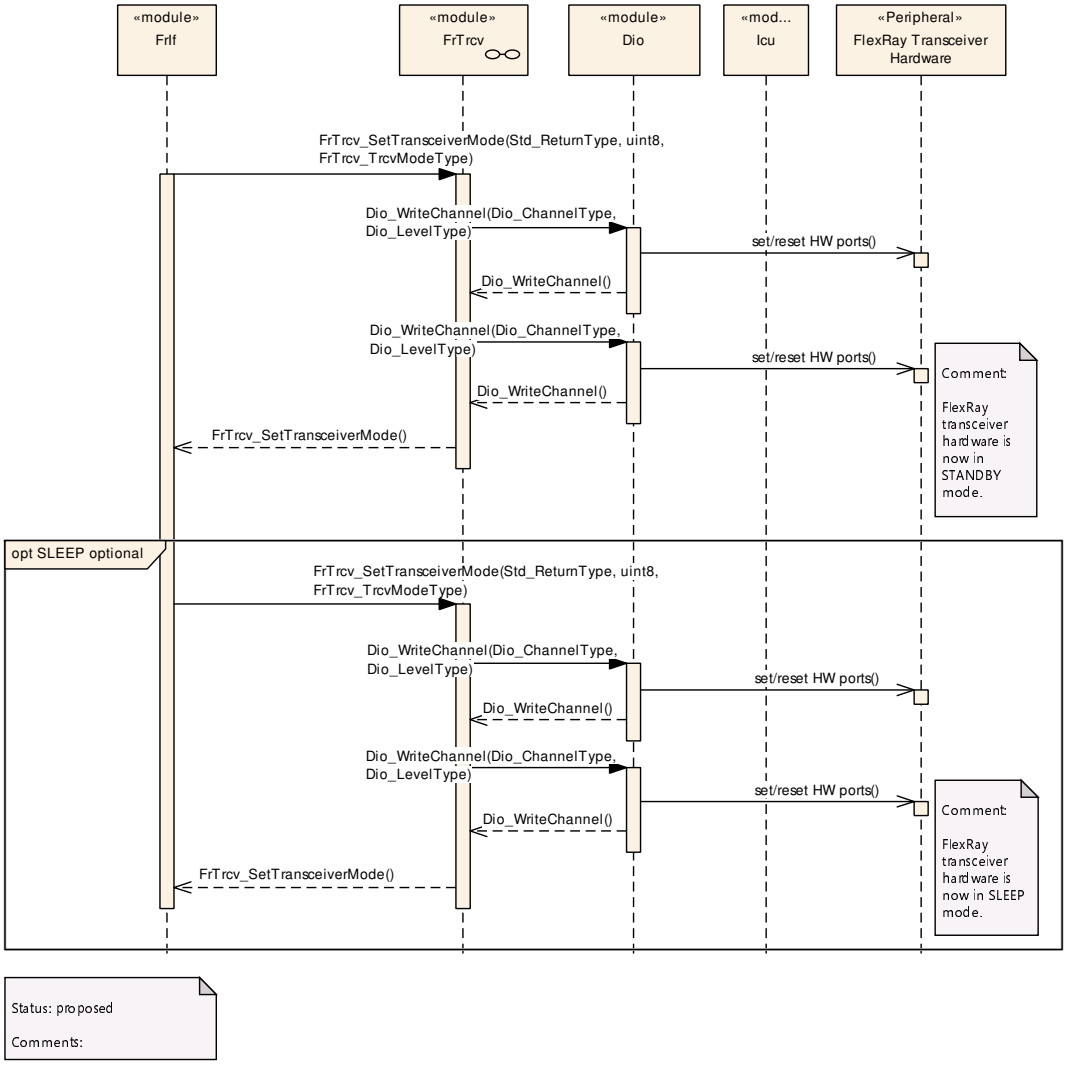

Figure 9.1: Transition to standby and sleep

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. 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 [4].

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module Flex Ray Transceiver Driver.

Chapter 10.3 specifies published information of the module Flex Ray Transceiver Driver.

10.1 How to read this chapter

For details refer to [5] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

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

10.2.1 General configuration requirements

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

[SWS_FrTrcv_00010]

Upstream requirements: SRS_BSW_00159

⌈A configuration tool is used to generate the configuration data and code if any.⌋

[SWS_FrTrcv_00018]

Upstream requirements: SRS_BSW_00170

⌈Data for reconfiguration of AUTOSAR SW-Components⌋

[SWS_FrTrcv_00047]

Upstream requirements: SRS_BSW_00334

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

[SWS_FrTrcv_00225]

Upstream requirements: SRS_Fr_05131

⌈Configuration Data for FlexRay Transceiver⌋

[SWS_FrTrcv_00016]

Upstream requirements: SRS_BSW_00167

⌈The configuration tool has to check the validity of the provided input data and the usability in the project context.⌋

[SWS_FrTrcv_00088] Containers shall have names

Upstream requirements: SRS_BSW_00389

⌈The configuration of the transceiver is assembled in a container⌋

[SWS_FrTrcv_00497] ⌈The Flexray Transceiver Driver module shall reject configurations with partition mappings which are not supported by the implementation.⌋

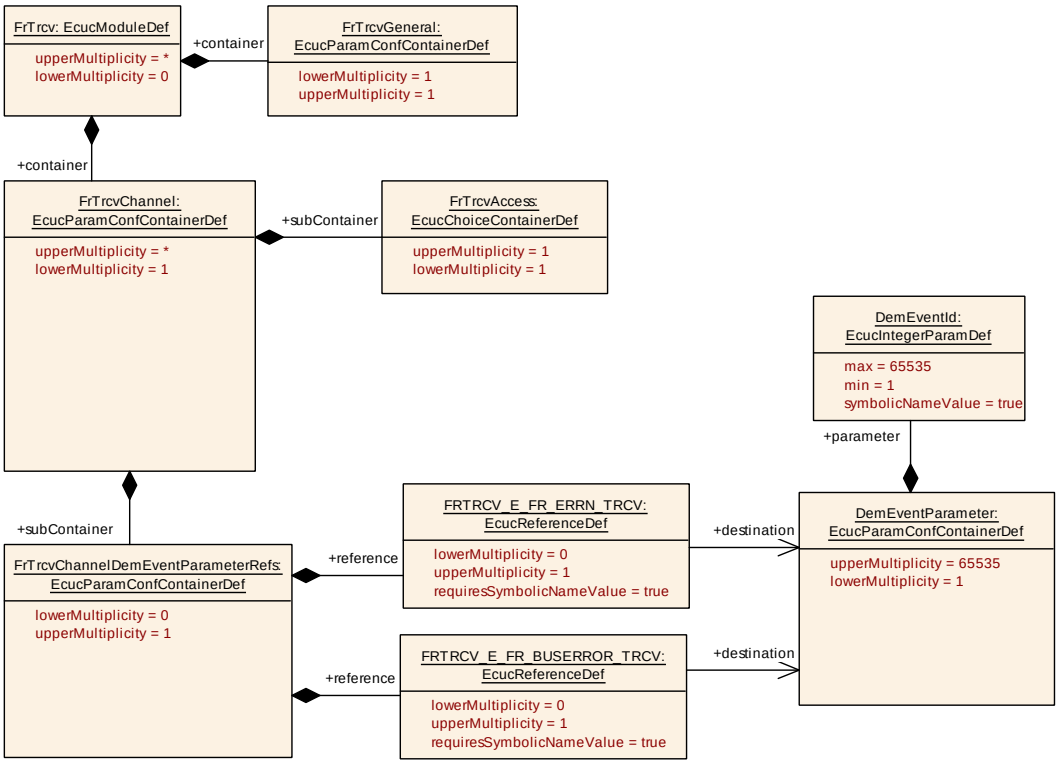

Figure 10.1: FlexRay Transceiver configuration structure

10.2.2 FrTrcv

[ECUC_FrTrcv_00459] Definition of EcucModuleDef FrTrcv ⌈

Module Name	FrTrcv
Description	Configuration of the FrTrcv (FlexRay Transceiver driver) module.
Post-Build Selectable Support	false
Post-Build Loadable Support	false

Included Containers		
Container Name	Multiplicity	Description
FrTrcvChannel	1..*	Container gives FlexRay transceiver driver information about a single FlexRay transceiver channel. Any FlexRay transceiver driver has such FlexRay transceiver channels.
FrTrcvGeneral	1	Container gives FlexRay transceiver driver basic information.

⌋

10.2.3 FrTrcvGeneral

[ECUC_FrTrcv_00055] Definition of EcucParamConfContainerDef FrTrcvGeneral
⌈

Container Name	FrTrcvGeneral
Parent Container	FrTrcv
Upstream mapping defined	–
Description	Container gives FlexRay transceiver driver basic information.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
FrTrcvDemReportErrorStatusConfiguration	0..1	[ECUC_FrTrcv_00455]
FrTrcvDevErrorDetect	1	[ECUC_FrTrcv_00341]
FrTrcvErrorCheckDuringCommunication	1	[ECUC_FrTrcv_00447]
FrTrcvErrorCheckInInit	1	[ECUC_FrTrcv_00446]
FrTrcvIndex	1	[ECUC_FrTrcv_00268]
FrTrcvMainFunctionCycleTime	0..1	[ECUC_FrTrcv_00343]
FrTrcvRetryCountInInit	1	[ECUC_FrTrcv_00445]
FrTrcvTimerType	0..1	[ECUC_FrTrcv_00457]
FrTrcvWaitTime	0..1	[ECUC_FrTrcv_00458]
FrTrcvWakeUpSupport	1	[ECUC_FrTrcv_00352]
FrTrcvEcucPartitionRef	0..*	[ECUC_FrTrcv_00472]

No Included Containers

⌋

[ECUC_FrTrcv_00455] Definition of EcucFunctionNameDef FrTrcvDemReportErrorStatusConfiguration ⌈

Parameter Name	FrTrcvDemReportErrorStatusConfiguration
Parent Container	FrTrcvGeneral
Upstream mapping defined	–
Description	Name of a C function which substitutes Dem_SetEventStatus.
Multiplicity	0..1
Type	EcucFunctionNameDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00341] Definition of EcucBooleanParamDef FrTrcvDevErrorDetect ⌈

Parameter Name	FrTrcvDevErrorDetect
Parent Container	FrTrcvGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00447] Definition of EcucBooleanParamDef FrTrcvErrorCheckDuringCommunication ⌈

Parameter Name	FrTrcvErrorCheckDuringCommunication
Parent Container	FrTrcvGeneral
Upstream mapping defined	–
Description	Enable a functionality to check transceiver's state during communication.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00446] Definition of EcucBooleanParamDef FrTrcvErrorCheckInInit ⌈

Parameter Name	FrTrcvErrorCheckInInit
Parent Container	FrTrcvGeneral
Upstream mapping defined	–
Description	Enable a functionality to check transceiver's state while initialization process of FrTrcv.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00268] Definition of EcucIntegerParamDef FrTrcvIndex ⌈

Parameter Name	FrTrcvIndex	
Parent Container	FrTrcvGeneral	
Upstream mapping defined	–	
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00343] Definition of EcucFloatParamDef FrTrcvMainFunctionCycleTime ⌈

Parameter Name	FrTrcvMainFunctionCycleTime	
Parent Container	FrTrcvGeneral	
Upstream mapping defined	–	
Description	Cyclic call time for function FrTrcvMainFunction in seconds. A multiplicity of 0 indicates no calls for this function. In this case function need not be present in compiled code.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	]0 .. INF[	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00445] Definition of EcucIntegerParamDef FrTrcvRetryCountInInit ⌈

Parameter Name	FrTrcvRetryCountInInit	
Parent Container	FrTrcvGeneral	
Upstream mapping defined	–	
Description	Specifies the number of retry count when error occurs while initialization process of Fr Trcv.	
Multiplicity	1	
Type	EcucIntegerParamDef	
Range	0 .. 255	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00457] Definition of EcucEnumerationParamDef FrTrcvTimerType ⌈

Parameter Name	FrTrcvTimerType	
Parent Container	FrTrcvGeneral	
Upstream mapping defined	–	
Description	Type of the Time Service Predefined Timer.	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	None	None
	Timer_1us16bit	16 bit 1us timer
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00458] Definition of EcucFloatParamDef FrTrcvWaitTime ⌈

Parameter Name	FrTrcvWaitTime	
Parent Container	FrTrcvGeneral	
Upstream mapping defined	–	
Description	Wait time for transceiver state changes in seconds.	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 2.55E-4]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00352] Definition of EcucEnumerationParamDef FrTrcvWakeUpSupport ⌈

Parameter Name	FrTrcvWakeUpSupport	
Parent Container	FrTrcvGeneral	
Upstream mapping defined	–	
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. Dependency: FrTrcvWakeupByBusUsed	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FRTRCV_WAKEUP_BY_POLLING	Wake up by polling
	FRTRCV_WAKEUP_NOT_SUPPORTED	Wake up is not supported
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00472] Definition of EcucReferenceDef FrTrcvEcucPartitionRef ⌈

Parameter Name	FrTrcvEcucPartitionRef
Parent Container	FrTrcvGeneral
Upstream mapping defined	–
Description	Maps the Flexray transceiver driver to zero or multiple ECUC partitions to make the modules API available in this partition.
Multiplicity	0..*
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

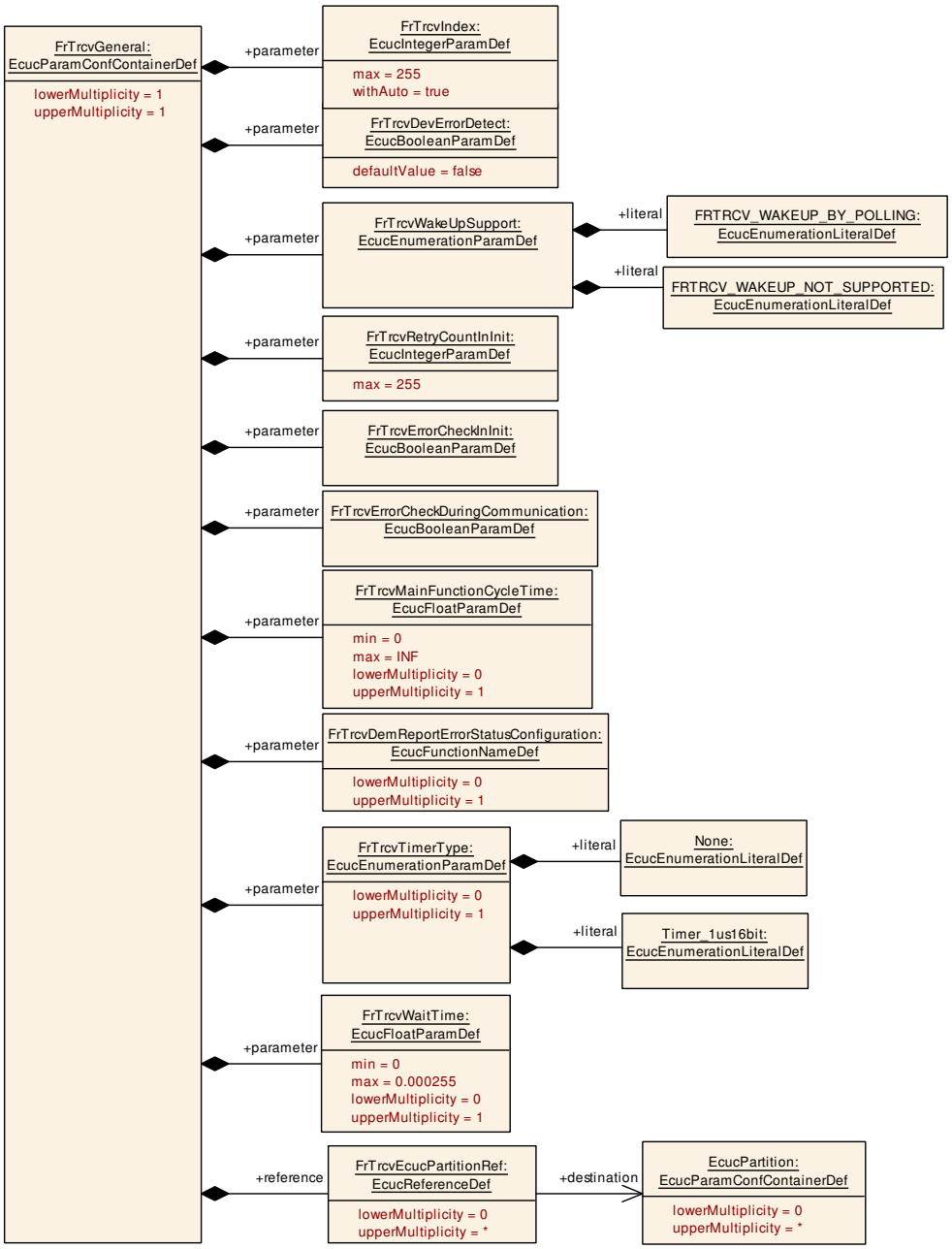

Figure 10.2: FlexRay Transceiver General configuration structure

10.2.4 FrTrcvChannel

[ECUC_FrTrcv_00091] Definition of EcucParamConfContainerDef FrTrcvChannel
⌈

Container Name	FrTrcvChannel
Parent Container	FrTrcv
Upstream mapping defined	–
Description	Container gives FlexRay transceiver driver information about a single FlexRay transceiver channel. Any FlexRay transceiver driver has such FlexRay transceiver channels.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTrcvChannelId	1	[ECUC_FrTrcv_00349]
FrTrcvChannelUsed	1	[ECUC_FrTrcv_00355]
FrTrcvControlsPowerSupply	1	[ECUC_FrTrcv_00346]
FrTrcvInitState	1	[ECUC_FrTrcv_00347]
FrTrcvMaxBaudrate	1	[ECUC_FrTrcv_00348]
FrTrcvWakeupByBusUsed	1	[ECUC_FrTrcv_00350]
FrTrcvChannelEcucPartitionRef	0..1	[ECUC_FrTrcv_00473]
FrTrcvIcuChannelRef	0..1	[ECUC_FrTrcv_00384]
FrTrcvWakeupSourceRef	0..1	[ECUC_FrTrcv_00269]

Included Containers		
Container Name	**Multiplicity**	**Description**
FrTrcvAccess	1	Choose if the FlexRay transceiver is accessed via an SPI Sequence or via DIO Pins.
FrTrcvBranchIdContainer	1..*	Only one SymbolicNameValue can be defined per container. Therefore this container is necessary.
FrTrcvChannelDemEvent ParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.

⌋

[ECUC_FrTrcv_00349] Definition of EcucIntegerParamDef FrTrcvChannelId ⌈

Parameter Name	FrTrcvChannelId	
Parent Container	FrTrcvChannel	
Upstream mapping defined	–	
Description	Unique identifier of the FlexRay Transceiver Channel.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 255	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00355] Definition of EcucBooleanParamDef FrTrcvChannelUsed ⌈

Parameter Name	FrTrcvChannelUsed
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Shall the related FlexRay transceiver channel be used?
Multiplicity	1
Type	EcucBooleanParamDef
Default value	true
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00346] Definition of EcucBooleanParamDef FrTrcvControlsPower Supply ⌈

Parameter Name	FrTrcvControlsPowerSupply
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Is ECU power supply controlled by this transceiver?
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00347] Definition of EcucEnumerationParamDef FrTrcvInitState ⌈

Parameter Name	FrTrcvInitState	
Parent Container	FrTrcvChannel	
Upstream mapping defined	–	
Description	State of FlexRay transceiver after power on. ImplementationType: FrTrcv_TrcvModeType	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FRTRCV_TRCVMODE_SLEEP	Sleep operation mode
	FRTRCV_TRCVMODE_STANDBY	Standby operation mode
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00348] Definition of EcucEnumerationParamDef FrTrcvMaxBaudrate ⌈

Parameter Name	FrTrcvMaxBaudrate
Parent Container	FrTrcvChannel
Upstream mapping defined	–

▽

△

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
Pre-Compile only	true	

⌋

[ECUC_FrTrcv_00350] Definition of EcucBooleanParamDef FrTrcvWakeupByBus Used ⌈

Parameter Name	FrTrcvWakeupByBusUsed
Parent Container	FrTrcvChannel
Upstream mapping defined	[7, CP_TPS_SystemTemplate]
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. Dependency: FRTRCV_WAKEUP_POLLING
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00473] Definition of EcucReferenceDef FrTrcvChannelEcucPartitionRef ⌈

Parameter Name	FrTrcvChannelEcucPartitionRef
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Maps one single Flexray transceiver channel to zero or one ECUC partitions. The ECUC partition referenced is a subset of the ECUC partitions where the Flexray transceiver driver is mapped to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00384] Definition of EcucReferenceDef FrTrcvIcuChannelRef ⌈

Parameter Name	FrTrcvIcuChannelRef
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Reference to the IcuChannel to enable/disable the interrupts for wakeups.

▽

△

Multiplicity	0..1
Type	Symbolic name reference to IcuChannel
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00269] Definition of EcucReferenceDef FrTrcvWakeupSourceRef

⌈

Parameter Name	FrTrcvWakeupSourceRef
Parent Container	FrTrcvChannel
Upstream mapping defined	–
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 Dependency: FrTrcvWakeUpSupport
Multiplicity	0..1
Type	Symbolic name reference to EcuMWakeupSource
Pre-Compile only	true

⌋

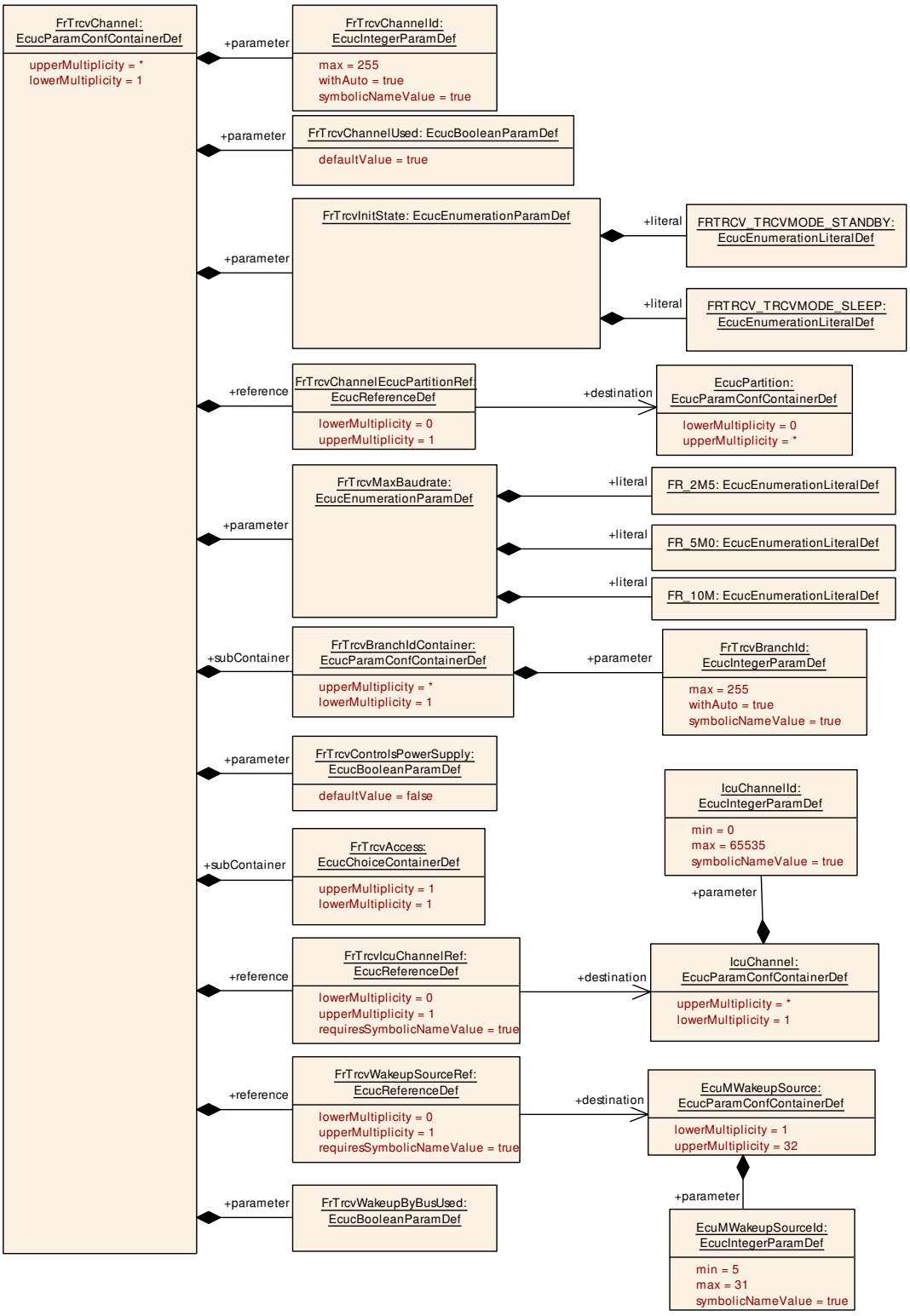

Figure 10.3: FlexRay Transceiver Channel configuration structure

10.2.5 FrTrcvChannelDemEventParameterRefs

[ECUC_FrTrcv_00450] Definition of EcucParamConfContainerDef FrTrcvChannelDemEventParameterRefs ⌈

Container Name	FrTrcvChannelDemEventParameterRefs
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FRTRCV_E_FR_BUSERROR_TRCV	0..1	[ECUC_FrTrcv_00453]
FRTRCV_E_FR_ERRN_TRCV	0..1	[ECUC_FrTrcv_00452]

No Included Containers

⌋

[ECUC_FrTrcv_00453] Definition of EcucReferenceDef FRTRCV_E_FR_BUSERROR_TRCV ⌈

Parameter Name	FRTRCV_E_FR_BUSERROR_TRCV
Parent Container	FrTrcvChannelDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to configured DEM event to report "Error Status of Class B (SPI) transceiver bus errors where TrcvIdx is the transceiver index" Dependency: Dem
Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00452] Definition of EcucReferenceDef FRTRCV_E_FR_ERRN_TRCV ⌈

Parameter Name	FRTRCV_E_FR_ERRN_TRCV
Parent Container	FrTrcvChannelDemEventParameterRefs
Upstream mapping defined	–
Description	Reference to configured DEM event to report "Error Status of Class A (GPIO) transceiver" Dependency: Dem

▽

△

Multiplicity	0..1
Type	Symbolic name reference to DemEventParameter
Pre-Compile only	true

⌋

10.2.6 FrTrcvBranchIdContainer

[ECUC_FrTrcv_00357] Definition of EcucParamConfContainerDef FrTrcvBranchIdContainer ⌈

Container Name	FrTrcvBranchIdContainer
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Only one SymbolicNameValue can be defined per container. Therefore this container is necessary.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTrcvBranchId	1	[ECUC_FrTrcv_00356]

No Included Containers

⌋

[ECUC_FrTrcv_00356] Definition of EcucIntegerParamDef FrTrcvBranchId ⌈

Parameter Name	FrTrcvBranchId	
Parent Container	FrTrcvBranchIdContainer	
Upstream mapping defined	–	
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	–	
With Auto	true	
Pre-Compile only	true	

⌋

10.2.7 FrTrcvAccess

[ECUC_FrTrcv_00454] Definition of EcucChoiceContainerDef FrTrcvAccess ⌈

Choice Container Name	FrTrcvAccess
Parent Container	FrTrcvChannel
Upstream mapping defined	–
Description	Choose if the FlexRay transceiver is accessed via an SPI Sequence or via DIO Pins.
Multiplicity	1
Pre-Compile only	true

No Included Parameters

Container Choices		
Container Name	**Multiplicity**	**Description**
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. Note: as ChannelIDs and JobIDs are hardlinked to the SequenceID through the SPI configuration, if the SPI configuration is modified (in particular the ChannelIDs associated to the SPI sequence FrTrcvSpiSequence refers to), FrTrcv must be compiled again to reflect the latest changes.

⌋

10.2.8 FrTrcvDioAccess

[ECUC_FrTrcv_00145] Definition of EcucParamConfContainerDef FrTrcvDioAccess ⌈

Container Name	FrTrcvDioAccess
Parent Container	FrTrcvAccess
Upstream mapping defined	–
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.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

No Included Parameters

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

⌋

10.2.9 FrTrcvDioChannelAccess

[ECUC_FrTrcv_00471] Definition of EcucParamConfContainerDef FrTrcvDioChannelAccess ⌈

Container Name	FrTrcvDioChannelAccess
Parent Container	FrTrcvDioAccess
Upstream mapping defined	–
Description	In this Container the relation between FR transceiver hardware pin names and Dio port access information is given.
Multiplicity	1..*
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTrcvHardwareInterfaceName	1	[ECUC_FrTrcv_00150]
FrTrcvDioSymNameRef	1	[ECUC_FrTrcv_00149]

No Included Containers

⌋

[ECUC_FrTrcv_00150] Definition of EcucStringParamDef FrTrcvHardwareInterfaceName ⌈

Parameter Name	FrTrcvHardwareInterfaceName
Parent Container	FrTrcvDioChannelAccess
Upstream mapping defined	–
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	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_FrTrcv_00149] Definition of EcucChoiceReferenceDef FrTrcvDioSymNameRef ⌈

Parameter Name	FrTrcvDioSymNameRef
Parent Container	FrTrcvDioChannelAccess
Upstream mapping defined	–
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]
Pre-Compile only	true

⌋

10.2.10 FrTrcvSpiSequence

[ECUC_FrTrcv_00144] Definition of EcucParamConfContainerDef FrTrcvSpiSequence ⌈

Container Name	FrTrcvSpiSequence
Parent Container	FrTrcvAccess
Upstream mapping defined	–
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 Flex Ray transceiver hardware has no SPI interface, there is no instance of this container. Note: as ChannelIDs and JobIDs are hardlinked to the SequenceID through the SPI configuration, if the SPI configuration is modified (in particular the ChannelIDs associated to the SPI sequence FrTrcvSpiSequence refers to), FrTrcv must be compiled again to reflect the latest changes.
Multiplicity	0..1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
FrTrcvSpiSequenceName	1	[ECUC_FrTrcv_00151]

No Included Containers

⌋

[ECUC_FrTrcv_00151] Definition of EcucReferenceDef FrTrcvSpiSequenceName
⌈

Parameter Name	FrTrcvSpiSequenceName
Parent Container	FrTrcvSpiSequence
Upstream mapping defined	–
Description	Reference to a Spi sequence configuration container. Dependency: SpiSequence
Multiplicity	1
Type	Symbolic name reference to SpiSequence
Pre-Compile only	true

⌋

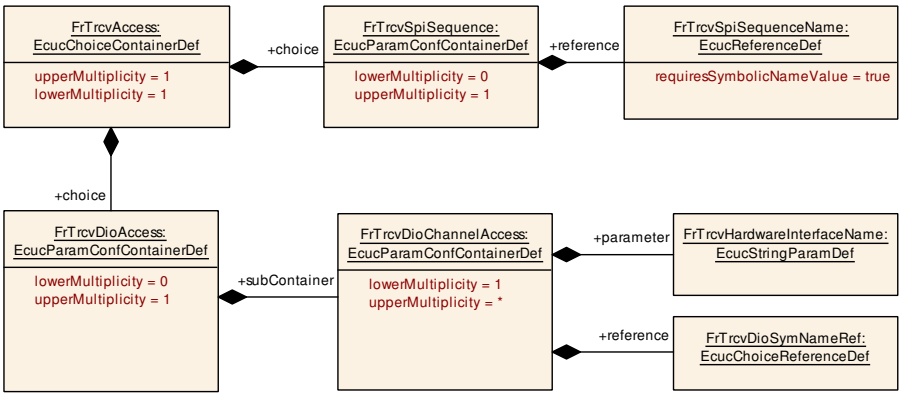

Figure 10.4: FlexRay Transceiver Dio Access configuration structure

10.3 Published Information

For details refer to [5] Chapter 10.3 *"Published Information"*.

A Not applicable requirements

[CP_SWS_FrTrcv_NA_00999]

Upstream requirements: SRS_BSW_00004, SRS_BSW_00168, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00386, SRS_BSW_00388, SRS_BSW_00392, SRS_BSW_00403, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00432, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

B.1 Traceable item history of this document according to AUTOSAR Release R26-11

B.1.1 Added Specification Items in R26-11

none

B.1.2 Changed Specification Items in R26-11

Number	Heading
[SWS_FrTrcv_00321]	Definition of imported datatypes of module FrTrcv

Table B.1: Changed Specification Items in R26-11

B.1.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_FrTrcv_00342]	Definition of EcucBooleanParamDef FrTrcvVersionInfoApi
[SWS_FrTrcv_00001]	Version identification
[SWS_FrTrcv_00285]	
[SWS_FrTrcv_00326]	Definition of API function FrTrcv_GetVersionInfo
[SWS_FrTrcv_00396]	

Table B.2: Deleted Specification Items in R26-11

B.2 Traceable item history of this document according to AUTOSAR Release R25-11

B.2.1 Added Specification Items in R25-11

none

B.2.2 Changed Specification Items in R25-11

Number	Heading
[SWS_FrTrcv_00488]	Error Status of Class B serial peripheral interface controlled transceiver.
[SWS_FrTrcv_00489]	Error Status of Class A (GPIO) transceiver where is the transceiver index.

Table B.3: Changed Specification Items in R25-11

B.2.3 Deleted Specification Items in R25-11

Number	Heading
[SWS_FrTrcv_00093]	Specify the scope of the parameters

Table B.4: Deleted Specification Items in R25-11

B.3 Traceable item history of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

Number	Heading
[SWS_FrTrcv_91002]	Definition of configurable interface <FrTrcvEnableInterruptCallout>

Table B.5: Added Specification Items in R24-11

B.3.2 Changed Specification Items in R24-11

Number	Heading
[ECUC_FrTrcv_00472]	Definition of EcucReferenceDef FrTrcvEcucPartitionRef
[SWS_FrTrcv_00330]	Definition of scheduled function FrTrcv_MainFunction
[SWS_FrTrcv_00334]	Definition of optional interfaces requested by module FrTrcv

Table B.6: Changed Specification Items in R24-11

B.3.3 Deleted Specification Items in R24-11

Number	Heading
[SWS_FrTrcv_00494]	

Table B.7: Deleted Specification Items in R24-11

B.4 Traceable item history of this document according to AUTOSAR Release R23-11

B.4.1 Added Specification Items in R23-11

Number	Heading
[SWS_FrTrcv_00001]	Version identification
[SWS_FrTrcv_00008]	Initialization interface
[SWS_FrTrcv_00010]	
[SWS_FrTrcv_00016]	
[SWS_FrTrcv_00018]	
[SWS_FrTrcv_00019]	
[SWS_FrTrcv_00020]	
[SWS_FrTrcv_00021]	
[SWS_FrTrcv_00022]	
[SWS_FrTrcv_00023]	
[SWS_FrTrcv_00033]	
[SWS_FrTrcv_00041]	
[SWS_FrTrcv_00045]	Separation of error and status values
[SWS_FrTrcv_00047]	
[SWS_FrTrcv_00048]	Status values naming convention
[SWS_FrTrcv_00057]	Pre-compile-time configuration
[SWS_FrTrcv_00059]	
[SWS_FrTrcv_00061]	
[SWS_FrTrcv_00064]	Standard API return type
[SWS_FrTrcv_00065]	
[SWS_FrTrcv_00069]	Do not return development error codes via API
[SWS_FrTrcv_00072]	
[SWS_FrTrcv_00074]	Definition of datatype FrTrcv_TrcvWUReasonType
[SWS_FrTrcv_00076]	Module specific API return types
[SWS_FrTrcv_00084]	
[SWS_FrTrcv_00085]	Definiton of development errors in module FrTrcv

▽

△

Number	Heading
[SWS_FrTrcv_00088]	Containers shall have names
[SWS_FrTrcv_00089]	Parameter content shall be unique within the module
[SWS_FrTrcv_00092]	Parameters shall have a range
[SWS_FrTrcv_00093]	Specify the scope of the parameters
[SWS_FrTrcv_00094]	List the required parameters (per parameter)
[SWS_FrTrcv_00117]	Separate C-Files for pre-compile time configuration parameters
[SWS_FrTrcv_00122]	
[SWS_FrTrcv_00123]	Trigger conditions for schedulable objects
[SWS_FrTrcv_00126]	
[SWS_FrTrcv_00225]	
[SWS_FrTrcv_00226]	
[SWS_FrTrcv_00227]	
[SWS_FrTrcv_00228]	Initialization Sequence for FlexRay Transceiver Driver
[SWS_FrTrcv_00229]	Configuration "Notification for Wake-up by Bus"
[SWS_FrTrcv_00230]	Initialize the FlexRay Transceiver Driver
[SWS_FrTrcv_00231]	
[SWS_FrTrcv_00232]	
[SWS_FrTrcv_00233]	Notification for Wake-up by Bus
[SWS_FrTrcv_00234]	Support for Wake-up During Sleep Transition
[SWS_FrTrcv_00236]	
[SWS_FrTrcv_00237]	
[SWS_FrTrcv_00238]	
[SWS_FrTrcv_00247]	
[SWS_FrTrcv_00252]	Set FlexRay Transceiver Operation Mode
[SWS_FrTrcv_00253]	
[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_00281]	
[SWS_FrTrcv_00282]	

▽

△

Number	Heading
[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_00306]	
[SWS_FrTrcv_00311]	
[SWS_FrTrcv_00312]	
[SWS_FrTrcv_00313]	
[SWS_FrTrcv_00321]	Definition of imported datatypes of module FrTrcv
[SWS_FrTrcv_00322]	Definition of API function FrTrcv_Init
[SWS_FrTrcv_00323]	Definition of API function FrTrcv_SetTransceiverMode
[SWS_FrTrcv_00324]	Definition of API function FrTrcv_GetTransceiverMode
[SWS_FrTrcv_00325]	Definition of API function FrTrcv_GetTransceiverWUReason
[SWS_FrTrcv_00326]	Definition of API function FrTrcv_GetVersionInfo
[SWS_FrTrcv_00329]	Definition of API function FrTrcv_ClearTransceiverWakeup
[SWS_FrTrcv_00330]	Definition of API function FrTrcv_MainFunction
[SWS_FrTrcv_00331]	Definition of callback function FrTrcv_CheckWakeupByTransceiver
[SWS_FrTrcv_00334]	Definition of optional interfaces in module FrTrcv
[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]	

▽

△

Number	Heading
[SWS_FrTrcv_00374]	
[SWS_FrTrcv_00375]	
[SWS_FrTrcv_00378]	
[SWS_FrTrcv_00379]	
[SWS_FrTrcv_00380]	
[SWS_FrTrcv_00384]	
[SWS_FrTrcv_00390]	
[SWS_FrTrcv_00391]	
[SWS_FrTrcv_00392]	
[SWS_FrTrcv_00395]	
[SWS_FrTrcv_00396]	
[SWS_FrTrcv_00397]	
[SWS_FrTrcv_00398]	
[SWS_FrTrcv_00412]	Detect Error Information in Bus Driver
[SWS_FrTrcv_00414]	Hardware Resetting Function on Bus Driver
[SWS_FrTrcv_00415]	Hardware Checking Function on Bus Driver
[SWS_FrTrcv_00419]	Definition of API function FrTrcv_GetTransceiverError
[SWS_FrTrcv_00420]	FlexRay Transceiver Driver mandatory errors
[SWS_FrTrcv_00421]	
[SWS_FrTrcv_00433]	
[SWS_FrTrcv_00434]	
[SWS_FrTrcv_00435]	
[SWS_FrTrcv_00436]	
[SWS_FrTrcv_00437]	
[SWS_FrTrcv_00438]	
[SWS_FrTrcv_00439]	
[SWS_FrTrcv_00440]	
[SWS_FrTrcv_00441]	
[SWS_FrTrcv_00442]	Definition of API function FrTrcv_DisableTransceiverBranch
[SWS_FrTrcv_00443]	Definition of API function FrTrcv_EnableTransceiverBranch
[SWS_FrTrcv_00444]	
[SWS_FrTrcv_00449]	
[SWS_FrTrcv_00451]	
[SWS_FrTrcv_00452]	
[SWS_FrTrcv_00453]	
[SWS_FrTrcv_00454]	
[SWS_FrTrcv_00455]	
[SWS_FrTrcv_00457]	

▽

△

Number	Heading
[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_00481]	Definition of datatype FrTrcv_TrcvModeType
[SWS_FrTrcv_00485]	
[SWS_FrTrcv_00487]	Definition of datatype FrTrcv_ConfigType
[SWS_FrTrcv_00488]	
[SWS_FrTrcv_00489]	
[SWS_FrTrcv_00490]	
[SWS_FrTrcv_00491]	
[SWS_FrTrcv_00492]	
[SWS_FrTrcv_00493]	Definition of mandatory interfaces in module FrTrcv
[SWS_FrTrcv_00494]	
[SWS_FrTrcv_00497]	
[SWS_FrTrcv_91001]	Definiton of runtime errors in module FrTrcv

Table B.8: Added Specification Items in R23-11

B.4.2 Changed Specification Items in R23-11

none

B.4.3 Deleted Specification Items in R23-11

none

B.5 Traceable item history of this document according to AUTOSAR Release R22-11

B.5.1 Added Specification Items in R22-11

Number	Heading
[SWS_FrTrcv_00489]	

Table B.9: Added Specification Items in R22-11

B.5.2 Changed Specification Items in R22-11

Number	Heading
[SWS_FrTrcv_00008]	Initialization interface
[SWS_FrTrcv_00019]	
[SWS_FrTrcv_00057]	Pre-compile-time configuration
[SWS_FrTrcv_00076]	Module specific API return types
[SWS_FrTrcv_00117]	Separate C-Files for pre-compile time configuration parameters
[SWS_FrTrcv_00270]	
[SWS_FrTrcv_00278]	
[SWS_FrTrcv_00291]	
[SWS_FrTrcv_00321]	Definition of imported datatypes of module FrTrcv
[SWS_FrTrcv_00323]	Definition of API function FrTrcv_SetTransceiverMode
[SWS_FrTrcv_00324]	Definition of API function FrTrcv_GetTransceiverMode
[SWS_FrTrcv_00325]	Definition of API function FrTrcv_GetTransceiverWUReason
[SWS_FrTrcv_00329]	Definition of API function FrTrcv_ClearTransceiverWakeup
[SWS_FrTrcv_00334]	Definition of optional interfaces in module FrTrcv
[SWS_FrTrcv_00360]	
[SWS_FrTrcv_00372]	
[SWS_FrTrcv_00373]	
[SWS_FrTrcv_00378]	
[SWS_FrTrcv_00379]	
[SWS_FrTrcv_00419]	Definition of API function FrTrcv_GetTransceiverError
[SWS_FrTrcv_00420]	FlexRay Transceiver Driver mandatory errors
[SWS_FrTrcv_00437]	
[SWS_FrTrcv_00442]	Definition of API function FrTrcv_DisableTransceiverBranch
[SWS_FrTrcv_00443]	Definition of API function FrTrcv_EnableTransceiverBranch
[SWS_FrTrcv_00457]	
[SWS_FrTrcv_00458]	
[SWS_FrTrcv_00461]	

▽

△

Number	Heading
[SWS_FrTrcv_00464]	
[SWS_FrTrcv_00467]	
[SWS_FrTrcv_00487]	Definition of datatype FrTrcv_ConfigType
[SWS_FrTrcv_00488]	
[SWS_FrTrcv_00493]	Definition of mandatory interfaces in module FrTrcv

Table B.10: Changed Specification Items in R22-11

B.5.3 Deleted Specification Items in R22-11

none

B.5.4 Added Constraints in R22-11

none

B.5.5 Changed Constraints in R22-11

none

B.5.6 Deleted Constraints in R22-11

none