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

Document Version	1.5.0
Document Status	Final
Part of Release	4.0
Revision	3

Document Change History

Date	Version	Changed by	Change Description
21.11.2011	1.5.0	AUTOSAR Administration	• Improved interrupt support by ICU • Improved production error concept
28.10.2010	1.4.0	AUTOSAR Administration	• Support of local wake up • Timing based on OS timer references • Support of error handling by Complex Device Drivers • Fixed constraints of configuration parameters • Removed APIs FrTrcv_EnableTransceiverWakeup and FrTrcv_DisableTransceiverWakeup
30.11.2009	1.3.0	AUTOSAR Administration	• Active star support added: a) Provided three new APIs: FrTrcv_GetTransceiverError(), FrTrcv_DisableTransceiverBranch(), FrTrcv_EnableTransceiverBranch()) b) Active stars can now be accessed as a single FlexRay transceiver by the FlexRay state manager via the FlexRay interface. c) Configuration support of of branches of active stars by FrTrcvBranchIdContainer d) Active stars can now be accessed as a single FlexRay transceiver by the FlexRay state manager via the FlexRay interface: • API FrTrcv_Cbk_WakeupByTransceiver has been renamed to FrTrcv_CheckWakeupByTransceiver • Legal disclaimer revised
30.01.2008	1.2.1	AUTOSAR Administration	Chapter 9 regenerated from BSW UML Model

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
23.06.2008	1.2.2	AUTOSAR Administration	Legal disclaimer revised
17.12.2007	1.2.0	AUTOSAR Administration	• Converted FrTrcv999 into SWS items • Wakeup consolidation • Resolve improvement ambiguities • Tables generated in Chapter 8 and 10 • Document meta information extended • Small layout adaptations made
31.01.2007	1.1.0	AUTOSAR Administration	• Added header file includes: MemMap_<ModuleId>.h and SchM_<ModuleId>.h • Renamed error codes • Support of wake up interrupt sharing (callback only if wake up occurred) • FrTrcv API is only called via FrIf FlexRay Interface, which is transparent to the transceiver driver • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added
18.05.2006	1.0.0	AUTOSAR Administration	Initial release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

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

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

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

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

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

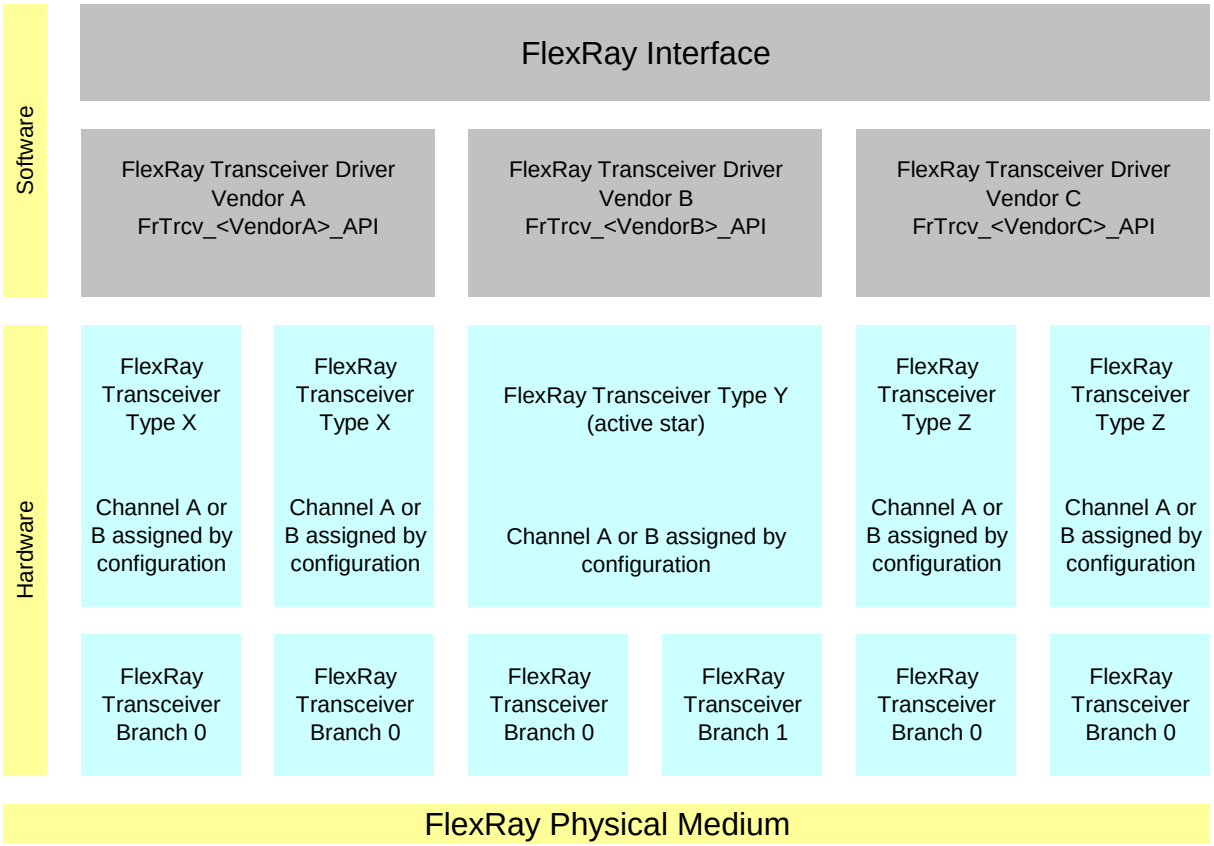


Figure 1: Description of the basic structure of the FlexRay Stack

One FlexRay Interface accesses several FlexRay Transceivers (FlexRay Transceiver Type X .. Z) using one or several FlexRay Transceiver Driver(s) (FrTrcv Driver Vendor A…C) from different vendors.
A zero based index (FrTrcv_TrcvIdx) identifies the transceiver within the context of the transceiver driver.
E.g., FlexRay transceiver A of FlexRay transceiver type Z is addressed by the index 0, FlexRay transceiver B by the index 1 in the example in the Figure above.
A zero based index (FrTrcv_BranchIdx) identifies the branch within the context of the transceiver. .

1.1 Goal of FlexRay transceiver driver

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

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

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

1.2 Explicitly uncovered FlexRay Transceiver Functionality

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

1.3 Active Stars

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

2 Acronyms and abbreviations

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

MCG	Module Configuration Generator
MISRA	Motor Industry Software Reliability Association
n/a	Not applicable
OS	Operating System
Port	Port, one of the SPAL SW modules
RAM	Random Access Memory
RxD	Receive Data
RxEN	Receive Enable
SBC	System Basis Chip; A device, which integrates e.g. CAN and/or FlexRay and/or LIN transceiver, watchdog and power control.
SchM	Schedule Manager
SPAL	Standard Peripheral Abstraction Layer
SPI/Spi	Serial Peripheral Interface.
SPI/Spi Channel	A channel is a software exchange medium for data that are defined with the same criteria: configuration parameters, number of data elements with same size and data pointers (source & destination) or location. See specification of SPI driver for more details.
SPI/Spi Job	A job is composed of one or several channels with the same chip select. A job is considered to be atomic and therefore cannot be interrupted. A job has also an assigned priority. See specification of SPI driver for more details.
SPI/Spi Sequence	A sequence is a number of consecutive jobs to be transmitted. A sequence depends on a static configuration. See specification of SPI driver for more details.
SRS	Software Requirement Specification
SW	Software
SW-C	Software-Component
SWS	Software Specification
XML	eXtended Markup Language

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

General Requirements on Basic Software
AUTOSAR_SRS_BSWGeneral.pdf

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

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

3.2 Related standards and norms

Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

Specification of Communication Manager
AUTOSAR_SWS_COMManager.pdf

Specification of DIO Driver
AUTOSAR_SWS_DIODriver.pdf

Specification of SPI Handler/Driver
AUTOSAR_SWS_SPIHandlerDriver.pdf

Requirements on FlexRay
AUTOSAR_SRS_FlexRay.pdf

Specification of Communication Stack Types
AUTOSAR_SWS_CommunicationStackTypes.pdf

Specification of Basic Software Scheduler
AUTOSAR_SWS_BSW_Scheduler.pdf

Specification of Memory Mapping
AUTOSAR_SWS_MemoryMapping.pdf

Basic Software Module Description Template
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

4 Constraints and assumptions

4.1 Limitations

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

[FrTrcv231] ⌈The FlexRay Transceiver Driver shall use the APIs of underlying DIO drivers synchronously.⌋(BSW05138)

[FrTrcv433] ⌈The FlexRay Transceiver Driver should use the APIs of underlying SPI drivers synchronously if possible and asynchronously where required.⌋( )

[FrTrcv441] ⌈The FlexRay transceiver requires a LEVEL 2, Enhanced (Synchronous/Asynchronous) SPI Handler/Driver⌋( )

[FrTrcv238] ⌈The FlexRay Transceiver Driver shall handle the transceiver-specific timing requirements internally.⌋(BSW05152)

The communication between the µC and the transceiver is performed via ports or SPI or both. If ports are used, applying values in a predefined sequence and with a given timing to the ports are used to communicate and change the hardware operation modes. These sequences and timings must be handled within the FlexRay Transceiver Driver.

4.2 Applicability to car domains

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

5 Dependencies to other modules

Module	*Dependencies*
FrIf	The FlexRay Interface controls the state of the FlexRay transceivers via the FlexRay Transceiver Driver
Det	The FlexRay Transceiver Driver informs the Development Error Tracer on development errors
Dem	Dem gets production error information from FlexRay Transceiver Driver.
Dio	Dio module is used to access FlexRay transceiver hardware connected via ports.
EcuM	EcuM gets wake up event information from FlexRay Transceiver Driver if supported by hardware.
RTE	The FlexRay Transceiver Driver main function may be scheduled by the by the RTE.
Spi	Spi module is used to access FlexRay transceiver hardware connected via SPI.

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

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

Here is an example:

Upper layer:

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

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

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

FlexRay Transceiver Driver FrTrcv D1 (has Transceiver HW knowledge):
NORMAL for 1st device is achieved by setting Dio signal S1 to HIGH and DIO Signal S2 to HIGH
"DIO set S1 and S2 to HIGH"

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

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

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

5.1 File structure

5.1.1 Naming convention for transceiver driver implementation

[FrTrcv059] ⌈A FlexRay Transceiver Driver implementation may support different FlexRay Transceiver hardware. ⌋(BSW00347)

[FrTrcv021] ⌈The BSW00347 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 BSW00347:
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⌋(BSW00300)

5.1.2 Code file structure

The FrTrcv module consists of the following code files:

[FrTrcv033] ⌈FrTrcv.c is the implementation general C file. It does not contain interrupt routines. ⌋(BSW00314)

[FrTrcv058] ⌈Basic set of module files: FrTrcv_Cfg.c is the pre compile time configuration code file. It is generated by the configuration tool.⌋(BSW00346)

[FrTrcv057] ⌈Pre-compile-time configuration
All modules of the AUTOSAR Basic Software, operating on Pre--compile--time configuration data (not to be modified after compile time), shall group and export the configuration data to configuration files.
Module specific configuration header file naming convention:

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

Static configuration is decoupled from implementation. Separation of configuration dependent data at compile time furthermore enhances flexibility, readability and reduces version management as no source code is affected.⌋(BSW00345)

[FrTrcv079] ⌈Separate C-Files for configuration parameters
Configuration parameters being stored in memory shall be placed into separate c--files (effected parameters are those from link--time configuration as well as those from post--build time configuration).
Rationale: Enable the use of different object files.⌋(BSW00380)

[FrTrcv117] ⌈Separate C-Files for pre-compile time configuration parameters
Configuration parameters being stored in memory shall be placed into separate c--files (effected parameters are those from link--time configuration as well as those from post--build time configuration).
Enable the use of different object files.⌋(BSW00419)

5.1.3 Header file structure

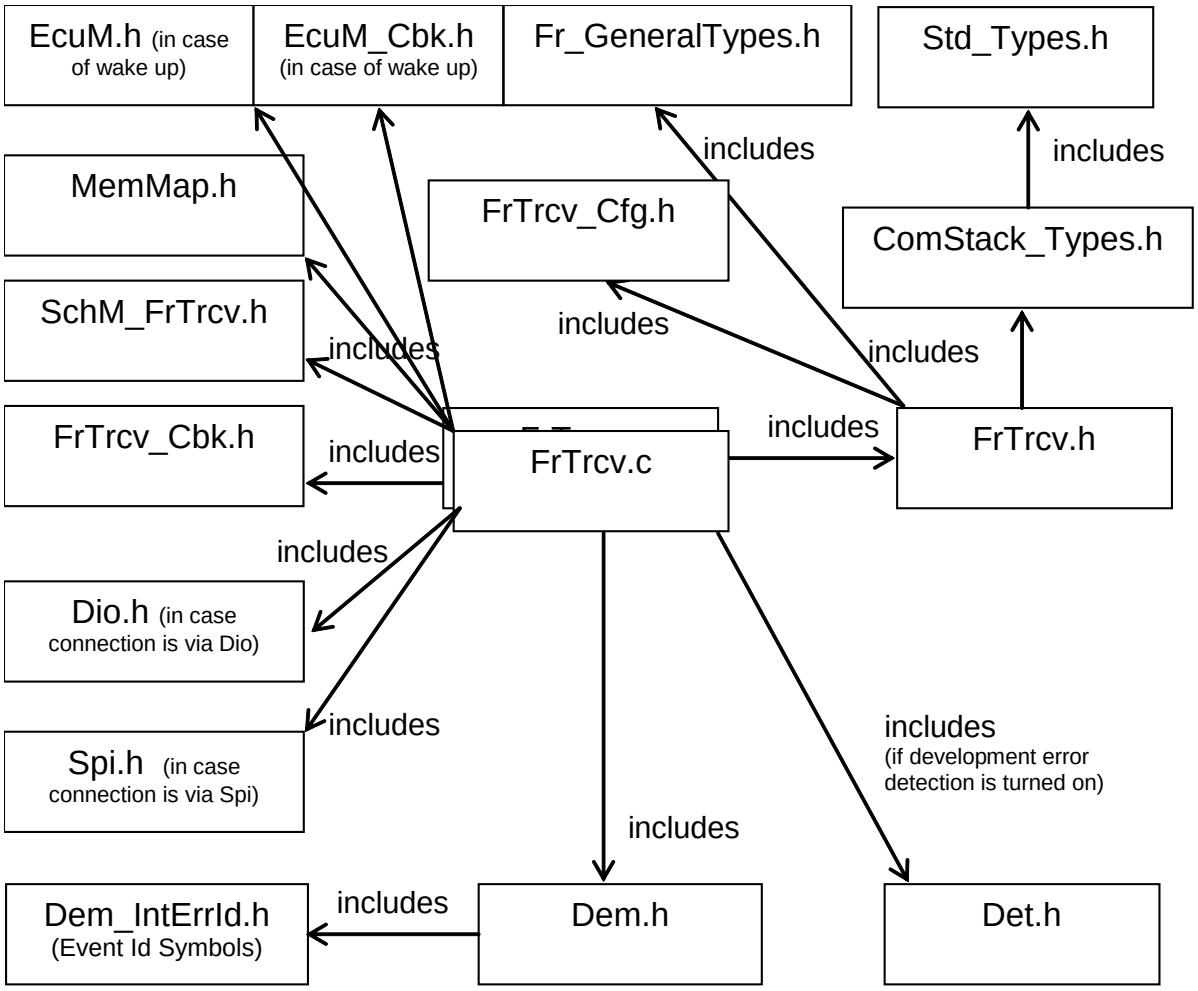


Figure 2 FlexRay Transceiver Driver Header File Structure

[FrTrcv022] ⌈All AUTOSAR Basic Software Modules shall only import the necessary information (i.e. header files) that is required to fulfill the modules functional requirements. ⌋(BSW00301)
The header file structure shall include the following FlexRay-specific header files:

[FrTrcv113] ⌈FrTrcv.h -- General header file of the FlexRay Transceiver Driver. It contains only information relevant for other BSW modules (API). Differences in API depending on the configuration are encapsulated. ⌋(BSW00415)

[FrTrcv023] ⌈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. ⌋(BSW00302)

[FrTrcv110] ⌈FrTrcv_Cfg.h -- Pre compile time configuration parameter file. It is generated by the configuration tool. ⌋(BSW00412)

[FrTrcv429] ⌈FrTrcv.h shall include FrTrcv_Cfg.h⌋( )

[FrTrcv430] ⌈FrTrcv.h shall include ComStack_Types.h⌋ ( )

[FrTrcv431] ⌈FrTrcv.h shall include Fr_GeneralTypes.h
Note: Fr_GeneralTypes.h -- Contains definitions and declarations shared by all AUTOSAR FlexRay BSW modules. ⌋ ( )

[FrTrcv266] ⌈SchM_FrTrcv.h -- contains Basic Software Scheduler declarations used by the FlexRay Transceiver Driver is included as specified by [21]
Hint: The Basic Software Scheduler offers concepts and services to integrate Basic Software Modules Hence, the Basic Software Scheduler

- embed Basic Software Module implementations into the AUTOSAR OS context
- trigger main processing functions of the Basic Software Modules
- apply data consistency mechanisms for the Basic Software Modules to communicate modes between Basic Software Modules⌋(BSW00435)

[FrTrcv267] ⌈MemMap.h -- the memory mapping abstraction mechanisms information is included as specified by [22]. ⌋(BSW00436)

[FrTrcv335] ⌈Dem.h -- The module FrTrcv shall include the Dem.h file (see FrTrcv107).
By this inclusion the APIs to report errors as well as the required Event Id symbols are included.
Note: This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration tool.
The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h. ⌋ ( )

[FrTrcv060] ⌈Std_Types.h -- includes platform specifc header files and compiler specifc header files. It defines standard data types and values for standard defines.
This file is indirectly included via ComStack_Types.h. ⌋(BSW00348)

[FrTrcv068] ⌈Compiler.h – the compiler specific header file is Compiler.h. All mappings of not standardized keywords of compiler specific scope shall be placed and organized in this compiler specific type and keyword header.
This file is indirectly included via ComStack_Types.h. ⌋(BSW00361)

[FrTrcv062] ⌈Platform_Types.h – the platform specific header file. All integer type definitions of target and compiler specific scope shall be placed and organized in this single type header.
This file is indirectly included via ComStack_Types.h. ⌋(BSW00353)

[FrTrcv424] ⌈Det.h -- FrTrcv.c shall include Det.h only if development error detection is turned on.⌋ ( )

[FrTrcv408] ⌈EcuM_WakeupSourceType shall be imported via EcuM.h in case wake up is configured and supported by hardware.⌋ ( )

[FrTrcv409] ⌈EcuM_Cbk.h – the transceiver driver indicates the wake up source and mode to the ECU State Manager if supported by hardware⌋ ( )

[FrTrcv476] ⌈FrTrcv_Cbk.h - the transceiver driver encapsulates declarations of the callouts/callbacks configured in FrTrcv456_Conf in this header file.⌋ ( )

6 Requirements traceability

Requirement	Satisfied by
-	FrTrcv312
-	FrTrcv283
-	FrTrcv472
-	FrTrcv461
-	FrTrcv354
-	FrTrcv372
-	FrTrcv395
-	FrTrcv444
-	FrTrcv442
-	FrTrcv439
-	FrTrcv390
-	FrTrcv085
-	FrTrcv402
-	FrTrcv454
-	FrTrcv433
-	FrTrcv373
-	FrTrcv452
-	FrTrcv405
-	FrTrcv437
-	FrTrcv304
-	FrTrcv380
-	FrTrcv463
-	FrTrcv406
-	FrTrcv465
-	FrTrcv458
-	FrTrcv340
-	FrTrcv362
-	FrTrcv424
-	FrTrcv321
-	FrTrcv455
-	FrTrcv367
-	FrTrcv467
-	FrTrcv460
-	FrTrcv398
-	FrTrcv420
-	FrTrcv330
-	FrTrcv441

-	FrTrcv384
-	FrTrcv285
-	FrTrcv360
-	FrTrcv368
-	FrTrcv449
-	FrTrcv440
-	FrTrcv276
-	FrTrcv278
-	FrTrcv335
-	FrTrcv419
-	FrTrcv421
-	FrTrcv311
-	FrTrcv371
-	FrTrcv277
-	FrTrcv392
-	FrTrcv430
-	FrTrcv334
-	FrTrcv429
-	FrTrcv331
-	FrTrcv326
-	FrTrcv404
-	FrTrcv332
-	FrTrcv366
-	FrTrcv352
-	FrTrcv464
-	FrTrcv270
-	FrTrcv450
-	FrTrcv313
-	FrTrcv393
-	FrTrcv459
-	FrTrcv359
-	FrTrcv378
-	FrTrcv329
-	FrTrcv284
-	FrTrcv397
-	FrTrcv305
-	FrTrcv396
-	FrTrcv339
-	FrTrcv435
-	FrTrcv272
-	FrTrcv325

-	FrTrcv364
-	FrTrcv443
-	FrTrcv462
-	FrTrcv275
-	FrTrcv295
-	FrTrcv274
-	FrTrcv438
-	FrTrcv375
-	FrTrcv322
-	FrTrcv323
-	FrTrcv434
-	FrTrcv431
-	FrTrcv291
-	FrTrcv466
-	FrTrcv296
-	FrTrcv401
-	FrTrcv358
-	FrTrcv279
-	FrTrcv408
-	FrTrcv363
-	FrTrcv409
-	FrTrcv308
-	FrTrcv403
-	FrTrcv407
-	FrTrcv361
-	FrTrcv236
-	FrTrcv476
-	FrTrcv457
-	FrTrcv324
-	FrTrcv282
-	FrTrcv379
-	FrTrcv451
-	FrTrcv280
-	FrTrcv338
-	FrTrcv273
-	FrTrcv374
-	FrTrcv262
-	FrTrcv474
-	FrTrcv341_Conf
-	FrTrcv475
-	FrTrcv453

BSW00003	FrTrcv001
BSW00004	FrTrcv002
BSW00005	FrTrcv478
BSW00006	FrTrcv478
BSW00007	FrTrcv478
BSW00009	FrTrcv478
BSW00010	FrTrcv478
BSW00101	FrTrcv008
BSW00161	FrTrcv478
BSW00164	FrTrcv478
BSW00168	FrTrcv478
BSW00171	FrTrcv019
BSW00172	FrTrcv020
BSW00300	FrTrcv021
BSW00301	FrTrcv022
BSW00302	FrTrcv023
BSW00304	FrTrcv478
BSW00306	FrTrcv478
BSW00307	FrTrcv478
BSW00308	FrTrcv478
BSW00309	FrTrcv478
BSW00310	FrTrcv030
BSW00312	FrTrcv478
BSW00314	FrTrcv033
BSW00318	FrTrcv034
BSW00321	FrTrcv478
BSW00323	FrTrcv037
BSW00325	FrTrcv478
BSW00326	FrTrcv478
BSW00327	FrTrcv041
BSW00328	FrTrcv478
BSW00329	FrTrcv043
BSW00330	FrTrcv478
BSW00331	FrTrcv045
BSW00333	FrTrcv478
BSW00335	FrTrcv048
BSW00336	FrTrcv478
BSW00337	FrTrcv050
BSW00338	FrTrcv051
BSW00339	FrTrcv052
BSW00341	FrTrcv478

BSW00342	FrTrcv478
BSW00344	FrTrcv478
BSW00345	FrTrcv057
BSW00346	FrTrcv058
BSW00347	FrTrcv059
BSW00348	FrTrcv060
BSW00350	FrTrcv061
BSW00353	FrTrcv062
BSW00355	FrTrcv478
BSW00357	FrTrcv064
BSW00358	FrTrcv065
BSW00359	FrTrcv478
BSW00360	FrTrcv478
BSW00361	FrTrcv068
BSW00369	FrTrcv069
BSW00370	FrTrcv478
BSW00371	FrTrcv071
BSW00373	FrTrcv072
BSW00374	FrTrcv073
BSW00375	FrTrcv074
BSW00376	FrTrcv075
BSW00377	FrTrcv076
BSW00378	FrTrcv478
BSW00379	FrTrcv078
BSW00380	FrTrcv079
BSW00382	FrTrcv478
BSW00383	FrTrcv478
BSW00384	FrTrcv478
BSW00385	FrTrcv084
BSW00386	FrTrcv478
BSW00387	FrTrcv086
BSW00390	FrTrcv089
BSW00391	FrTrcv090
BSW00392	FrTrcv478
BSW00393	FrTrcv092
BSW00394	FrTrcv093
BSW00395	FrTrcv094
BSW00398	FrTrcv478
BSW00399	FrTrcv478
BSW00400	FrTrcv478
BSW00401	FrTrcv478

BSW00404	FrTrcv478
BSW00405	FrTrcv478
BSW00406	FrTrcv104
BSW00407	FrTrcv105
BSW00409	FrTrcv107
BSW00410	FrTrcv478
BSW00411	FrTrcv109
BSW00412	FrTrcv110
BSW00413	FrTrcv478
BSW00414	FrTrcv112
BSW00415	FrTrcv113
BSW00416	FrTrcv478
BSW00417	FrTrcv478
BSW00419	FrTrcv117
BSW00420	FrTrcv478
BSW00421	FrTrcv119
BSW00422	FrTrcv478
BSW00423	FrTrcv478
BSW00424	FrTrcv122
BSW00425	FrTrcv123
BSW00426	FrTrcv478
BSW00427	FrTrcv478
BSW00428	FrTrcv126
BSW00429	FrTrcv478
BSW00431	FrTrcv478
BSW00432	FrTrcv478
BSW00433	FrTrcv478
BSW00434	FrTrcv478
BSW00435	FrTrcv266
BSW00436	FrTrcv267
BSW05000	FrTrcv478
BSW05001	FrTrcv478
BSW05002	FrTrcv478
BSW05003	FrTrcv478
BSW05004	FrTrcv478
BSW05005	FrTrcv478
BSW05006	FrTrcv478
BSW05007	FrTrcv478
BSW05009	FrTrcv478
BSW05010	FrTrcv478
BSW05011	FrTrcv478

BSW05012	FrTrcv478
BSW05013	FrTrcv478
BSW05015	FrTrcv478
BSW05016	FrTrcv478
BSW05018	FrTrcv478
BSW05019	FrTrcv478
BSW05022	FrTrcv478
BSW05023	FrTrcv478
BSW05024	FrTrcv478
BSW05025	FrTrcv478
BSW05027	FrTrcv478
BSW05031	FrTrcv478
BSW05033	FrTrcv478
BSW05034	FrTrcv478
BSW05035	FrTrcv478
BSW05038	FrTrcv478
BSW05039	FrTrcv478
BSW05040	FrTrcv478
BSW05041	FrTrcv478
BSW05042	FrTrcv478
BSW05044	FrTrcv478
BSW05045	FrTrcv478
BSW05046	FrTrcv478
BSW05047	FrTrcv478
BSW05048	FrTrcv478
BSW05049	FrTrcv478
BSW05050	FrTrcv478
BSW05051	FrTrcv478
BSW05052	FrTrcv478
BSW05053	FrTrcv478
BSW05055	FrTrcv478
BSW05056	FrTrcv478
BSW05058	FrTrcv478
BSW05059	FrTrcv478
BSW05060	FrTrcv478
BSW05063	FrTrcv478
BSW05064	FrTrcv478
BSW05065	FrTrcv478
BSW05066	FrTrcv478
BSW05067	FrTrcv478
BSW05068	FrTrcv478

BSW05069	FrTrcv478
BSW05072	FrTrcv478
BSW05073	FrTrcv478
BSW05074	FrTrcv478
BSW05075	FrTrcv478
BSW05076	FrTrcv478
BSW05077	FrTrcv478
BSW05078	FrTrcv478
BSW05079	FrTrcv478
BSW05082	FrTrcv478
BSW05083	FrTrcv478
BSW05084	FrTrcv478
BSW05085	FrTrcv478
BSW05088	FrTrcv478
BSW05089	FrTrcv478
BSW05090	FrTrcv478
BSW05093	FrTrcv478
BSW05095	FrTrcv478
BSW05096	FrTrcv478
BSW05097	FrTrcv478
BSW05101	FrTrcv478
BSW05102	FrTrcv478
BSW05104	FrTrcv478
BSW05106	FrTrcv478
BSW05107	FrTrcv478
BSW05109	FrTrcv478
BSW05111	FrTrcv478
BSW05113	FrTrcv478
BSW05114	FrTrcv478
BSW05115	FrTrcv478
BSW05116	FrTrcv478
BSW05117	FrTrcv478
BSW05120	FrTrcv478
BSW05121	FrTrcv478
BSW05123	FrTrcv478
BSW05124	FrTrcv478
BSW05125	FrTrcv478
BSW05126	FrTrcv478
BSW05129	FrTrcv478
BSW05130	FrTrcv478
BSW05132	FrTrcv226

BSW05133	FrTrcv227
BSW05134	FrTrcv228
BSW05136	FrTrcv229
BSW05137	FrTrcv230
BSW05138	FrTrcv231
BSW05144	FrTrcv232
BSW05147	FrTrcv233
BSW05148	FrTrcv234
BSW05151	FrTrcv237, FrTrcv281, FrTrcv306
BSW05152	FrTrcv238
BSW05153	FrTrcv478
BSW05154	FrTrcv478
BSW05155	FrTrcv478
BSW05156	FrTrcv478
BSW05157	FrTrcv478
BSW05158	FrTrcv478
BSW05161	FrTrcv247
BSW05162	FrTrcv478
BSW05163	FrTrcv478
BSW05164	FrTrcv478
BSW05165	FrTrcv478
BSW05166	FrTrcv252
BSW05167	FrTrcv253
BSW05168	FrTrcv391
BSW05169	FrTrcv478
BSW05170	FrTrcv478
BSW05171	FrTrcv478
BSW05172	FrTrcv478
BSW05173	FrTrcv478
BSW05174	FrTrcv478
BSW05175	FrTrcv478
BSW05200	FrTrcv478
BSW05201	FrTrcv478
BSW05202	FrTrcv478
BSW05203	FrTrcv436
BSW05204	FrTrcv478
BSW05205	FrTrcv478
BSW05206	FrTrcv478
BSW05207	FrTrcv478
BSW05208	FrTrcv478
BSW05209	FrTrcv478

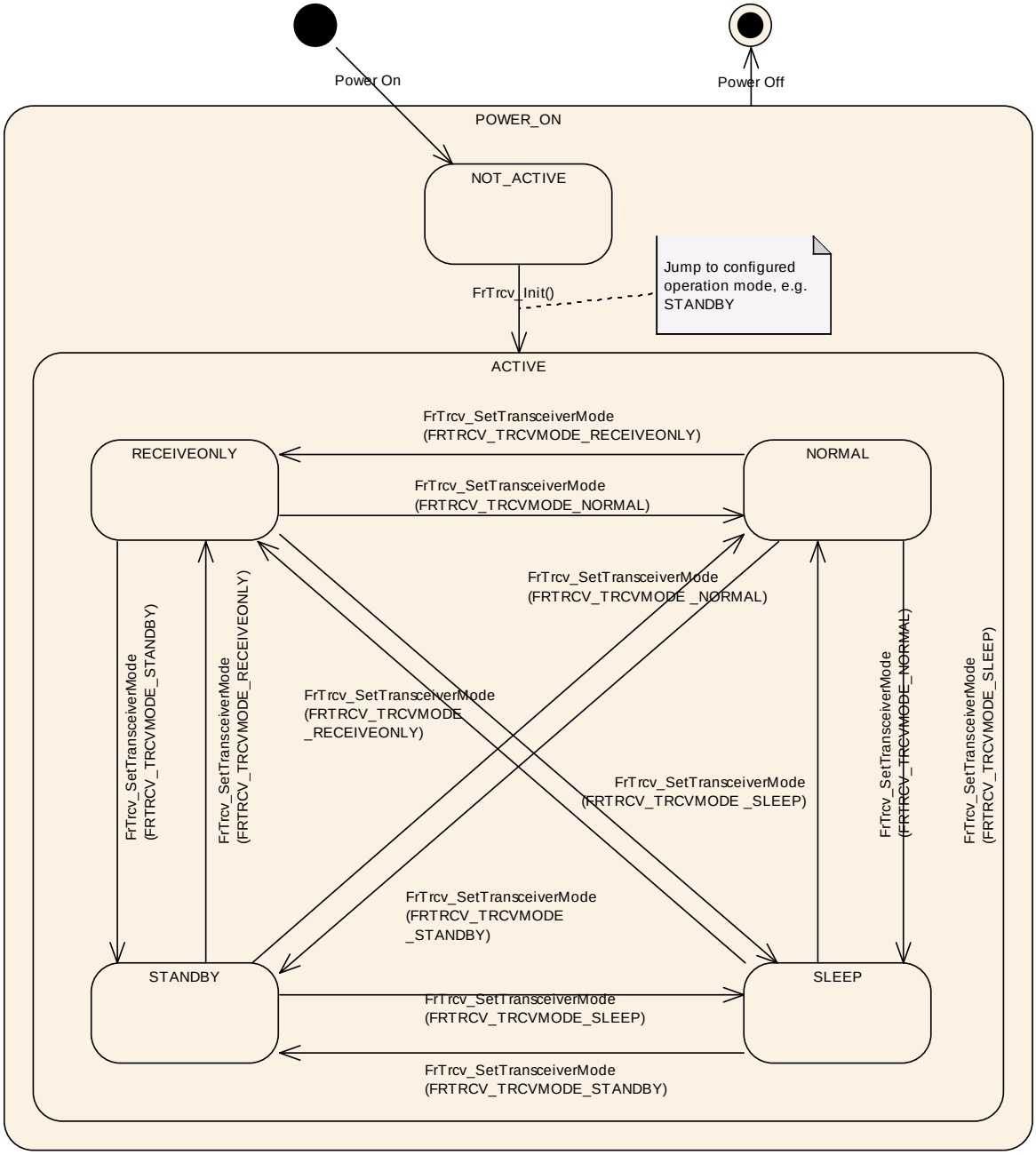
BSW05210	FrTrcv478
BSW05211	FrTrcv478
BSW05212	FrTrcv412
BSW05213	FrTrcv415
BSW05214	FrTrcv414
BSW05215	FrTrcv478

7 Functional specification

7.1 AUTOSAR FlexRay Transceiver Operation Mode Model

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

[FrTrcv227] ⌈The FlexRay Transceiver Driver shall support the independent configuration of the bus operation mode for each supported Cluster. Due to the different startup requirements on a multiple-FlexRay-Cluster-ECU, the FlexRay Transceiver Driver support the independent pre-selection of the bus operation mode to which each FlexRay Transceiver device is set during the driver initialization. ⌋ (BSW05133)



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.

[FrTrcv291] ⌈On initialization, the FrTrcv module shall switch all covered FlexRay transceivers into the state ACTIVE. This is observable, see FrTrcv277
In state ACTIVE each FlexRay transceiver may be in a different sub state.
Only the states NORMAL and STANDBY are mandatory for FlexRay transceivers; all other states are optional.
If a state is optional according to [5] and NOT supported by the transceiver and ECU hardware (e.g. SLEEP or RECEIVEONLY), the transceiver driver substitutes an equivalent state (i.e. STANDBY instead of SLEEP; and NORMAL instead of RECEIVEONLY) and returns the state actually supported by the transceiver hardware by the FrTrcv_GetTransceiverMode() function.⌋( )

7.2 FlexRay transceiver hardware operation modes

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

7.2.1 Temporary “Go-To-Sleep” Mode

The mode often referred to as "Go-to-sleep” is a temporary mode when switching from NORMAL to (optional) SLEEP. The FlexRay transceiver driver encapsulates such a temporary mode within one of the FlexRay transceiver driver software states. In addition, the FlexRay transceiver driver switches first from NORMAL to STANDBY and then with an additional (optional) API call from STANDBY to (optional) SLEEP.
The transition from NORMAL to STANDBY is not affected and will be performed directly.

[FrTrcv352] ⌈The FlexRay transceiver driver encapsulates transient or temporary modes within one of the static optional or mandatory FlexRay transceiver driver software states. ⌋( )

7.2.2 "Active Star" Mode

[FrTrcv451] ⌈If a transceiver supports active star mode, do NOT assume it is in node mode. ⌋( )

7.3 Error classification

[FrTrcv107] ⌈Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h. ⌋(BSW00409)

[FrTrcv050] ⌈Development error values are of type uint8. ⌋(BSW00337)

[FrTrcv084] ⌈Production code errors and development errors of FlexRay Transceiver Driver are provided in the table below. This list must be mapped into the code (i.e. the respective function calls to the error notifications must be in the code). ⌋(BSW00385)

[FrTrcv085] ⌈Development and Production Errors used by the FlexRay Transceiver Driver:

Type or error	***Relevance***	***Related error code***	***Value [hex]***
API service called with wrong parameter	Development	FRTRCV_E_FR_INVALID_TRCVIDX FlexRay transceiver index out of range	0x01
API service called with wrong parameter	Development	FRTRCV_E_FR_INVALID_BRANCHIDX FlexRay transceiver branch index out of range	0x02
API Service used without initialization	Development	FRTRCV_E_FR_UNINIT	0x10
API service called in wrong transceiver operation mode	Development	FRTRCV_E_FR_TRCV_NOT_STANDBY FRTRCV_E_FR_TRCV_NOT_NORMAL FRTRCV_E_FR_TRCV_NOT_SLEEP FRTRCV_E_FR_TRCV_NOT_RECEIVEONLY	0x11 0x12 0x13 0x14
API service called passing a NULL pointer as a parameter	Development	FRTRCV_E_FR_TRCV_NULL_PTR	0x15
Error Status of Class A (GPIO) transceiver	Production	FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>	* (Assigned by

			DEM)
Error Status of Class B (SPI) transceiver bus errors where TrcvIdx is the transceiver index	Production	FRTRCV_E_FR_BUSERROR_TRCV<TrcvIdx>	* (Assigned by DEM)
No/incorrect communication to transceiver	Development	FRTRCV_E_FR_NO_CONTROL_TRCV	0x16

⌋( )

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

[FrTrcv041] ⌈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. ⌋(BSW00327)

7.4 Error detection

[FrTrcv061] ⌈The detection of development errors is configurable (*ON* / *OFF*) at pre-compile time. The switch *FRTRCV_DEV_ERROR_DETECT* shall activate or deactivate the detection of all development errors. ⌋(BSW00350)

[FrTrcv037] ⌈If the *FRTRCV_DEV_ERROR_DETECT* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.3. ⌋(BSW00323)

[FrTrcv119] ⌈The detection of production code errors cannot be switched off. ⌋(BSW00421)

[FrTrcv237] ⌈The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness if supported by hardware. ⌋(BSW05151)

[FrTrcv354] ⌈In case of faults of the transceiver hardware, the FlexRay Transceiver Driver shall raise a development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled. ⌋( )

Example: Depending on the supported transceiver device, the driver could check the correctness of the executed control communication and the operation mode of the transceiver in order to detect defective or faulty transceiver hardware and/or corrupted SPI communication.
This check only applies to errors within the transceiver or the transceiver control communication (ports or SPI), i.e. errors caused by malfunction of the µC, SW or a defect transceiver device.

[FrTrcv295] ⌈The FrTrcv module shall check for bus errors and report them to DEM executing Dem_ReportErrorStatus(FRTRCV_E_FR_BUSERROR_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREFAILED).⌋( )

[FrTrcv472] ⌈If a no bus error is detected, the module shall execute Dem_ReportErrorStatus(FRTRCV_E_FR_BUSERROR_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREPASSED).⌋( )

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

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

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

7.5 Error notification

[FrTrcv051] ⌈Detected development errors will be reported to the error hook of the Development Error Tracer (DET) if the pre-processor switch FRTRCV_DEV_ERROR_DETECT is set (see chapter 10).⌋(BSW00338)

[FrTrcv052] ⌈Production errors shall be reported to Diagnostic Event Manager.⌋(BSW00339)

[FrTrcv391] ⌈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.⌋(BSW05168)

[FrTrcv384] ⌈If an error (e.g. the state of the ERRN pin is active low) is detected the module shall execute Dem_ReportErrorStatus(FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREFAILED).
In the above description, <TrcvIdx> represents the transceiver index.⌋( )

[FrTrcv395] ⌈If an error is not detected (e.g. the state of the ERRN pin is passive high) the module shall execute Dem_ReportErrorStatus(FRTRCV_E_FR_ERRN_TRCV<TrcvIdx>, DEM_EVENT_STATUS_PREPASSED).
In the above descriptions, <TrcvIdx> represents the transceiver index.⌋( )

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

7.6 Preconditions for driver initialization

[FrTrcv296] ⌈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.

[FrTrcv228] ⌈The schedule of the initialization of the FlexRay Transceiver Driver shall be configurable.⌋(BSW05134)

[FrTrcv358] ⌈The FlexRay bus transceiver driver shall fulfill the FlexRay Transceiver hardware timing requirements also on initialization.⌋( )

[FrTrcv359] ⌈The FlexRay transceiver driver initialization shall schedule before other BSW modules (e.g. the FlexRay State manager) access its software services.⌋( )

[FrTrcv360] ⌈The runtime of the underlying services used shall be short enough and synchronous in order to fulfill the requirements defined by the FlexRay EPL [5] and the timing requirements of the hardware device used. (FrTrcv231).⌋( )

[FrTrcv361] ⌈The FlexRay Transceiver Driver runtime shall support setup and hold times of the FlexRay Transceiver Hardware devices in all states including low power states, e.g. sleep. ⌋ ( )

7.7 Instance concept

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

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

[FrTrcv226] ⌈Multiple FlexRay transceivers of the same type are handled by a single FlexRay transceiver driver. ⌋(BSW05132)

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

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

7.8 Debug Support

[FrTrcv401] ⌈Each variable that shall be accessible by AUTOSAR Debugging, shall be defined as global variable. ⌋ ( )

[FrTrcv402] ⌈All type definitions of variables which shall be debugged, shall be accessible by the header file FrTrcv.h. ⌋ ( )

[FrTrcv403] ⌈The declaration of variables in the header file shall be such, that it is possible to calculate the size of the variables by C-"sizeof".⌋ ( )

[FrTrcv404] ⌈Variables available for debugging shall be described in the respective Basic Software Module Description⌋ ( )

[FrTrcv405] ⌈The mode (FrTrcv_TrcvModeType) of the FlexRay Transceiver shall be available for debugging. ⌋ ( )

[FrTrcv406] ⌈The wake up reason (FrTrcv_TrcvWUReasonType) of the FlexRay Transceiver shall be available for debugging if supported by hardware.⌋ ( )

[FrTrcv407] ⌈The wake up state of the FlexRay Transceiver shall be available for debugging if supported by hardware.⌋ ( )

[FrTrcv450] ⌈The branch state of the FlexRay Transceiver active stars shall be available for debugging if supported by hardware.⌋ ( )

7.9 Wake Up Support

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

7.9.1 Power-on:

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

7.9.2 Active wakeup:

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

7.9.3 Passive wakeup:

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

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

7.9.4 Starting FlexRay Communication without Losing Wake up Events

While the CC is in state HALT until it is in state READY, the FlexRay transceiver shall remain in STANDBY in order to detect wake up events if supported by hardware.
Caveat: Wake up events may get lost during power supply failure (this is detected and reported to DEM).

The FlexRay transceiver driver will store the wake up information (including a time stamp) internally if supported by hardware.
Wake up events will be cleared with the API FrTrcv_ClearTransceiverWakeup (intended use: internal only)
As soon as the CC is in state READY, NORMAL is requested. Until the requested state has been achieved (polling SPI or DIO), the CC shall not leave READY.

7.9.5 Stopping FlexRay Communication without Losing Wake Up Events

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

7.10 Version checking

[FrTrcv002] ⌈The FlexRay Transceiver Driver module shall perform Inter Module Checks to avoid integration of incompatible files.
The imported included files shall be checked by preprocessing directives.
The following version numbers shall be verified:
- <MODULENAME>_AR_RELEASE_MAJOR_VERSION
- <MODULENAME>_AR_RELEASE_MINOR_VERSION
Where <MODULENAME> is the module short name of the other (external) modules which provide header files included by the FlexRay Transceiver Driver module.
If the values are not identical to the expected values, an error shall be reported. ⌋(BSW00004)

8 API specification

8.1 Imported types

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

[FrTrcv321] ⌈

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

⌋ ( )

8.2 Type definitions

[FrTrcv030] ⌈API naming convention
All AUTOSAR Basic Software Modules shall implement an API based on the following naming rules: - Composition of API: <Module name>_ServiceName() - Module name: 2..8 letters, derived from WP Architecture SW Module List - Only one underscore between module name and service name - Spelling of API: First letter of each word upper case, consecutive letters lower case. ⌋ (BSW00310)

[FrTrcv045] ⌈Separation of error and status values
All Basic Software Modules shall strictly separate error and status information. This requirement applies to return values and also to internal variables. ⌋ (BSW00331)

[FrTrcv069] ⌈Do not return development error codes via API
All AUTOSAR Basic Software Modules shall not return specific development error codes via the API. In case of a detected development error the error shall only be reported to the DET. If the API-- function which detected the error has a return type it shall return E_NOT_OK. ⌋ (BSW00369)

[FrTrcv071] ⌈Do not pass function pointers via API

The passing of function pointers as API parameter is forbidden for all AUTOSAR Basic Software Modules.」(BSW00371)

[FrTrcv076]「Module specific API return types
If a Basic Software Module needs module specific return types, it shall use one of the following possibilities:
1. Use uint8 as return value, take the standard E_OK value from Std_Types.h and define additional return values using #define.
2. Define a module specific return value with typedef enum.
Hint: Within this enum, E_OK cannot be used (because E_OK is already #defined in Std_Types.h and OSEK OS)」(BSW00377)

8.2.1 FrTrcv_TrcvModeType

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

[FrTrcv048]「Status values naming convention:
The following naming rules apply for status values that are visible outside of the module: - Status values shall have only CAPITAL LETTERS - Naming convention: FRTRCV_<STATUSNAME>
If <STATUSNAME> consists of several words, they shall be separated by underscores.」(BSW00335)

[FrTrcv434] 「The type definition FrTrcv_TrcvModeType shall be kept in a file named Fr_GeneralTypes.h and be protected by a FR_GENERAL_TYPES define in order:
- to be shared between different FlexRay Transceiver Drivers
- to be included into the FrIf
If different FlexRay Transceiver Drivers are used, only one instance of this file has to be included in the source tree」( )

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

8.2.2 FrTrcv_TrcvWUReasonType

[FrTrcv435] ⌈The type definition FrTrcv_TrcvWUReasonType shall be kept in a file named Fr_GeneralTypes.h and be protected by a FR_GENERAL_TYPES define in order:
- to be shared between different FlexRay Transceiver Drivers
- to be included into the FrIf
If different FlexRay Transceiver Drivers are used, only one instance of this file has to be included in the source tree.⌋( )

[FrTrcv074] ⌈

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

⌋(BSW00375)

8.3 Function definitions

[FrTrcv043] ⌈Avoidance of generic interfaces
All Basic Software Modules shall not use generic interfaces. A ‘generic interface’ is an interface without a defined scope and content.⌋(BSW00329)

[FrTrcv089] ⌈Parameter content shall be unique within the module
The same intention, logical contents or semantic shall be placed in one parameter only (There must not be several parameters with the same intention, logical contents or semantic).⌋(BSW00390)

[FrTrcv090] ⌈Parameter shall have unique names
A parameters name must be unique per module. If the parameter is exported it must be unique to all modules using this parameter.⌋(BSW00391)

[FrTrcv092] ⌈Parameters shall have a range

Each parameter shall have a list of valid values or the minimum as well as maximum values shall be specified. ⌋(BSW00393)

[FrTrcv093] ⌈Specify the scope of the parameters
A parameter may only be applicable for the module it is defined in. In this case, the parameter is marked as “local”. Alternatively, the parameter may be shared with other modules (i.e. exported). In that case, the scope shall list the names of the other modules sharing this parameter. Each parameter shall only be defined once in one module. All other modules sharing the parameter must not define the parameter again. Instead, the parameter is to be imported. This is applicable for c--code as well as for .XML configuration. ⌋(BSW00394)

[FrTrcv094] ⌈List the required parameters (per parameter)
The Basic Software Module specifications must list configuration parameters of this or other modules this parameter relies on. A dependency is for example: the value of another parameter influences or invalidates the setting of this parameter. ⌋(BSW00395)

[FrTrcv104] ⌈Check module initialization
A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called. The initialization function of the BSW modules shall set the static status variable to a value not equal to 0. Exception shall be the “<Module name>_GetVersionInfo” function. It shall be possible to call this function at any time. ⌋(BSW00406)

8.3.1 FrTrcv_Init

[FrTrcv322] ⌈

Service name:	FrTrcv_Init

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

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

⌋( )

[FrTrcv008] ⌈Initialization interface
The FlexRay transceiver driver initializes variables and hardware resources in a separate initialization function. This function shall be named FrTrcv_Init().
Note: According to EcuM2562: Drivers which serve wakeup sources must be re-initialized in the restart block. The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back (see 7.7.4.1 WAKEUP I). ⌋(BSW00101)

[FrTrcv228] ⌈Initialization Sequence for FlexRay Transceiver Driver
The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack.
To start the ECU from power-up or reset, a fixed sequence of driver and manager initialization is necessary to reach the required startup times and to set the FlexRay stack into working state. The sequence itself depends on many requirements, partly dependent on the FlexRay controller and the power supply concept⌋ (BSW05134)

[FrTrcv230] ⌈Initialize the FlexRay Transceiver Driver
The FlexRay Transceiver Driver shall provide an API to initialize the driver internally and set then all attached FlexRay Transceivers in their preselected operation modes.

- The FlexRay Transceiver Driver must be initialized during the power-up/reset sequence of the ECU.
- Depending on the used drivers to control the transceivers (e.g. DIO, SPI), they must be already available and working when the FlexRay Transceiver Driver is initialized.
- The wake-up reason has to be detected and stored during the execution of the driver initialization, too. ⌋ (BSW05137)

[FrTrcv270] ⌈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 FrTrcv434. ⌋ ( )

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.

[FrTrcv437] ⌈In case of a fault during transceiver access, the initialization process shall be restarted from the beginning. It shall be retried until the retry counter exceeds the number specified by FrTrcvRetryCountInInit. If the process doesn't succeed, the function FrTrcv_Init shall raise a development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also FrTrcv237). ⌋ ( )

[FrTrcv390] ⌈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>.⌋ ( )

[FrTrcv438] ⌈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). ⌋( )

[FrTrcv362] ⌈The driver has to notify ECU State Manager by invoking the EcuM_SetWakeupEvent service once only if a wakeup event is detected. ⌋( )

[FrTrcv363] ⌈The driver has to detect a pending wakeup event during the initialization. ⌋( )

[FrTrcv112] ⌈The init function in general shall have no parameter. ⌋(BSW00414)

[FrTrcv065] ⌈The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void. ⌋(BSW00358)

[FrTrcv366] ⌈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. ⌋( )

[FrTrcv367] ⌈The FlexRay Transceiver Driver driver shall support a wakeup ISR if supported by hardware. ⌋( )

[FrTrcv414] ⌈Hardware Resetting Function on Bus Driver
The FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality
Hint: When trouble occurs in the hardware level, it is likely to fix the cause by resetting the hardware. This function shall be executed when a configurable amount of errors are detected in by the FlexRay modules and have been reported to DEM. ⌋(BSW05214)

[FrTrcv455] ⌈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

[FrTrcv323] ⌈

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.	

」( )

[FrTrcv252] 「Set FlexRay Transceiver Operation Mode
The FlexRay Transceiver Driver shall provide a software interface to set the operation mode of a specific FlexRay Transceiver device. 」(BSW05166)

[FrTrcv392] 「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. 」( )

[FrTrcv393] 「After setting FlexRay Trcv to STANDBY the FrTrcv shall call a configurable callout function (see FrTrcv456_Conf), in which the integrator can enable the interrupt pin. If optional states are used, this requirement applies, whenever a transition from a state not capable to a state capable of detecting and/or latching wake up occurs (e.g.: a transition from RECEIVEONLY to SLEEP) and if wake up is supported by hardware. 」( )

[FrTrcv086] 「The callout function of FrTrcv393 shall be pre compile configurable, its name is provided in FrTrcv456_Conf. 」(BSW00387)

[FrTrcv474] 「A mode request of the current mode is allowed and shall not lead to an error even if DET is enabled. 」( )

[FrTrcv064] 「Standard API return type:
The Std_ReturnType shall normally be used with value E_OK or E_NOT_OK. If those return values are not sufficient user specific values can be defined by using the 6 least specific bits. 」(BSW00357)

[FrTrcv272] 「The function FrTrcv_SetTransceiverMode shall switch the internal state of the transceiver identified by FrTrcv_TrcvIdx to the state indicated by FrTrcv_TrcvMode. 」( )

[FrTrcv273] ⌈The function FrTrcv_SetTransceiverMode shall return E_NOT_OK and doesn't change the current state if a transition not defined in FrTrcv_TrcvModeType is requested. ⌋( )

[FrTrcv274] ⌈If an optional state is NOT supported by the transceiver and ECU hardware, the function FrTrcv_SetTransceiverMode shall switch to an equivalent state. ⌋( )

[FrTrcv440] ⌈If the FlexRay transceiver and ECU hardware does not support a receive only state, FrTRCV_TRCVMODE_NORMAL shall be used. ⌋( )

[FrTrcv236] ⌈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.) ⌋( )

[FrTrcv278] ⌈In case of a fault during transceiver access, the function FrTrcv_SetTransceiverMode shall raise development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also FrTrcv237) and return E_NOT_OK. ⌋( )

[FrTrcv368] ⌈The API function calls to the FlexRay Transceiver Driver shall be synchronuous. ⌋( )

[FrTrcv275] ⌈If development error detection for the module FrTrcv is enabled: If the parameter FrTrcv_TrcvIdx is not within the allowed range, the function FrTrcv_SetTransceiverMode shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX and return E_NOT_OK. ⌋( )

[FrTrcv276] ⌈If development error detection for the module FrTrcv is enabled: If the mode transition fails (FrTrcv452), the function FrTrcv_SetTransceiverMode shall raise the following development error and return E_NOT_OK:

- FRTRCV_E_FR_TRCV_NOT_STANDBY:
Transition to FRTRCV_TRCVMODE_STANDBY failed
- FRTRCV_E_FR_TRCV_NOT_NORMAL:
Transition to FRTRCV_TRCVMODE_NORMAL failed
- FRTRCV_E_FR_TRCV_NOT_SLEEP:
Transition to FRTRCV_TRCVMODE_SLEEP failed
- FRTRCV_E_FR_TRCV_NOT_RECEIVEONLY:
Transition to FRTRCV_TRCVMODE_RECEIVEONLY failed⌋( )

[FrTrcv452] ⌈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().⌋( )

[FrTrcv277] ⌈If development error detection for the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_SetTransceiverMode shall raise development error FRTRCV_E_FR_UNINIT and return E_NOT_OK. ⌋( )

[FrTrcv415] ⌈Hardware Checking Function on Bus Driver
The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly.
This functionality ensures that the hardware is working as expected.
Improvement of hardware reliability. ⌋(BSW05213)

[FrTrcv454] ⌈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

[FrTrcv324] ⌈

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.	

⌋( )

[FrTrcv253] ⌈The function FrTrcv_GetTransceiverMode shall return the state of the transceiver identified by FrTrcv_TrcvIdx. ⌋(BSW05167)

[FrTrcv281] ⌈In case of a fault during transceiver access, the function FrTrcv_GetTransceiverMode shall raise the development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also FrTrcv237) and return E_NOT_OK. ⌋ (BSW05151)

See FrTrcv_Init (FrTrcv270) 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.

[FrTrcv279] ⌈If development error detection for the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_GetTransceiverMode shall raise the development error FRTRCV_E_FR_INVALID_TRCVIDX and return E_NOT_OK. ⌋ ( )

[FrTrcv280] ⌈If development error detection for the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_GetTransceiverMode shall raise the development error FRTRCV_E_FR_UNINIT and return E_NOT_OK. ⌋ ( )

[FrTrcv397] ⌈When the caller provides a NULL pointer as a parameter value to the API
FrTrcv_GetTransceiverMode, the return value shall be E_NOT_OK and the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DET if development error detection is enabled. ⌋ ( )

8.3.4 FrTrcv_GetTransceiverWUReason

[FrTrcv325] ⌈

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.	

⌋( )

[FrTrcv232] ⌈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. ⌋(BSW05144)

[FrTrcv284] ⌈In case of a fault during transceiver access, the function FrTrcv_GetTransceiverWUReason shall raise development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also FrTrcv237) and return E_NOT_OK. ⌋( )

Please be aware, that if more than one bus is available, each bus may report a different wake up reason. E.g. if an ECU has FlexRay, a wake up by FlexRay may occur and the incoming data may cause an internal wake up for another FlexRay bus.

[FrTrcv453] ⌈The FlexRay Transceiver Driver shall report the wake up reason in the order defined by FrTrcv074.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.

[FrTrcv282] ⌈If development error detection of the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_GetTransceiverWUReason shall raise the development error FRTRCV_E_FR_INVALID_TRCVIDX and return E_NOT_OK. ⌋( )

[FrTrcv283] ⌈If development error detection of the module FrTrcv is enabled: if the transceiver has not been initialized, the function FrTrcv_GetTransceiverWUReason shall raise the development error FRTRCV_E_FR_UNINIT and return E_NOT_OK. ⌋( )

[FrTrcv398] ⌈When the caller provides a NULL pointer as a parameter value to the API FrTrcv_GetTransceiverWUReason, the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DET if development error detection is enabled. ⌋( )

8.3.5 FrTrcv_GetVersionInfo

[FrTrcv326] ⌈

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

⌋ ( )

[FrTrcv001] ⌈Version identification: The FlexRay transceiver driver shall support a version information API. ⌋ (BSW00003)

[FrTrcv105] ⌈Function to read out published parameters: The function FrTrcv_GetVersionInfo shall return the version information of the FrTrcv module.

The version information consists of three parts:
• Two bytes for the vendor ID
• One byte for the module ID
• Three bytes version number.
The numbering shall be vendor specific; it consists of:
The major, the minor and the patch version number of the module.
The AUTOSAR specification version number shall not be included.
It shall be possible to call this function at any time
(e.g. before the init function is called). ⌋ (BSW00407)

[FrTrcv109] ⌈Get version info keyword: The FlexRay transceiver driver shall apply the following naming rule for enabling/disabling the existence of the API.
FrTrcv_GetVersionInfo(…) (see BSW00407): FRTRCV_VERSION_INFO_API⌋ (BSW00411)

[FrTrcv073] ⌈Module vendor identification
All Basic Software Modules shall provide readable module vendor identification (according to HIS) in their published parameters.
Naming convention:
FRTRCV_VENDOR_ID
The vendor ID shall be represented in uint16 (16 bit).
Allow identification of module vendor ⌋ (BSW00374)

[FrTrcv078] ⌈Module identification
All software modules shall provide a module identifier in the header file and in the module XML description file.
The value shall be taken from the Basic Software Module List.

Naming convention:
FRTRCV_MODULE_ID

The module ID shall be represented in uint8 (8 bit). ⌋(BSW00379)

[FrTrcv034] ⌈Format of module version numbers:
Each AUTOSAR Basic Software Module file shall provide version numbers in the header file as defined below:
Naming convention:
FRTRCV_SW_MAJOR_VERSION
FRTRCV_SW_MINOR_VERSION
FRTRCV_SW_PATCH_VERSION
FRTRCV_AR_RELEASE_MAJOR_VERSION
FRTRCV_AR_RELEASE_MINOR_VERSION
FRTRCV_AR_RELEASE_REVISION_VERSION
AR: Major/minor/patch version number of AUTOSAR specification which the appropriate implementation is based on.
SW: Major/minor/patch version number of the vendor specific implementation of the module.
The numbering shall be vendor specific, but it shall follow requirement BSW00321.

Each number shall be represented in uint8 (8 bit). ⌋(BSW00318)

[FrTrcv285] ⌈The function FrTrcv_GetVersionInfo shall return the version information of the FrTrcv module, NOT the version of the FlexRay transceiver hardware. ⌋( )

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

FRTRCV_GET_VERSION_INFO. ⌋( )

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

[FrTrcv396] ⌈When a NULL pointer is passed as a parameter value of FrTrcv_GetVersionInfo, the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DETshall be reported to DET if development error detection is enabled. ⌋( )

8.3.6 FrTrcv_ClearTransceiverWakeup

[FrTrcv329] ⌈

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.	

」( )

[FrTrcv247] ⌈The function FrTrcv_ClearTransceiverWakeup shall clear a pending wake up event on the transceiver identified by FrTrcv_TrcvIdx. ⌋(BSW05161)

[FrTrcv371] ⌈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.

[FrTrcv306] ⌈In case of a fault during transceiver access, the function FrTrcv_ClearTransceiverWakeup shall raise the development error FRTRCV_E_FR_NO_CONTROL_TRCV if development error detection for the module FrTrcv is enabled (see also FrTrcv237) and return E_NOT_OK. ⌋ (BSW05151)

[FrTrcv304] ⌈If development error detection is enabled for the module FrTrcv: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_ClearTransceiverWakeup shall raise the development error code FRTRCV_E_FR_INVALID_TRCVIDX and return E_NOT_OK. ⌋( )

[FrTrcv305] ⌈If development error detection is enabled for the module FrTrcv: if the transceiver has not been initialized, the function FrTrcv_ClearTransceiverWakeup shall raise the development error code FRTRCV_E_FR_UNINIT and return E_NOT_OK. ⌋( )

8.3.7 FrTrcv_CheckWakeupByTransceiver

[FrTrcv331] ⌈

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:	--	

⌋ ( )

[FrTrcv364] ⌈The driver shall notify ECU State Manager of wakeup events if triggered by the API call FrTrcv_CheckWakeupByTransceiver. ⌋ ( )

[FrTrcv233] ⌈Notification for Wake-up by Bus
The FlexRay Transceiver Driver shall support a notification to inform higher layers about the wake-up by bus. It must be possible to support more than one bus within the ECU with this notification.
The FlexRay Transceiver Driver shall call this notification when the transceiver detects a wake-up by bus.
The FlexRay Transceiver Driver is notified by a notification from the underlying SPI or DIO driver in that case. The notification is executed in the context of the caller (may be interrupt context!). Because the delay from wake-up detection until the start of the necessary actions has a large influence on the startup time of an ECU, this event shall be processed internally and transferred immediately via this notification to the next layer.
The call context and the reaction time depend on the call context of the lower layer DIO or SPI. In case of interrupt it is very fast but data consistency issues must be covered in all layers. In case of polling data consistency issues are reduced but reaction time may be too slow.
Rationale: Support wake-up by FlexRay Transceiver devices.
Use Case: The FlexRay Transceiver detects a wake-up condition on the bus and shows this to the µC via e.g. a port pin.
Further handling depends on current ECU state. Assuming the ECU is halted, the change on the port may terminate the "HALT" statement and let the processor continue its work. The assigned port interrupt will be executed and this handler is called. Now, the FlexRay Transceiver Driver will store the wake-up reason and give the call via this notification to e.g. the NM to let the NM decide how to handle the event. ⌋ (BSW05147)

[FrTrcv262] ⌈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. ⌋ ( )

[FrTrcv311] ⌈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. ⌋ ( )

[FrTrcv374] ⌈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. ⌋ ( )

[FrTrcv375] ⌈The FlexRay Transceiver Driver shall check for wake up events immediately after the API call FrTrcv_SetTransceiverMode if supported by hardware. ⌋ ( )

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

[FrTrcv379] ⌈Calling FrTrcv_CheckWakeupByTransceiver in an interrupt context shall be supported. Hint: This has to be documented according to (BSW00333). 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. ⌋ ( )

[FrTrcv380] ⌈Calling FrTrcv_CheckWakeupByTransceiver by a polling process in sleep mode shall be supported. ⌋ ( )

[FrTrcv312] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is out of range, the function FrTrcv_CheckWakeupByTransceiver shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX. ⌋ ( )

[FrTrcv313] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_CheckWakeupByTransceiver shall raise development error FRTRCV_E_FR_UNINIT. ⌋ ( )

[FrTrcv229] ⌈Configuration "Notification for Wake-up by Bus"
The FlexRay Transceiver Driver shall support the compile time configuration of one notification to a higher layer for change notification for "wake-up by bus" events.
One wake-up by bus event notification shall be supported to one higher layer.
If a transceiver device does not support "wake-up by bus", this notification is never called for this bus.

Efficient coupling between FlexRay Transceiver Driver and higher layer. ⌋ (BSW05136)

[FrTrcv234] ⌈Support for Wake-up During Sleep Transition
The FlexRay Transceiver Driver shall support situations where a wake-up by bus occurs at the same moment the transition to standby/sleep is executed by the driver.
Wake-up by bus is always asynchronous to the internal transition to sleep.
In worst case, the wake-up occurs during the transition to sleep.
This situation must be covered by the design and explicitly tested for each ECU.
The driver shall create a wake-up notification by bus immediately after the API to enter the standby/sleep mode has finished.
The calling/controlling component (NM or ECU state manager) must be capable to handle the wake-up immediately after requesting the standby/sleep.
Safe wake-up and sleep handling.

All busses with a wake-up by bus are affected. ⌋ (BSW05148)

8.3.8 FrTrcv_GetTransceiverError

[FrTrcv419] ⌈

Service name:	FrTrcv_GetTransceiverError	

Syntax:	`Std_ReturnType FrTrcv_GetTransceiverError(`
	`  uint8 FrTrcv_TrcvIdx,`
	`  uint8 FrTrcv_BranchIdx,`
	`  uint32* FrTrcv_BusErrorState`
	`)`	

Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
	FrTrcv_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	FrTrcv_BusErrorState	Pointer to structure of detailed transceiver error state; - Parameter is reference to variable: The transceiver driver will write the current transceiver error state information according to FrTrcv420 there, if the transceiver supports this information.
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver error state has been provided E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx or FrTrcv_BranchIdx is out of range or the

		transceiver error state is not available. Output parameters remain unchanged.
Description:	All mandatory errors defined by the FlexRay EPL [5] which are supported by the FlexRay transceiver hardware can be accessed via this API:In addition to errors on the physical layer and local to the ECU hardware, a global error flag is provided.	

⌋( )

[FrTrcv412] ⌈Detect Error Information in Bus Driver
The FlexRay Transceiver Driver shall provide an API that detects errors in the bus driver and notifies the application level.

The FlexRay modules should provide information that only the modules can detect. ⌋ (BSW05212)

[FrTrcv420] ⌈The FlexRay Transceiver Driver shall support all mandatory errors defined by the FlexRay EPL [5] if supported by hardware:

Availability	Topology	Description	Bit
Mandatory	Global error	Any of the mandatory errors defined in this table, please see FrTrcv457, FrTrcv458	0
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between bus lines according to [5]	1
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between positive bus line and ground according to [5]	2
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between positive bus line and power supply according to [5]	3
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between negative bus line and power supply according to [5]	4
Mandatory	on the bus (i. e. external to the ECU):	Short circuit between negative bus line and ground according to [5]	5
Mandatory	on the bus (i. e. external to the ECU):	Any bus fault according to [5], which cannot be resolved according to the description of bit 1...5	6
Mandatory	Local error	Under voltage of transceiver power supply according to [5]	7
Mandatory	Local error	FlexRay transceiver is permanently enabled according to [5]	8
Mandatory	Local error	Over temperature of transceiver according to [5]	9

⌋( )

[FrTrcv421] ⌈Additional transceiver errors, which are supported by hardware shall be appended to the table in FrTrcv420. ⌋( )

[FrTrcv439] ⌈When the caller provides a NULL pointer as a parameter value to the API FrTrcv_GetTransceiverError the return value shall be E_NOT_OK and the development error FRTRCV_E_FR_TRCV_NULL_PTR shall be reported to DET if development error detection is enabled.⌋( )

[FrTrcv444] ⌈The FlexRay Transceiver Driver shall identify bus faults per branch on active star transceivers.⌋( )

[FrTrcv449] ⌈The FlexRay Transceiver Driver shall ignore the parameter FrTrcv_BranchIdx in case the transceiver is not an active star device.⌋( )

[FrTrcv457] ⌈The FlexRay Transceiver Driver shall set bit 0 of FrTrcv_BusErrorState if the state of ERRN is active low for transceivers according to class A of [5]⌋( )

[FrTrcv458] ⌈The FlexRay Transceiver Driver shall set bit 0 of FrTrcv_BusErrorState if any of bit 1...9 is set for transceivers according to class B of [5]⌋( )

[FrTrcv459] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_GetTransceiverError shall raise development error FRTRCV_E_FR_UNINIT.⌋( )

[FrTrcv460] ⌈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.⌋( )

[FrTrcv461] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_BranchIdx is out of range, the function FrTrcv_GetTransceiverError shall raise development error FRTRCV_E_FR_INVALID_BRANCHIDX.

FrTrcv_DisableTransceiverBranch⌋( )

[FrTrcv442] ⌈The FlexRay Transceiver Driver shall disable the faulty branches of active stars

Service name:	FrTrcv_DisableTransceiverBranch	
Syntax:	`Std_ReturnType FrTrcv_DisableTransceiverBranch( uint8 FrTrcv_TrcvIdx, uint8 FrTrcv_BranchIdx )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
	FrTrcv_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters	None	

(inout):		
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver branch has been disabled E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx or FrTrcv_BranchIdx is out of range. Branch state remains unchanged.
Description:	This function disables the specified branch on the addressed (active star) transceiver.	

⌋( )

[FrTrcv462] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_DisableTransceiverBranch shall raise development error FRTRCV_E_FR_UNINIT. ⌋( )

[FrTrcv463] ⌈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. ⌋( )

[FrTrcv464] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_BranchIdx is out of range, the function FrTrcv_DisableTransceiverBranch shall raise development error FRTRCV_E_FR_INVALID_BRANCHIDX.

FrTrcv_EnableTransceiverBranch⌋( )

[FrTrcv443] ⌈The FlexRay Transceiver Driver shall enable the branches of active stars synchronously to the FlexRay bus schedule

Service name:	FrTrcv_EnableTransceiverBranch	
Syntax:	`Std_ReturnType FrTrcv_EnableTransceiverBranch( uint8 FrTrcv_TrcvIdx, uint8 FrTrcv_BranchIdx )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrTrcv_TrcvIdx	This zero based index identifies the transceiver within the context of the transceiver driver to which the API call has to be applied.
	FrTrcv_BranchIdx	This zero based index identifies the branch of the (active star) transceiver to which the API call has to be applied.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: will be returned if the transceiver branch has been enabled E_NOT_OK: will be returned if the parameter FrTrcv_TrcvIdx or FrTrcv_BranchIdx is out of range. Branch state remains unchanged.
Description:	This function enables the specified branch on the addressed (active star) transceiver.	

⌋( )

[FrTrcv465] ⌈If development error detection of module FrTrcv is enabled: if the FrTrcv module is not initialized, the function FrTrcv_EnableTransceiverBranch shall raise development error FRTRCV_E_FR_UNINIT. ⌋( )

[FrTrcv466] ⌈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. ⌋( )

[FrTrcv467] ⌈If development error detection of module FrTrcv is enabled: if the parameter FrTrcv_BranchIdx is out of range, the function FrTrcv_EnableTransceiverBranch shall raise development error FRTRCV_E_FR_INVALID_BRANCHIDX.
Scheduled functions
This section lists functions that are directly called by Basic Software Scheduler.
FrTrcv_MainFunction⌋( )

[FrTrcv330] ⌈

Service name:	FrTrcv_MainFunction

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

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

⌋( )

[FrTrcv020] ⌈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. ⌋(BSW00172)

[FrTrcv072] ⌈Main processing function naming convention: The main function of the FlexRay transceiver driver shall be named FrTrcv_MainFunction. ⌋(BSW00373)

[FrTrcv075] ⌈Return type and parameters of main processing functions: The main processing function of the FlexRay transceiver driver shall have no parameters and no return value. ⌋(BSW00376)

[FrTrcv126] ⌈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. ⌋(BSW00428)

[FrTrcv340] ⌈The function FrTrcv_MainFunction shall scan all busses in STANDBY and SLEEP for wake up events and store them internally.

Note: EcuM will invoke EcuM_CheckWakeup. This results in the invocation of FrTrcv_CheckWakeupByTransceiver. So, in case of POLLING, the API FrTrcv_CheckWakeupByTransceiver shall invoke the EcuM_SetWakeupEvent. 」( )

[FrTrcv122] 「The function FrTrcv_MainFunction shall be implemented in such a way that it can run inside a basic task (scheduled by the AUTOSAR RTE). 」(BSW00424)

[FrTrcv372] 「The Basic Software Scheduler shall execute FrTrcv_MainFunction with a period configured by the parameter FRTRCV_MAIN_FUNCTION_CYCLE_TIME. See FrTrcv343_Conf for more details. 」( )

[FrTrcv373] 「If a cycle time of 0 is configured in FrTrcvMainFunctionCycleTime this function is not executed by the Basic Software Scheduler and need not be present in compiled code. See FrTrcv343_Conf for more details. 」( )

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

[FrTrcv123] 「Trigger conditions for schedulable objects
The BSW module description template shall provide means to model the following trigger conditions of schedulable objects:
Cyclic timings (fixed and selectable during runtime)
Sporadic events」(BSW00425)

[FrTrcv436] 「Error Information in Bus Driver
The FrTrcv_MainFunction shall call API of BSW05212 (FrTrcv412)
FrTrcv_GetTransceiverError () periodically to detect error information in BD. 」(BSW05203)

[FrTrcv341_Conf] 「A pre-compile configuration parameter FrTrcvDevErrorDetect shall determine whether this functionality FrTrcv436 is activated.
Note: The FlexRay modules should provide information that only the modules can detect.
Applications could take actions to recover the failure cause like resetting the modules when they receive this error information. 」( )

8.4 Call-back notifications

This is a list of functions provided for lower layer modules.
E.g. the SPI driver might provide a call back whenever an transfer is finished.
(There are none).
Please see FrTrcv393 and FrTrcv456_Conf.

8.5 Expected Interfaces

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

8.6 Mandatory Interfaces

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

[FrTrcv332] ⌈

API function	*Description*
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function.

⌋ ( )

8.7 Optional Interfaces

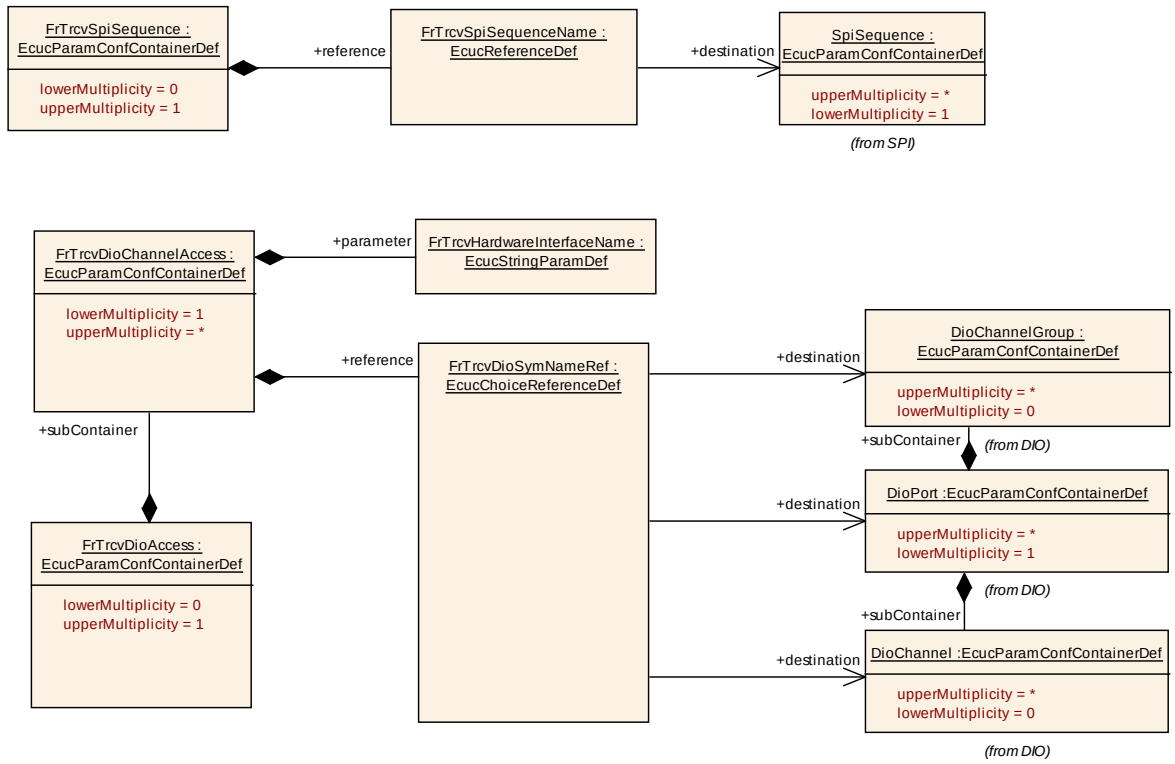
[FrTrcv334] ⌈The FlexRay Transceiver Driver uses these optional Interfaces:

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

⌋ ( )
Configuration of the container FlexRayTransceiverDioAccess FrTrcv145_Conf enables the FlexRay Transceiver Driver to use the API of the DIO module.

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

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



[FrTrcv061] ⌈If FRTRCV_DEV_ERROR_DETECT is configured, the FlexRay Transceiver Driver uses the API of the DET module. ⌋(BSW00350)

8.8 Configurable interfaces

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

[FrTrcv475] ⌈If the optional configuration parameter FrTrcvDemReportErrorStatusConfiguration is provided in the global FlexRay Transceiver Driver configuration, the function defined by this configuration parameter shall be called instead of Dem_ReportErrorStatus with the same signature. ⌋( )

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

[FrTrcv019] ⌈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). ⌋(BSW00171)

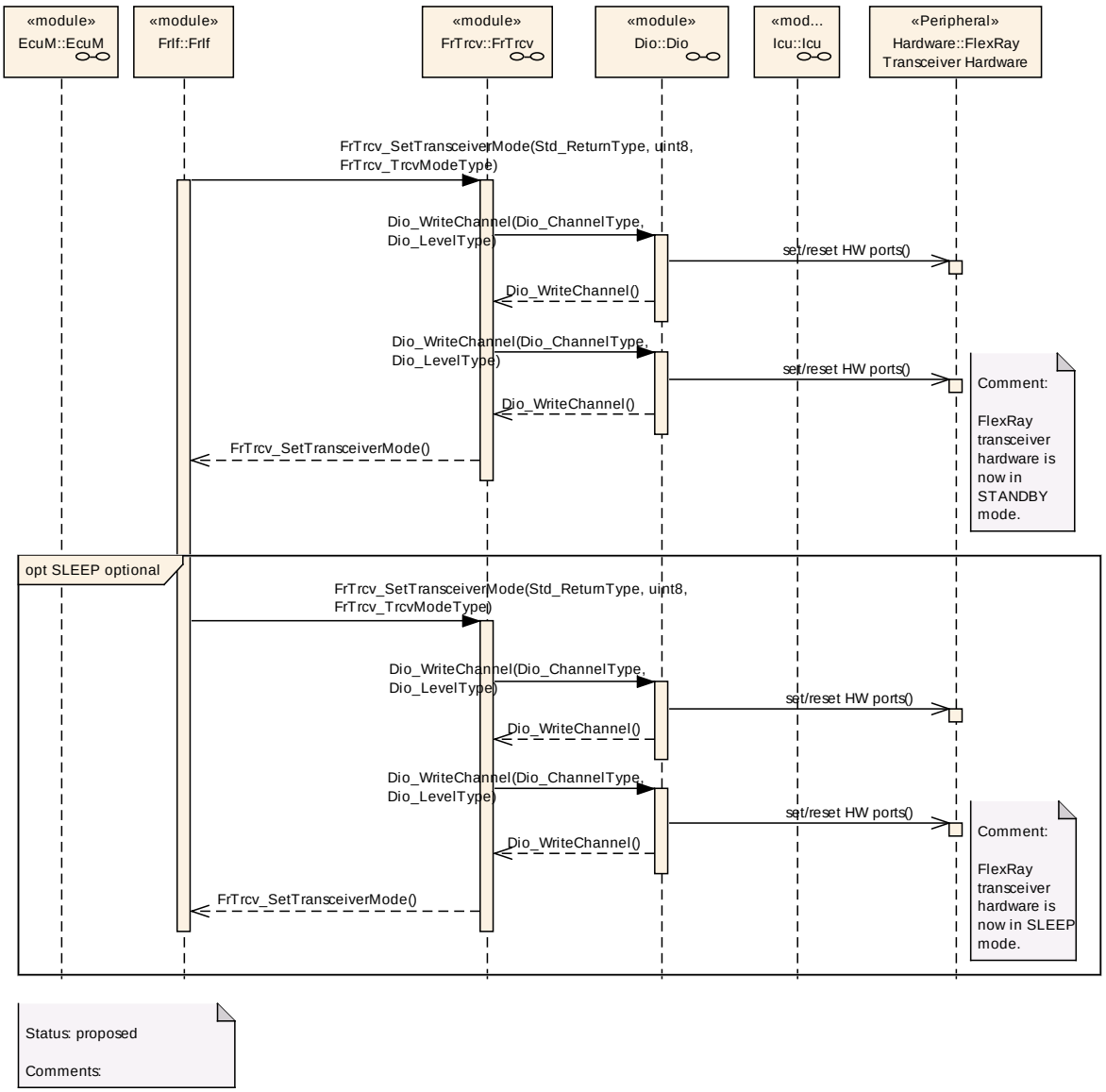
If branches of active stars using FrTrcv357_Conf are configured, these additional APIs shall be available:

The API to enable branches FrTrcv443

The API to disable branches FrTrcv442
The API to detect errors of branches FrTrcv419

Please see FrTrcv393 and FrTrcv456_Conf.

9 Sequence diagrams



ATTENTION: These sequence charts are application examples only. They focus on interaction between the FlexRay transceiver driver (FrTrcv), FlexRay Interface (Frlf) and BSW module Dio. For details see [7] and [14]. Depending on FlexRay transceiver hardware one or more calls to Dio_WriteChannels may be necessary.
For details on FlexRay Transceiver wakeup please refer to chapter 9 of [13].

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals.

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

Chapter 10.3 specifies published information of the module FrTrcv.

10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture
- AUTOSAR ECU Configuration Specification
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

The following is only a short survey of the topic and it will not replace the ECU Configuration Specification document.

10.1.1 Configuration and configuration parameters

Configuration parameters define the variability of the generic part(s) of an implementation of a module. This means that only generic or configurable module implementation can be adapted to the environment (software/hardware) in use during system and/or ECU configuration.

The configuration of parameters can be achieved at different times during the software process: before compile time, before link time or after build time. In the following, the term “configuration class” (of a parameter) shall be used in order to refer to a specific configuration point in time.

[FrTrcv106] ⌈Configuration parameter naming convention
All AUTOSAR Basic Software Modules configuration parameters shall be named according to the following naming rules:
Naming convention:

- <ModuleShortName><ParameterName>

< ModuleShortName > is the prefix according to [1]

If < ParameterName > consists of several words, they shall be written in UpperCamelCase
Avoidance of name clashes, uniform AUTOSAR configuration naming.
< ModuleShortName > shall be derived from [1], [DOC_MOD_LIST] (2…8 characters) ⌋ (BSW00408)

10.1.2 Variants

[FrTrcv095] ⌈Configuration classes: Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class. ⌋(BSW00396)

10.1.3 Containers

[FrTrcv087] ⌈Containers structure the set of configuration parameters. This means:

- all configuration parameters are kept in containers.
- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters. ⌋(BSW00388)

10.2 Containers and configuration parameters

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

10.2.1 Variants

[FrTrcv314] ⌈VARIANT-PRE-COMPILE: Only parameters with "Pre-compile time" configuration are allowed in this variant.⌋( )

[FrTrcv315] ⌈VARIANT-LINK-TIME: Only parameters with "Pre-compile time" and "Link time" are allowed in this variant.⌋( )

[FrTrcv316] ⌈VARIANT-POST-BUILD: Parameters with "Pre-compile time", "Link time" and "Post-build time" are allowed in this variant.⌋( )

[FrTrcv317] ⌈The FrTrcv module shall support pre compile time configuration.⌋ (BSW00397)

[FrTrcv318] ⌈The FrTrcv module shall not use link time parameters within the FlexRay Transceiver Driver.⌋( )

[FrTrcv319] ⌈The FrTrcv module shall not use post build time configuration changes by flashing within the FlexRay Transceiver Driver.⌋( )

10.2.2 General configuration requirements

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

[FrTrcv010] ⌈A configuration tool is used to generate the configuration data and code if any.⌋(BSW00159)

[FrTrcv011] ⌈Human-readable configuration data⌋(BSW00160)

[FrTrcv018] ⌈Data for reconfiguration of AUTOSAR SW-Components⌋ (BSW00170)

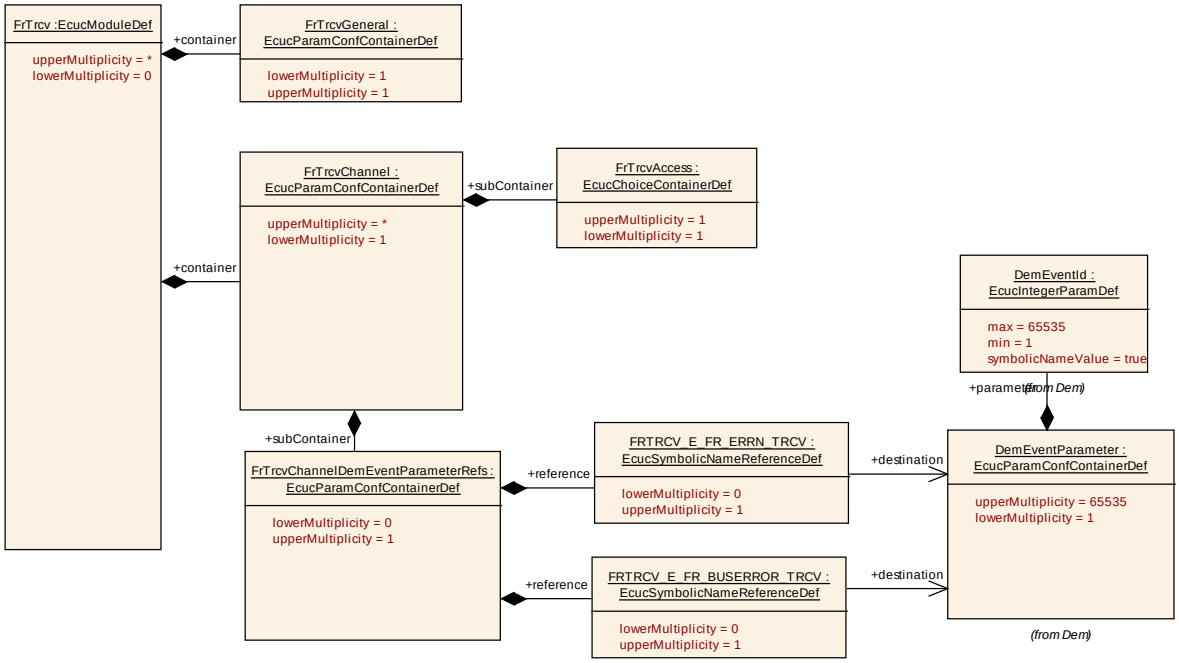
[FrTrcv047] ⌈All Basic Software Modules shall provide an XML file that contains the meta data which is required for the SW configuration and integration process.⌋(BSW00334)

[FrTrcv225] ⌈Configuration Data for FlexRay Transceiver⌋(BSW05131)

[FrTrcv016] ⌈The configuration tool has to check the validity of the provided input data and the usability in the project context. ⌋(BSW00167)

[FrTrcv080] ⌈The pre--compile time parameters shall be placed into a separate configuration header file. ⌋(BSW00381)

[FrTrcv088] ⌈Containers shall have names.
The configuration of the transceiver is assembled in a container⌋(BSW00389)



10.2.3 FrTrcvGeneral

SWS Item	**FrTrcv055_Conf :**
Container Name	FrTrcvGeneral{FlexRayTransceiverDriverBasic}
Description	Container gives FlexRay transceiver driver basic information.
Configuration Parameters	

SWS Item	**FrTrcv455_Conf :**		
Name	FrTrcvDemReportErrorStatusConfiguration {FRTRCV_DEM_REPORT_ERROR_STATUS_CONFIGURATION}		
Description	Name of a C function which substitutes Dem_ReportErrorStatus.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv341_Conf :**		
Name	FrTrcvDevErrorDetect {FRTRCV_DEV_ERROR_DETECT}		
Description	Switches development error detection and notification on and off. If switched on, #define FRTRCV_DEV _ERROR_DETECT ON shall be generated. If switched off, #define FRTRCV_DEV_ERROR _DETECT OFF shall be generated. Define shall be part of file FrTrcv_Cfg.h.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv447_Conf :**		
Name	FrTrcvErrorCheckDuringCommunication {FRTRCV_ERROR_CHECK_DURING_COMMUNICATION}		
Description	Enable a functionality to check transceiver's state during communication.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**FrTrcv446_Conf :**		
Name	FrTrcvErrorCheckInInit {FRTRCV_ERROR_CHECK_IN_INIT}		
Description	Enable a functionality to check transceiver's state while initialization process of FrTrcv.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**FrTrcv342_Conf :**		
Name	FrTrcvGetVersionInfo {FRTRCV_GET_VERSION_INFO}		
Description	Switches version information API on and off. If switched off, function need not be present in compiled code.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv268_Conf :**
Name	FrTrcvIndex
Description	Specifies the InstanceId of this module instance. If only one instance is present it shall have the Id 0.
Multiplicity	1

Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**FrTrcv343_Conf :**		
Name	FrTrcvMainFunctionCycleTime {FRTRCV_MAIN_FUNCTION_CYCLE_TIME}		
Description	Cyclic call time for function FrTrcvMainFunction in seconds. A call time of 0ms indicates no calls for this function. In this case function need not be present in compiled code.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	`0 .. INF`		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

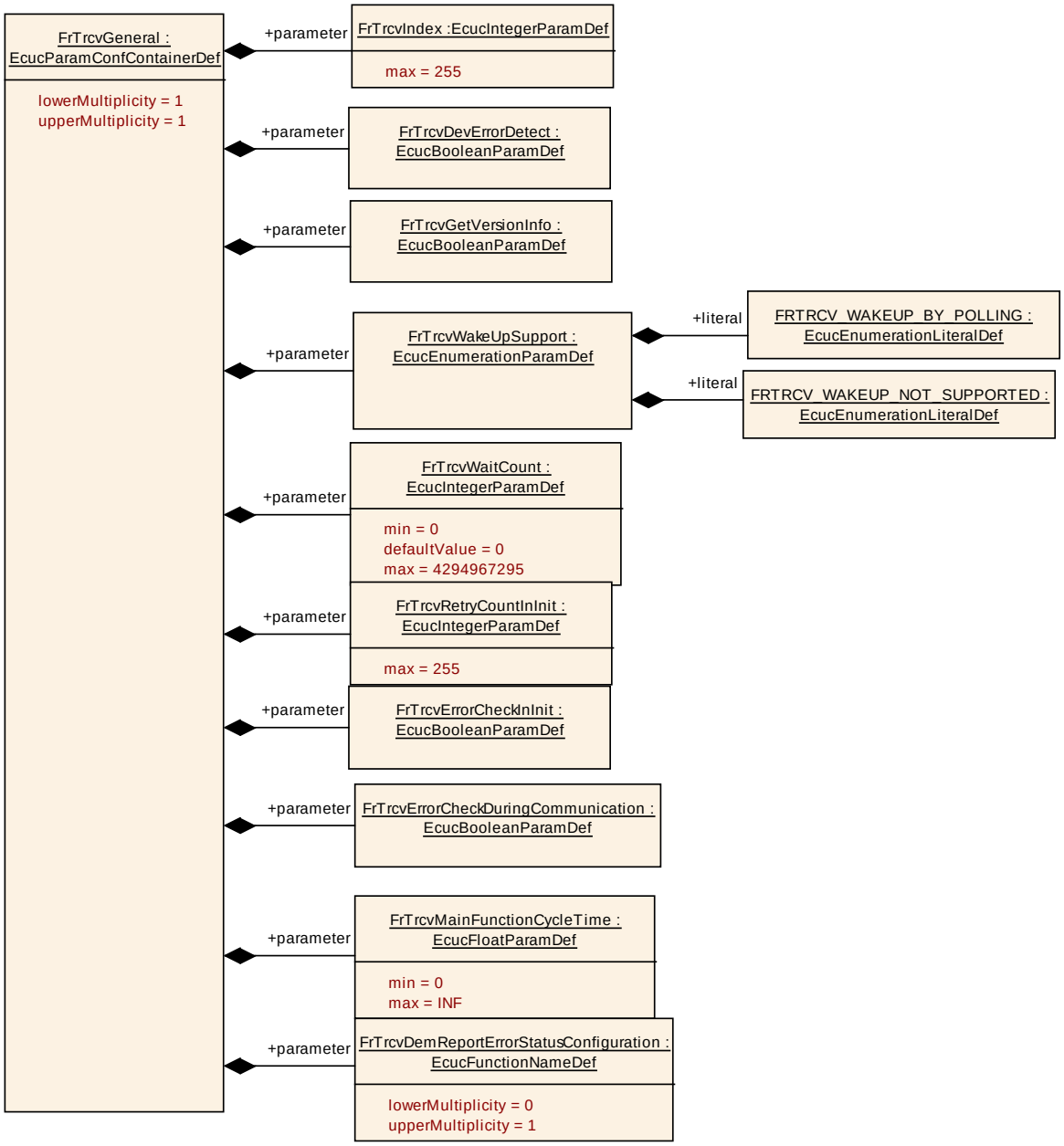
SWS Item	**FrTrcv445_Conf :**		
Name	FrTrcvRetryCountInInit {FRTRCV_RETRY_COUNT_IN_INIT}		
Description	Specifies the number of retry count when error occurs while initialization process of FrTrcv.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**FrTrcv353_Conf :**		
Name	FrTrcvWaitCount		
Description	Wait count for transceiver state changes.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 4294967295		
Default value	0		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrTrcv352_Conf :**
Name	FrTrcvWakeUpSupport {FRTRCV_GENERAL_WAKE_UP_SUPPORT}
Description	Informs whether wake up is supported by polling or whether it is not supported. In case no wake up is supported by FlexRay transceiver hardware setting has to be always NO. Only in case wake up is supported by polling main function FlexRayTrcv_main has to be present in source code. In case of support for wake up either by polling wake up ability may be switched on or off for each channel of one FlexRay transceiver channel

	independently by FrTrcvWakeupByBusUsed.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FRTRCV_WAKEUP_BY_POLLING	Wake up by polling	
	FRTRCV_WAKEUP_NOT_SUPPORTED	Wake up is not supported	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module dependency: FrTrcvWakeupByBusUsed		

No Included Containers



10.2.4 FrTrcvChannel

SWS Item	**FrTrcv091_Conf :**
Container Name	FrTrcvChannel{FlexRayTransceiverNode}
Description	Container gives FlexRay transceiver driver information about a single FlexRay transceiver channel. Any FlexRay transceiver driver has such FlexRay transceiver channels.
Configuration Parameters	

SWS Item	**FrTrcv349_Conf :**
Name	FrTrcvChannelId {FRTRCV_NODE_ID}
Description	Unique identifier of the FlexRay Transceiver Channel.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for

	this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv355_Conf :**		
Name	FrTrcvChannelUsed {FRTRCV_CHANNEL_USED}		
Description	Shall the related FlexRay transceiver channel be used?		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv346_Conf :**		
Name	FrTrcvControlsPowerSupply {FRTRCV_CONTROLS_POWER_SUPPLY}		
Description	Is ECU power supply controlled by this transceiver?		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv456_Conf :**		
Name	FrTrcvEnableInterruptCallout {FRTRCV_ENABLE_INTERRUPT_CALLOUT}		
Description	This parameter defines the existence and the name of a callout function that enables the interrupt pin for the wakeup. If this parameter is omitted no callout shall take place.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv347_Conf :**	
Name	FrTrcvInitState {FRTRCV_INIT_STATE}	
Description	State of FlexRay transceiver after power on. ImplementationType: FrTrcv_TrcvModeType	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FRTRCV_TRCVMODE_NORMAL	Normal operation

			mode
	FRTRCV_TRCVMODE_RECEIVEONLY		Receive only mode
	FRTRCV_TRCVMODE_SLEEP		Sleep operation mode
	FRTRCV_TRCVMODE_STANDBY		Standby operation mode
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

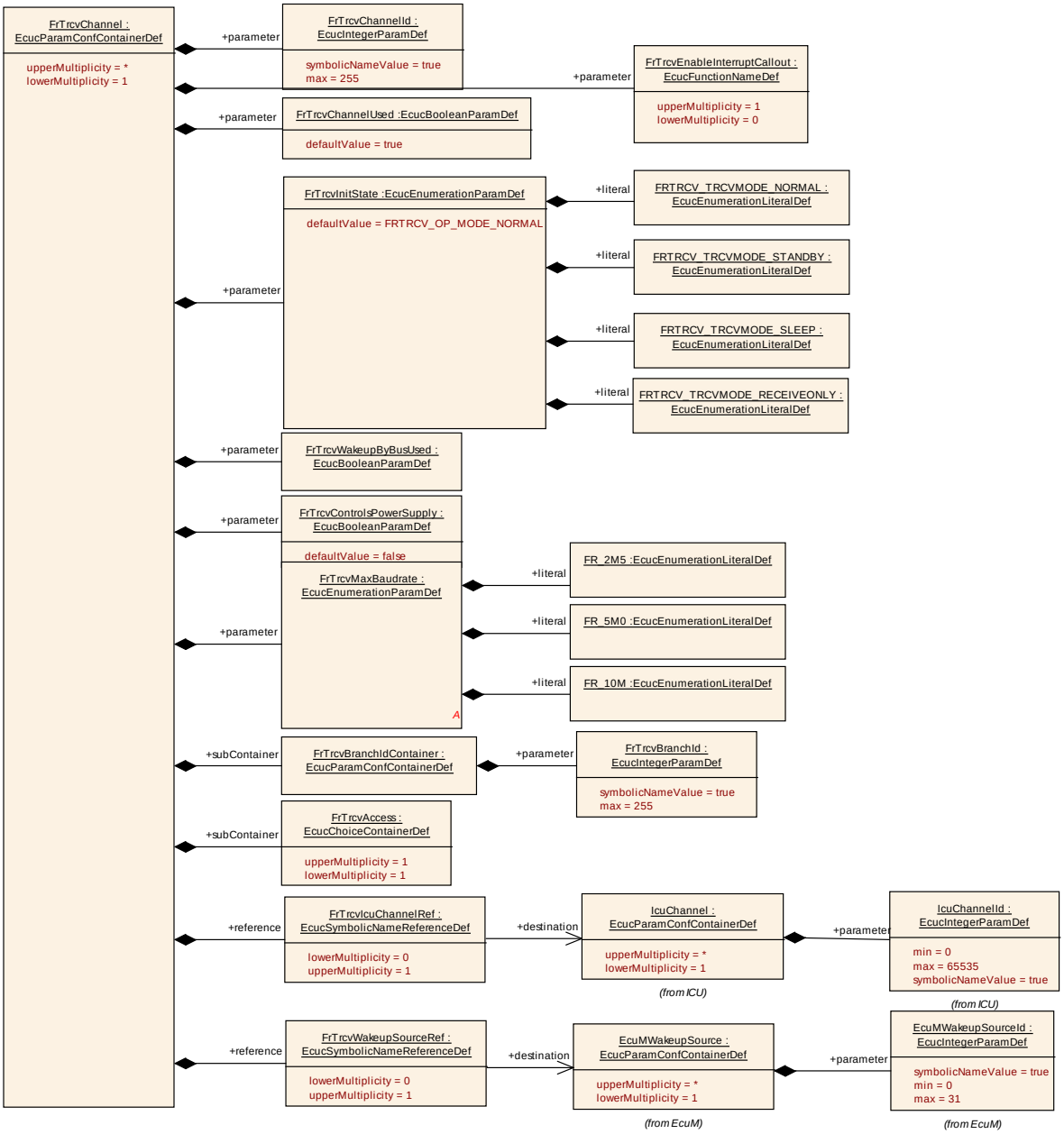
SWS Item	**FrTrcv348_Conf :**		
Name	FrTrcvMaxBaudrate {FRTRCV_MAX_BAUDRATE}		
Description	Max baudrate for transceiver hardware type. Only used for validation purposes. Value shall be configured by configuration tool based on FRTRCV_HARDWARE_NAME and internal information about ability of this hardware typel.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FR_10M	10.0 MBaud	
	FR_2M5	2.5 MBaud	
	FR_5M0	5.0 MBaud	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv350_Conf :**		
Name	FrTrcvWakeupByBusUsed {FRTRCV_WAKEUP_BY_NODE_USED}		
Description	Is wake up by node supported? If FlexRay transceiver hardware does not support wake up by node value is always FALSE. If FlexRay transceiver hardware supports wake up by node value is TRUE or FALSE depending whether it is used or not.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance dependency: FRTRCV_WAKEUP_POLLING		

SWS Item	**FrTrcv384_Conf :**		
Name	FrTrcvIcuChannelRef {FRTRCV_ICU_CHANNEL_REF}		
Description	Reference to the IcuChannel to enable/disable the interrupts for wakeups.		
Multiplicity	0..1		
Type	Reference to [IcuChannel]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**FrTrcv269_Conf :**		
Name	FrTrcvWakeupSourceRef {FRTRCV_WAKEUP_SOURCE_REF}		
Description	Reference to a wakeup source in the EcuM configuration. If FrTrcvWakeUpSupport is configured as FRTRCV_WAKEUP_NOT_SUPPORTED the FrTrcvWakeupSourceRef is not needed. Implementation Type: reference to EcuM_WakeupSourceType		
Multiplicity	0..1		
Type	Reference to [EcuMWakeupSource]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: FrTrcvWakeUpSupport		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrTrcvAccess	1	--
FrTrcvBranchIdContainer	1	Only one SymbolicNameValue can be defined per container. Therefore this container is necessary.
FrTrcvChannelDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.



10.2.5 FrTrcvChannelDemEventParameterRefs

SWS Item	**FrTrvc450_Conf :**
Container Name	FrTrcvChannelDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**FrTrcv453_Conf :**
Name	FRTRCV_E_FR_BUSERROR_TRCV {FRTRCV_E_FR_BUSERROR_TRCV}

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

SWS Item	**FrTrcv452_Conf :**		
Name	FRTRCV_E_FR_ERRN_TRCV {FRTRCV_E_FR_ERRN_TRCV}		
Description	Reference to configured DEM event to report "Error Status of Class A (GPIO) transceiver"		
Multiplicity	0..1		
Type	Reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Dem		

No Included Containers

10.2.6 FrTrcvBranchIdContainer

SWS Item	**FrTrcv357_Conf :**
Container Name	FrTrcvBranchIdContainer
Description	Only one SymbolicNameValue can be defined per container. Therefore this container is necessary.
Configuration Parameters	

SWS Item	**FrTrcv356_Conf :**		
Name	FrTrcvBranchId {FRTRCV_BRANCH_ID}		
Description	Unique branch id. It is used by CDDs and internally.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

No Included Containers

10.2.7 FrTrcvDioAccess

SWS Item	**FrTrcv145_Conf :**
Container Name	FrTrcvDioAccess{FrTransceiverDioAccess}
Description	Container gives FR transceiver driver information about accessing ports and port pins. If a FR transceiver hardware has no Dio interface, there is no instance of this container.
Configuration Parameters	

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

10.2.8 FrTrcvDioChannelAccess

SWS Item	**FrTrcv471_Conf :**
Container Name	FrTrcvDioChannelAccess{FrTrcvDioChannelAccess}
Description	In this Container the relation between FR transceiver hardware pin names and Dio port access information is given.
Configuration Parameters	

SWS Item	**FrTrcv150_Conf :**		
Name	FrTrcvHardwareInterfaceName {FRTRCV_HARDWARE_INTERFACE_NAME}		
Description	FR transceiver hardware interface name. It is typically the name of a pin. From a Dio point of view it is either a port, a single channel or a channel group. Depending on this fact either FRTRCV_DIO_PORT_SYMBOLIC_NAME or FRTRCV_DIO_CHANNEL_SYMBOLIC_NAME or FRTRCV_DIO_CHANNEL_GROUP_SYMBOLIC_NAME shall reference a Dio configuration. The FR transceiver driver implementation description shall list up this name for the appropriate FR transceiver hardware.		
Multiplicity	1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance		

SWS Item	**FrTrcv149_Conf :**
Name	FrTrcvDioSymNameRef
Description	Choice Reference to a DIO Port, DIO Channel or DIO Channel Group. This reference replaces the FRTRCV_DIO_PORT_SYM_NAME, FRTRCV_DIO_CHANNEL_SYM_NAME and FRTRCV_DIO_GROUP_SYM_NAME references in the Fr Trcv SWS.
Multiplicity	1
Type	Choice reference to [DioChannel , DioChannelGroup , DioPort]

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

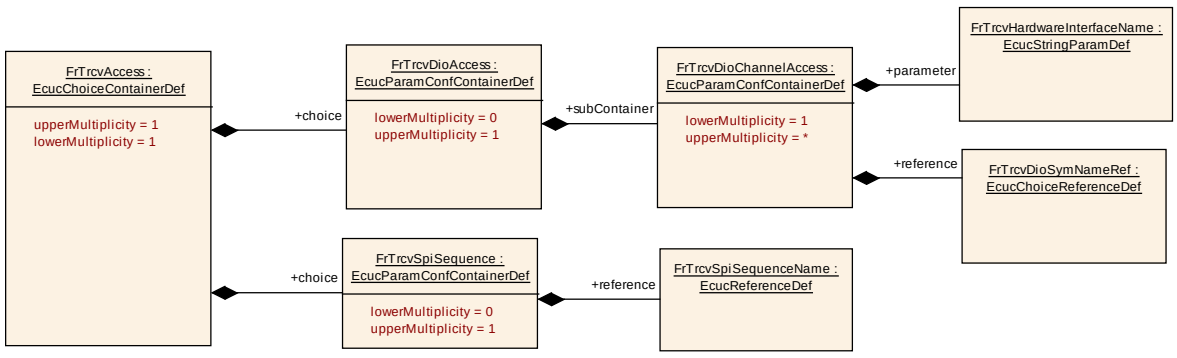
No Included Containers

10.2.9 FrTrcvSpiSequence

SWS Item	**FrTrcv144_Conf :**
Container Name	FrTrcvSpiSequence{FlexRayTransceiverSPISequences}
Description	Container gives FlexRay transceiver driver information about one SPI sequence. One SPI sequence used by FlexRay transceiver driver is in exclusive use for it. No other driver is allowed to access this sequence. FlexRay transceiver driver may use one sequence to access n FlexRay transceiver hardwares chips of the same type or n sequences are used to access one single FlexRay transceiver hardware chip. If a FlexRay transceiver hardware has no SPI interface, there is no instance of this container.
Configuration Parameters	

SWS Item	**FrTrcv151_Conf :**		
Name	FrTrcvSpiSequenceName {FRTRCV_SPI_SEQUENCE_NAME}		
Description	Reference to a Spi sequence configuration container.		
Multiplicity	1		
Type	Reference to [SpiSequence]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Instance dependency: SpiSequence		

No Included Containers



10.3 Published Information

[FrTrcv477] ⌈The standardized common published parameters as required by BSW00402 in the General Requirements on Basic Software Modules 0 shall be

published within the header file of this module and need to be provided in the BSW Module Description. The according module abbreviation can be found in the List of Basic Software Modules [1]. 」( )

Additional module-specific published parameters are listed below if applicable.

11 Changes to Release 3

11.1 Deleted SWS Items

SWS Item	*Rationale*
CanTrcv354_Conf	Sunstituted by
EcuM2561	EcuM2561 The restart list will typically only contain a subset of drivers. But drivers shall appear in the same order as in the combined list of init block I and init block II (see 10.3 Configurable Parameters, ECUM_DRIVER_RESTART_LIST). not applicable, this is out of FlexRay transceiver driver's scope
EcuM2563	EcuM2563 If hardware is put into a sleep mode during SHUTDOWN then this hardware must be restarted by its driver. The restart list will be invoked in state WAKEUP I (see 7.1.5 WAKEUP State). not applicable, this is out of FlexRay transceiver driver's scope
EcuM2569	EcuM2569 Pending events are validated with a call to EcuM_ValidateWakeupEvent. This call must be placed in the driver or the consuming stack on top of the driver (e.g. the handler). The best place to put this depends on hardware and software design. See also 7.8.5 Requirements for Drivers with Wakeup Sources. not applicable, this is out of FlexRay transceiver driver's scope
EcuM2627	EcuM2627 During the initialization, the driver must detect if the CAN transceiver switched on the power supply because of a passive wakeup. The driver shall notify this. In this case, the system designer is responsible to clear all previous wakeup sources and to set the CAN transceiver as the wakeup source by using the EcuM_SetWakeupEvent. This requirement only applies for systems where the CAN transceiver controls the power supply to implement the SLEEP state. not applicable, this is out of FlexRay transceiver driver's scope
EcuM2745	EcuM2745 The restart list will be invoked in state WAKEUP I (see 7.1.5 WAKEUP State). not applicable, this is out of FlexRay transceiver driver's scope
FrTrcv091	BSW00392 Parameters shall have a type Each Parameter shall have a type. Types shall be based on primitive or, complex types defined within AUTOSAR specifications. I.e. they may be combined to structures, arrays etc. Parameters based on a "define" statement shall be put down in a way that the type can be checked by tool.
FrTrcv219	BSW05123 Configuration Modifiable by a Flashing Process is not applicable, this is out of FlexRay transceiver driver's scope.
FrTrcv263	EcuM2485 If the wakeup source is capable of generating faulty events then the driver or the software stack consuming the driver or another appropriate BSW module shall either provide a validation callout for the wakeup event under validation or directly call the wakeup validation service of the ECU State Manager. If validation is not necessary, then this requirement is not applicable for the according wakeup source. not applicable, FlexRay transceiver driver has no such needs
FrTrcv263	EcuM2560 Some drivers may need re-initialization when the ECU is woken up. This is especially true for drivers with wakeup sources. For re-initialization, a restart block is defined. The restart block is part of the WAKEUP state.not applicable, FlexRay transceiver driver has no such needs
FrTrcv263	EcuM2562 Drivers which serve wakeup sources must be re-initialized in the restart block. The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back (see 7.7.4.1 WAKEUP I). FrTrcv263□

FrTrcv263	EcuM2562: Drivers which serve wakeup sources must be re-initialized in the restart block. The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back (see 7.7.4.1 WAKEUP I).
FrTrcv264	COMTYPE021: General Codes BUSTRCV_OK 0x00 There is no bus transceiver error seen by the driver or transceiver does not support the detection of bus errors. BUSTRCV_E_ERROR 0x01 Bus transceiver detected an unclassified error. 0x02-0x1E Reserved values for future usage.
FrTrcv265	COMTYPE022: The Communication System dependent Return codes shall be named as follows: <BUS>TRCV_E_FR_<Error Code Name>. Error Code Name: self explaining name of error return code.
FrTrcv268	Requirement for FrTrcv_Init
FrTrcv269	Requirement for FrTrcv_Init
FrTrcv271	If development error detection of the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is not within the allowed range, the function FrTrcv_Init shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX and return without any further action.
FrTrcv310	The FlexRay Interface shall call the API FrTrcv_CheckWakeupByTransceiver in case a wakeup is detected by Transceiver
FrTrcv311	EcuM2492 Upon occurrence of a wakeup event, the responsible driver shall invoke an indication to notify the ECU State Manager about the wakeup. FrTrcv311
FrTrcv320	The FrTrcv module's environment shall make sure that all SPAL drivers that are used by the FrTrcv module to access the transceiver hardware, are initialized and usable before FrTrcv_Init is called
FrTrcv353	Active Star mode shall be transparent to the transceiver driver.
FrTrcv355	The application controller (host) has to ensure that the BD enters BD_Normal or BD_Receiveonly mode, before the CC enters one of its states where the CC starts to listen to the channel. This is ensured by the FlexRay State Manager,
FrTrcv356	In case the host commands BD_Normal and the BD does not enter BD_Normal, an error is indicated at the BD host interface. In this case the host shall force the CC to step back to a non listening state (i.e. BD_STANDY). The reason for this is that as long as the BD is not in BD_Normal or BD_Receiveonly mode, no information about the status of the channel is available via the signals RxD and RxEN.
FrTrcv357	The environment of the FrTrcv module shall make sure that all necessary BSW drivers (used by the FrTrcv module) have been initialized and are usable before FrTrcv_Init is called.
FrTrcv365	Drivers which serve wakeup sources must be re-initialized in the restart block.
FrTrcv376	The EcuM shall be able to handle the wake up event immediately after requesting the standby or sleep mode.
FrTrcv377	Drivers which serve wakeup sources must be re-initialized in the restart block.
FrTrcv382	Deleted because of global scope
FrTrcv383	The FlexRay Transceiver Driver shall reject state transitions not defined in FrTrcv_TrcvModeType.
FrTrcv399	
FrTrcv400	The driver restart shall re-arm the trigger mechanism of the 'wakeup detected' call-back.
FrTrcv425	MCG shall generate the symbol FRTRCV_E_FR_NO_CONTROL_TRCV<TrcvIdx> where <TrcvIdx> represents transceiver index. The symbol is generated for each transceiver that is managed in the transceiver driver module. Assignment is done in a header file of module Dem

	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
FrTrcv006	equivalent to FtTrcv419
FrTrcv432	
FrTrcv448_Conf	
FrTrcv271	Removed index from init function. -> If development error detection of the module FrTrcv is enabled: if the parameter FrTrcv_TrcvIdx is not within the allowed range, the function FrTrcv_Init shall raise development error FRTRCV_E_FR_INVALID_TRCVIDX and return without any further action.
FrTrcv473	FrTrcv473 If the optional function FrTrcv_ReportErrorStatus() is configured in the global FlexRay Transceiver Driver configuration, if shall be called instead of Dem_ReportErrorStatus().
FrTrcv456	Calling FrTrcv_CheckWakeupByTransceiver by FrTrcv_MainFunction() shall be supported.

11.2 Replaced SWS Items

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

11.3 Changed SWS Items

SWS Item	***Rationale***
FrTrcv231	Asynchronous access is required with SPI interface The FlexRay Transceiver Driver shall use the APIs of underlying drivers (SPI and Dio) synchronously.--
FrTrcv019	BSW00171 Configurability of optional functionality. This chapter defines all interfaces that are required to fulfill an optional functionality of the module.
FrTrcv055_Conf	FrTrcv055 renamed to FrTrcv055_Conf
FrTrcv074	EcuM2482 To support the wakeup and validation protocol, the driver has to fulfill the following requirements: FrTrcv074
FrTrcv112	BSW00414 Parameter of init function The init function in general shall have no parameter
FrTrcv236	EcuM2486 The driver shall provide an explicit service to put the wakeup source to sleep. This service shall put the wakeup source into a energy saving and inert operation mode and re-arm the wakeup notification mechanism. FrTrcv236
FrTrcv262	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.
FrTrcv266	INTEGR092: Each BSW module implementation <ModulePrefix>.c shall include its respective header file SchM_<ModulePrefix>.h.
FrTrcv267	MEMMAP003: Each AUTOSAR software module shall wrap declaration and definition of code, variables and constants using the following mechanism: 1. Definition of start symbol for module memory section

	2. Inclusion of MemMap.h 3. Declaration/definition of code, variables or constants belonging to the specified section 4. Definition of stop symbol for module memory section 5. Inclusion of MemMap.h The inclusion of MemMap.h within the code is a MISRA violation. As neither executable code nor symbols are included (only pragmas) this violation is an approved exception without side effects.
FrTrcv393	added reference to FrTrcv456_Conf
FrTrcv086	added reference to FrTrcv456_Conf
FrTrcv105	modified phrasing
FrTrcv343_Conf	
FrTrcv352_Conf	FrTrcv main function not called by FrIf
FrTrcv085	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv278	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv281	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv284	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv306	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv354	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv390	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error
FrTrcv437	FRTRCV_E_FR_NO_CONTROL_TRCV changed to development error

11.4 Added SWS Items

SWS Item	*Rationale*
FrTrcv002	
FrTrcv055_Conf	
FrTrcv384	-
FrTrcv385	
FrTrcv386	
FrTrcv387	
FrTrcv388	
FrTrcv389	
FrTrcv390	
FrTrcv391	
FrTrcv392	
FrTrcv393	
FrTrcv394	
FrTrcv395	
FrTrcv396	
FrTrcv397	
FrTrcv398	
FrTrcv400	
FrTrcv401	
FrTrcv402	
FrTrcv403	
FrTrcv404	
FrTrcv405	
FrTrcv406	
FrTrcv407	
FrTrcv408	
FrTrcv409	
FrTrcv410	
FrTrcv411	
FrTrcv412	
FrTrcv413	

FrTrcv414	
FrTrcv415	
FrTrcv416	
FrTrcv417	
FrTrcv418	
FrTrcv419	
FrTrcv420	
FrTrcv421	
FrTrcv422	
FrTrcv423	
FrTrcv424	
FrTrcv426	
FrTrcv427	
FrTrcv428	
FrTrcv091_Conf	
FrTrcv429	
FrTrcv430	
FrTrcv431	
FrTrcv433	
FrTrcv434	
FrTrcv435	
FrTrcv436	
FrTrcv437	Requirement for FrTrcv_Init
FrTrcv438	Requirement for FrTrcv_Init
FrTrcv439	
FrTrcv440	
FrTrcv441	
FrTrcv442	Requirement to disable transceiver branches
FrTrcv443	Requirement to enable transceiver branches
FrTrcv444	Requirement to get bus errors per branch
FrTrcv445	Concept 406
FrTrcv446	Concept 406
FrTrcv447	Concept 406
FrTrcv448_Conf	Requirement to support active star transceivers with branches
FrTrcv449	Requirement to provide APIs of active stars for regular transceivers, too
FrTrcv449_Conf	Additional DEM Reference
FrTrcv450	Debug support of branches of active stars
FrTrcv451	
FrTrcv451_Conf	Additional DEM Reference
FrTrcv452	
FrTrcv452_Conf	Additional DEM Reference
FrTrcv453	
FrTrcv454	
FrTrcv455	
FrTrcv456	
FrTrcv457	Refined specification of transceiver faults
FrTrcv458	Refined specification of transceiver faults
FrTrcv459	Additional DET checks
FrTrcv460	Additional DET checks
FrTrcv461	Additional DET checks
FrTrcv462	Additional DET checks
FrTrcv463	Additional DET checks
FrTrcv464	Additional DET checks
FrTrcv465	Additional DET checks
FrTrcv466	Additional DET checks
FrTrcv467	Additional DET checks
FrTrcv468_Conf	Additional DEM Reference
FrTrcv469_Conf	Additional DEM Reference

FrTrcv470_Conf	Additional DEM Reference
FrTrcv471_Conf	Substitutes typo CanTrcv354_Conf
FrTrcv472	Prepassed on no bus fault
FrTrcv473	Support of integration of DEM reports
FrTrcv474	
FrTrcv475	CDD support of error reporting
FrTrcv456_Conf	Figure 2 to be corrected and Configuration Id to be corrected
FrTrcv476	
FrTrcv384_Conf	[diverse] Possible loss of CAN wakeup added FrTrcvIcuChannelRef

12 Changes during SWS Improvements by Technical Office

12.1 Deleted SWS Items

SWS Item	*Rationale*
FrTrcv065	The return-type of a function is part of the table generated from UML
FrTrcv066	The return-type of a function is part of the table generated from UML
FrTrcv067	The parameter-types of a function are part of the table generated from UML
FrTrcv084	Description of the function EcuM_SetWakeupEvent is not a requirement
FrTrcv009	Redundant with FrTrcv033, FrTrcv080, FrTrcv057, FrTrcv079, FrTrcv117
FrTrcv309	Requirement for FrTrcv_MainFunction

12.2 Replaced SWS Items

SWS Item of Release 1	*replaced by SWS Item*	*Rationale*
FrTrcv096	FrTrcv317, FrTrcv318, FrTrcv319	Made requirement atomic

12.3 Changed SWS Items

Many requirements have been changed to improve understandability without changing the technical contents.

12.4 Added SWS Items

SWS Item	*Rationale*
FrTrcv268	Requirement for FrTrcv_Init
FrTrcv269	Requirement for FrTrcv_Init
FrTrcv270	Requirement for FrTrcv_Init
FrTrcv271	Requirement for FrTrcv_Init
FrTrcv272	Requirement for FrTrcv_SetTransceiverMode
FrTrcv273	Requirement for FrTrcv_SetTransceiverMode
FrTrcv274	Requirement for FrTrcv_SetTransceiverMode
FrTrcv275	Requirement for FrTrcv_SetTransceiverMode
FrTrcv276	Requirement for FrTrcv_SetTransceiverMode
FrTrcv277	Requirement for FrTrcv_SetTransceiverMode
FrTrcv278	Requirement for FrTrcv_SetTransceiverMode
FrTrcv279	Requirement for FrTrcv_GetTransceiverMode
FrTrcv280	Requirement for FrTrcv_GetTransceiverMode
FrTrcv281	Requirement for FrTrcv_GetTransceiverMode
FrTrcv282	Requirement for FrTrcv_GetTransceiverWUReason
FrTrcv283	Requirement for FrTrcv_GetTransceiverWUReason
FrTrcv284	Requirement for FrTrcv_GetTransceiverWUReason
FrTrcv285	Requirement for FrTrcv_GetVersionInfo
FrTrcv286	Requirement for FrTrcv_DisableTransceiverWakeup
FrTrcv287	Requirement for FrTrcv_DisableTransceiverWakeup
FrTrcv288	Requirement for FrTrcv_DisableTransceiverWakeup

FrTrcv289	Requirement for FrTrcv_DisableTransceiverWakeup
FrTrcv290	Requirement for FrTrcv_DisableTransceiverWakeup
FrTrcv300	Requirement for FrTrcv_EnableTransceiverWakeup
FrTrcv301	Requirement for FrTrcv_EnableTransceiverWakeup
FrTrcv302	Requirement for FrTrcv_EnableTransceiverWakeup
FrTrcv303	Requirement for FrTrcv_EnableTransceiverWakeup
FrTrcv304	Requirement for FrTrcv_ClearTransceiverWakeup
FrTrcv305	Requirement for FrTrcv_ClearTransceiverWakeup
FrTrcv306	Requirement for FrTrcv_ClearTransceiverWakeup
FrTrcv308	Requirement for FrTrcv_MainFunction
FrTrcv310	Requirement for FrTrcv_CheckWakeupByTransceiver
FrTrcv311	Requirement for FrTrcv_CheckWakeupByTransceiver
FrTrcv312	Requirement for FrTrcv_CheckWakeupByTransceiver
FrTrcv313	Requirement for FrTrcv_CheckWakeupByTransceiver
FrTrcv314	Each variant gets an individual requirement ID
FrTrcv315	Each variant gets an individual requirement ID
FrTrcv316	Each variant gets an individual requirement ID
FrTrcv291	Requirement for FrTrcv module
FrTrcv295	Requirement for FrTrcv module
FrTrcv296	Requirement for FrTrcv module
FrTrcv317	Requirement for FrTrcv module
FrTrcv318	Requirement for FrTrcv module
FrTrcv319	Requirement for FrTrcv module
FrTrcv320	Requirement for FrTrcv module's environment
FrTrcv321	UML Model linking of imported types
FrTrcv322	UML Model linking of FrTrcv_Init
FrTrcv323	UML Model linking of FrTrcv_SetTransceiverMode
FrTrcv324	UML Model linking of FrTrcv_GetTransceiverMode
FrTrcv325	UML Model linking of FrTrcv_GetTransceiverWUReason
FrTrcv326	UML Model linking of FrTrcv_GetVersionInfo
FrTrcv327	UML Model linking of FrTrcv_DisableTransceiverWakeup
FrTrcv328	UML Model linking of FrTrcv_EnableTransceiverWakeup
FrTrcv329	UML Model linking of FrTrcv_ClearTransceiverWakeup
FrTrcv330	UML Model linking of FrTrcv_MainFunction
FrTrcv331	UML Model linking of FrTrcv_CheckWakeupByTransceiver
FrTrcv332	UML Model linking of mandatory interfaces
FrTrcv334	UML Model linking of optional interfaces
FrTrcv335	Requirement for FrTrcv module
FrTrcv338	Standard requirement for GetVersionInfo
FrTrcv339	Standard requirement for GetVersionInfo
FrTrcv340	Gave ID to existing requirement
FrTrcv352	Gave ID to existing requirement
FrTrcv353	Gave ID to existing requirement
FrTrcv354	Gave ID to existing requirement
FrTrcv355	Gave ID to existing requirement
FrTrcv356	Gave ID to existing requirement
FrTrcv357	Gave ID to existing requirement
FrTrcv358	Gave ID to existing requirement
FrTrcv359	Gave ID to existing requirement
FrTrcv360	Gave ID to existing requirement
FrTrcv361	Gave ID to existing requirement
FrTrcv362	Gave ID to existing requirement
FrTrcv363	Gave ID to existing requirement
FrTrcv364	Gave ID to existing requirement
FrTrcv365	Gave ID to existing requirement
FrTrcv366	Gave ID to existing requirement
FrTrcv367	Gave ID to existing requirement
FrTrcv368	Gave ID to existing requirement

FrTrcv369	Gave ID to existing requirement
FrTrcv370	Gave ID to existing requirement
FrTrcv371	Gave ID to existing requirement
FrTrcv372	Gave ID to existing requirement
FrTrcv373	Gave ID to existing requirement
FrTrcv374	Gave ID to existing requirement
FrTrcv375	Gave ID to existing requirement
FrTrcv376	Gave ID to existing requirement
FrTrcv377	Gave ID to existing requirement
FrTrcv378	Gave ID to existing requirement
FrTrcv379	Gave ID to existing requirement
FrTrcv380	Gave ID to existing requirement
FrTrcv381	Gave ID to existing requirement
FrTrcv382	Gave ID to existing requirement
FrTrcv383	Gave ID to existing requirement
FrTrcv477	Rework of Published Information

13 Not applicable requirements

[FrTrcv478] ⌈These requirements are not applicable to this specification.⌋
(BSW00005, BSW00006, BSW00007, BSW00009, BSW00010, BSW00161, BSW00164, BSW00168, BSW00304, BSW00306, BSW00307, BSW00308, BSW00309, BSW00312, BSW00321, BSW00325, BSW00326, BSW00328, BSW00330, BSW00333, BSW00336, BSW00341, BSW00342, BSW00344, BSW00355, BSW00359, BSW00360, BSW00370, BSW00378, BSW00382, BSW00383, BSW00384, BSW00386, BSW00392, BSW00398, BSW00399, BSW00400, BSW00401, BSW00404, BSW00405, BSW00410, BSW00413, BSW00416, BSW00417, BSW00420, BSW00422, BSW00423, BSW00426, BSW00427, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW05000, BSW05001, BSW05002, BSW05003, BSW05004, BSW05005, BSW05006, BSW05007, BSW05009, BSW05010, BSW05011, BSW05012, BSW05013, BSW05015, BSW05016, BSW05018, BSW05019, BSW05022, BSW05023, BSW05024, BSW05025, BSW05027, BSW05031, BSW05033, BSW05034, BSW05035, BSW05038, BSW05039, BSW05040, BSW05041, BSW05042, BSW05044, BSW05045, BSW05046, BSW05047, BSW05048, BSW05049, BSW05050, BSW05051, BSW05052, BSW05053, BSW05055, BSW05056, BSW05058, BSW05059, BSW05060, BSW05063, BSW05064, BSW05065, BSW05066, BSW05067, BSW05068, BSW05069, BSW05072, BSW05073, BSW05074, BSW05075, BSW05076, BSW05077, BSW05078, BSW05079, BSW05082, BSW05083, BSW05084, BSW05085, BSW05088, BSW05089, BSW05090, BSW05093, BSW05095, BSW05096, BSW05097, BSW05101, BSW05102, BSW05104, BSW05106, BSW05107, BSW05109, BSW05111, BSW05113, BSW05114, BSW05115, BSW05116, BSW05117, BSW05120, BSW05121, BSW05123, BSW05124, BSW05125, BSW05126, BSW05129, BSW05130, BSW05153, BSW05154, BSW05155, BSW05156, BSW05157, BSW05158, BSW05162, BSW05163, BSW05164, BSW05165, BSW05169, BSW05170, BSW05171, BSW05172, BSW05173, BSW05174, BSW05175, BSW05200, BSW05201, BSW05202, BSW05204, BSW05205, BSW05206, BSW05207, BSW05208, BSW05209, BSW05210, BSW05211, BSW05215)