Document Title	Requirements on FlexRay
Document Category	Requirement specification (RS)
Document UID	5
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History

Date	Release	Changed by	Description
2026-11-26	R26-11	AUTOSAR Release Management	• No content changes
2025-11-27	R25-11	AUTOSAR Release Management	• No content changes
2024-11-27	R24-11	AUTOSAR Release Management	• No content changes
2023-11-23	R23-11	AUTOSAR Release Management	• No content changes
2022-11-24	R22-11	AUTOSAR Release Management	• No content changes
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• No content changes
2019-11-28	R19-11	AUTOSAR Release Management	• Modification of initialization requirements • Changed Document Status from Final to published
2018-10-31	4.4.0	AUTOSAR Release Management	• Editorial changes
2017-12-08	4.3.1	AUTOSAR Release Management	• Editorial changes

▽

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Minor corrections
2015-07-31	4.2.2	AUTOSAR Release Management	• Editorial changes
2014-10-31	4.2.1	AUTOSAR Release Management	• Editorial changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Removed non-implementable runtime checks • Editorial changes
2013-10-31	4.1.2	AUTOSAR Release Management	• Correction in E2E variant 1C • Various minor corrections • Editorial changes
2013-03-15	4.1.1	AUTOSAR Administration	• Added support for ISO TP and AUTOSAR TP • Extended Transport connection properties ISO 15762-2 and ISO 10681-2 • Added requirements traceability • Editorial changes
2011-12-22	4.0.3	AUTOSAR Administration	• Added "Wake-pin" as wake-up reason • Update of ISO 15765-2 and ISO 15765-4 support
2010-09-30	3.1.5	AUTOSAR Administration	• Added support for FlexRay 3.0 • Added functionalities to get detailed (error) information of the communication bus • Bus Guardian support removed • Support for cluster related APIs removed • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2007-12-21	3.0.1	AUTOSAR Administration	• Document meta information extended • Small layout adaptations made
2007-01-24	2.1.15	AUTOSAR Administration	• "Advice for users" revised • "Revision Information" added

▽

△

2006-11-28	2.1	AUTOSAR Administration	• Legal disclaimer revised

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Scope of Document 7

2 Conventions to be used 8
2.1 Document Conventions . . . 8
2.2 Requirements structure . . . 9

3 Acronyms and abbreviations 11

4 Requirements Specification 12
4.1 Functional Overview . . . 12
4.1.1 Common Requirements (FlexRay Interface and FlexRay Driver) . . . 12
4.1.2 FlexRay Interface . . . 13
4.1.3 FlexRay Driver . . . 13
4.1.4 Transport Layer FlexRay . . . 13
4.1.5 FlexRay Time Service . . . 13
4.1.6 FlexRay Transceiver Driver . . . 13
4.1.6.1 Transceiver Operation Modes . . . 14
4.1.6.2 Transceiver Error Status . . . 14
4.1.6.3 Transceiver Wake-up Reason . . . 14
4.1.6.4 Remarks to the FlexRay Transceiver Driver . . . 15
4.1.6.5 Explicitly uncovered FlexRay Transceiver Functionality . . . 15
4.1.7 System Basis Chip and FlexRay Transceiver Driver . . . 15
4.2 Functional Requirements . . . 16
4.2.1 General requirements . . . 16
4.2.1.1 FlexRay Interface . . . 24
4.2.1.2 FlexRay Driver . . . 39
4.2.1.3 Transport Layer FlexRay . . . 51
4.2.1.4 FlexRay Time Service . . . 54
4.2.1.5 FlexRay Transceiver Driver . . . 59
4.3 Non-Functional Requirements (Qualities) . . . 67
4.3.1 FlexRay Interface . . . 67
4.3.1.1 Abstraction Requirements . . . 67
4.3.1.2 Resource Usage . . . 67
4.3.1.3 Configuration . . . 68
4.3.2 Transport Layer FlexRay . . . 68
4.3.3 FlexRay Transceiver Driver . . . 70
4.3.3.1 Timing Requirements . . . 71

5 References 72

A Change history of AUTOSAR traceable items 73
A.1 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 73
A.1.1 Added Traceables in R22-11 . . . 73

A.1.2 Changed Traceables in R22-11 77
A.1.3 Deleted Traceables in R22-11 77
A.2 Traceable item history of this document according to AUTOSAR Release R23-11 78
A.2.1 Added Requirements in R23-11 78
A.2.2 Changed Requirements in R23-11 82
A.2.3 Deleted Requirements in R23-11 82
A.3 Traceable item history of this document according to AUTOSAR Release R24-11 83
A.3.1 Added Requirements in R24-11 83
A.3.2 Changed Requirements in R24-11 83
A.3.3 Deleted Requirements in R24-11 83
A.4 Traceable item history of this document according to AUTOSAR Release R25-11 83
A.4.1 Added Requirements in R25-11 83
A.4.2 Changed Requirements in R25-11 83
A.4.3 Deleted Requirements in R25-11 83
A.5 Traceable item history of this document according to AUTOSAR Release R26-11 84
A.5.1 Added Requirements in R26-11 84
A.5.2 Changed Requirements in R26-11 84
A.5.3 Deleted Requirements in R26-11 84

1 Scope of Document

The goal of this document is to define to what extent elements of the basic software have to be configurable and what preliminaries they have to comply with to meet the tailoring requirements.

As far as possible, the set of basic software elements should consist of already existing elements of modules of automotive software. Only in case of "good reasons" new elements of basic software should be part of the set.

If such the definition of these new elements is not part of this work package. Nevertheless the information about basic software elements additionally required has to be given to related work groups.

Constraints and restrictions

The first scope for specification of requirements on basic software modules is systems which are not safety relevant. For this reason safety requirements are assigned to medium priority.

Data transmission accesses to the FlexRay Interface while FlexRay Communication Controllers are out of sync or in shutdown state shall not return an error.

Redundancy (in the time or spatial domain) is not supported by the FlexRay Interface; redundancy has to be explicitly modeled on a higher BSW layer.

There are functional restrictions imposed by the FlexRay Protocol Specification [FR_PROT_SPEC] concerning Wake-Up and Start-up. In both cases, it is not guaranteed by the FlexRay Protocol Specification that the intended result will be reached, i.e. in case of:

- Start-up: It is not guaranteed that the start-up process will be successful, i.e. after initiating a start-up, it is not guaranteed that the FlexRay Communication Controller will successfully establish a communication on the Cluster, or integrate into an ongoing communication, resp.
- Wake-up: It is not guaranteed that the wake-up process will be successful, i.e. after the FlexRay Communication Controller has sent a wake-up pattern on a specific FlexRay Channel, it is not guaranteed that all other FlexRay Communication Controllers connected to this Channel actually do wake up.

A higher AUTOSAR BSW module has to provide a supervision mechanism to ensure that the intended result of these operations will be reached. This mechanism is supported by features requested by the FlexRay Software Requirements Specification documents in hand.

2 Conventions to be used

2.1 Document Conventions

The representation of requirements in AUTOSAR documents follows the table specified in [TPS_STDT_00078], see [1, Standardization Template].

The verbal forms for the expression of obligation specified in [TPS_STDT_00053] shall be used to indicate requirements, see [1, Standardization Template].

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as follows.

Note that the requirement level of the document in which they are used modifies the force of these words.

- MUST: This word, or the adjective “LEGALLY REQUIRED”, means that the definition is an absolute requirement of the specification due to legal issues.
- MUST NOT: This phrase, or the phrase “MUST NOT”, means that the definition is an absolute prohibition of the specification due to legal issues.
- SHALL: This phrase, or the adjective “REQUIRED”, means that the definition is an absolute requirement of the specification.
- SHALL NOT: This phrase means that the definition is an absolute prohibition of the specification.
- SHOULD: This word, or the adjective “RECOMMENDED”, means that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
- SHOULD NOT: This phrase, or the phrase “NOT RECOMMENDED”, means that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
- MAY: This word, or the adjective “OPTIONAL”, means that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item.

An implementation, which does not include a particular option, SHALL be prepared to interoperate with another implementation, which does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option, SHALL be prepared to interoperate with another implementation, which does not include the option (except, of course, for the feature the option provides).

2.2 Requirements structure

Each module-specific chapter contains a short functional description of the Basic Software Module. Requirements of the same kind within each chapter are grouped under the following headlines (where applicable):

Functional Requirements:

- Configuration (Which elements of the module need to be configurable?)
- Initialization
- Normal Operation
- Shutdown Operation
- Fault Operation
- ...

Non-Functional Requirements:

- Timing Requirements
- Resource Usage
- Usability
- Dependencies on other SW modules (e.g. Description Templates, Tooling, ...)
- ...

Not each requirement in this SRS does necessarily map to exactly one API call of the respective AUTOSAR FlexRay SW module. It might be possible that one API call fulfills multiple requirements.

It might also be possible that some requirements address internal features of the AUTOSAR FlexRay SW modules and do not require an API call at all in order to be fulfilled.

Implementations of the AUTOSAR FlexRay SW modules used in safety-critical environments might only be allowed to fulfill a subset of the requirements described herein.

Apart from this safety-critical case, every requirement listed in this document is mandatory, i.e. each implementation of an AUTOSAR FlexRay SW module MUST fulfill all respective requirements in this document so if that SW module is being used with a hardware supporting the full feature set, the full functionality of the AUTOSAR FlexRay SW module is available.

However, if the underlying hardware does NOT support a specific feature, the AUTOSAR FlexRay SW modules do NOT have to emulate this functionality if that requirement is explicitly marked as "if supported by the FlexRay CC".

If a higher layer BSW module calls an API of the AUTOSAR FlexRay SW modules that operates on an optional feature of the FlexRay Specification, and this feature is not supported in the system in question, the respective FlexRay SW modules shall raise an error in debug mode.

In general, the FlexRay Interface has no knowledge of the origin of a PDU passed to it in an API call.

Therefore, throughout this document, unless stated otherwise, the term "PDU" is being used for PDUs originating or sent to:

- AUTOSAR COM (I-PDU), or
- FlexRay TP (N-PDU), or
- FlexRay NM

Throughout this document the term "configuration" is being used in several requirements. Unless otherwise noted, this term refers to the configuration of the respective software module that affects the runtime behavior but not the generated executable code of this software module. The most prominent case of such a configuration is the FlexRay communication schedule. Enabling the development error handling in the executable code is explicitly not meant.

Accordingly, the term "configuration data" in general refers to pure data (representing a certain configuration) and not to any executable code.

Throughout this document, several scenarios for changing configuration data are mentioned. They are being used as follows:

- **"pre compile time"** = carried out before compiling the code of the FlexRay Interface/Driver, since the code generation might or will depend on this setting.
- **"at system configuration time"** = static configuration parameters stored in the FlexRay Interface/Driver; may be defined after compilation of the code of the FlexRay Interface/Driver, but have to be defined before the first execution of the FlexRay Interface/Driver code.
- **"by a flashing process"** = data (not code!) manipulation carried out in a flashing process of a flashable memory (in general a Flash-EEPROM) e.g. in a garage, but not while the car is being driven. Usually used to replace a static configuration already stored in the ECU, or a part thereof.
- **"during runtime"** = dynamically switching (in normal active mode of the FlexRay CC, if supported by the FlexRay CC) between different configuration parameter sets statically stored in the FlexRay Interface.

3 Acronyms and abbreviations

The glossary below includes acronyms and abbreviations relevant to SRS FlexRay that are not included in the AUTOSAR Glossary [2, AUTOSAR_TR_Glossary].

Acronym:	Description:
SW	Software
BSW	Basic Software
DEM	Diagnostic Event Manager BSW module
ComM	CommunicationManager BSW module
CC	(FlexRay) Communication Controller

Table 3.1: Acronyms used in the scope of this Document

Abbreviation:	Description:
i.e.	[lat.] id est = [engl.] that is
e.g.	[lat.] exempli gratia = [engl.] for example

Table 3.2: Abbreviations used in the scope of this Document

4 Requirements Specification

This chapter describes all requirements driving the work to define the FlexRay.

4.1 Functional Overview

4.1.1 Common Requirements (FlexRay Interface and FlexRay Driver)

- Support of the FlexRay Protocol 2.1 specification
- Support of the FlexRay Protocol 3.0 specification, for details see chapter 6.1
- Abstraction of FlexRay specific features
- Abstraction of FlexRay Controller type specific features
- Support of at least 4 (possibly different) FlexRay Controllers per ECU with up to 2 Channels each and data rates up to 10 MBit/s
- Connect the communication interface to the specific FlexRay Controller
- Transmit and receive FlexRay Frames in the static and in the dynamic segment
- Support of multiple FlexRay Clusters
- Support of (with respect to the FlexRay global time) asynchronous and synchronous operation of the communication stack
- Packing and unpacking of multiple PDU into/out of one or more FlexRay Frames
- Interface: PDU-based API to upper software layer
- Single/multiple sender Cycle Multiplexing for scalable efficient bandwidth exploitation
- Configuration of the complete FlexRay system at system configuration time
- Run time reconfiguration of FlexRay Controllers
- Provision of basic network management functions: Transmit/detect wake-up patterns; start-up, shutdown, sleep mode, start and stop communication
- Provision of the FlexRay Controller state e.g. for usage by FlexRay NM
- Provision of FlexRay bus errors

For an overview of the structure of AUTOSAR software modules please see the current AUTOSAR Layered Software Architecture Specification [LSA].

4.1.2 FlexRay Interface

The FlexRay Interface shall provide a standardized interface to access the FlexRay Communication system/hardware. The FlexRay Interface shall be independent of the specific FlexRay CC(s) being used and its/their access through the FlexRay Driver(s). The FlexRay Interface shall give access to one or several FlexRay Drivers via one uniform interface.

4.1.3 FlexRay Driver

The FlexRay Driver module shall offer to the FlexRay Interface Module, the user of the API, a uniform interface to access a number of FlexRay Communication Controllers, usually of the same type. The FlexRay Driver is a software layer which maps abstract functional requests to sequences of CC specific hardware accesses. The hardware implementation of a CC will be hidden from the FlexRay Interface; e. g. the number of transmit/receive buffers will not be visible outside a FlexRay Driver.

4.1.4 Transport Layer FlexRay

The FlexRay Transport Layer supports the protocol handling according ISO 10681-2.

Hence the FlexRay Transport Layer provides support for

- unsegmented unacknowledged 1:1 communication
- segmented unacknowledged 1:1 communication
- unsegmented acknowledged 1:1 communication
- segmented acknowledged 1:1 communication
- unsegmented unacknowledged 1:n communication

4.1.5 FlexRay Time Service

Provides FlexRay time services, synchronization to FlexRay global time, interrupt services based on FlexRay global time, and synchronization of FlexRay Clusters to external time references

4.1.6 FlexRay Transceiver Driver

The FlexRay Transceiver Driver is responsible to handle the FlexRay transceivers on an ECU according to the expected state of the bus specific NM.

The FlexRay Transceiver is a hardware device, which mainly transforms the logical 1/0 signals of the FlexRay Communication Controller's ports to the bus compliant electrical

levels, currents and timings. Within an automotive environment, there is currently only one physical layer specified for FlexRay.

In addition, the transceivers are often able to detect electrical malfunctions like wiring issues, ground offsets or collisions. Depending on the interface, they are capable of flagging the detected errors by a single port pin or very detailed via SPI.

Some transceivers also support power supply control and wake-up via the bus. A lot of different wake-up/sleep and power supply concepts are available on the market with focus on the best cost-optimized solution for a given task.

Latest developments are so called System Basis Chips (SBC) where not only the FlexRay transceivers but also power-supply control and advanced watchdogs are implemented in one housing and are controlled via one interface (e.g. via SPI).

A typical FlexRay Transceiver device is the TJA1080 device for a 10 MBit FlexRay bus with support for wake-up via the bus.

4.1.6.1 Transceiver Operation Modes

Important transceiver operation modes are:

- Un-powered: Neither communication nor wake-up is possible
- Normal: Full communication is possible
- Read Only: Transmission is inhibited, reception is possible
- Standby: No communication is possible but ECU is still powered. In addition, a transition to Sleep is only valid from this mode. Wake-up by bus is possible, too.
- Sleep: No communication is possible and ECU may be un-powered. Wake-up by bus is possible, too.

4.1.6.2 Transceiver Error Status

In parallel to the transceiver operation modes, the transceiver provides status information for the bus. The status is only relevant for operation mode Normal:

- Good: No error, bus physics work correctly without any drawbacks.
- Error: The transceiver has detected a serious error on the bus which does not allow further communication.

4.1.6.3 Transceiver Wake-up Reason

The transceiver driver is able to store the local view on who has requested the wake-up:

- Remote wake-up event: A remote wake-up event is the reception of at least two consecutive wake-up symbols via the bus.
- Local wake-up event: A pulse on the FlexRay Transceiver device's WAKE pin (if present) has caused the wake-up.
- Power on wake-up: The FlexRay Transceiver device has become sufficiently supplied after being not powered.
- No wake-up: No wake-up has been detected.

4.1.6.4 Remarks to the FlexRay Transceiver Driver

FlexRay Transceivers are very different in their behavior and supported features. The range starts with very simple FlexRay Transceivers, which are "always on", includes transceivers with support for advanced limp-home-handling and error detection and ends with so called system basis chips (SBC) which internally contain multiple FlexRay Transceivers, watchdog, voltage regulators and more.

The target of this document is to specify interfaces and behavior, which are applicable to most current and future FlexRay Transceivers on the market for nearly all use cases. The target is that the "user" of the transceiver functionality, typically the AUTOSAR CommunicationManager (ComM) and the AUTOSAR ECU State Manager (EcuM), are bus independent and therefore reusable.

It will not be possible to cover all possible combinations of FlexRay Transceivers with all conceivable power concepts within one AUTOSAR implementation.

4.1.6.5 Explicitly uncovered FlexRay Transceiver Functionality

Some FlexRay Transceivers offer additional functionality to improve e.g. ECU self-test or enhanced error detection capability for diagnostics.

ECU self-test and enhanced error detection are not defined within AUTOSAR and requiring such functionality in general will lock out most currently used (and cheap) transceiver devices. Therefore, features like "ground shift detection", "selective wake-up", "slope control" and others are not supported within these requirements. A general and "open" API like IOControl() is not applicable (and accepted) within AUTOSAR due to portability and reuse.

4.1.7 System Basis Chip and FlexRay Transceiver Driver

A system basis chip (SBC) contains, beside the FlexRay Transceivers, additional hardware related to power control and safety (e.g. multiple voltage regulators and a watchdog) and even more features (e.g. persistent memory).

In the AUTOSAR concept, a separate manager/driver/handler (in AUTOSAR called: Interface) is responsible for each identified hardware device. Therefore, the additional manager/driver/handler covers the functionality inside a SBC beside the transceiver driver (e.g. Watchdog Manager, non-volatile memory manager, power control driver, etc.). Due to the shared communication access and the (security-related) restrictions within this communication, independent handling of each SBC-sub-functionality will not be possible.

This will lead to the situation that either a SBC could not be used within an AUTOSAR compliant ECU or (the better solution) a specialized manager/driver/handler for the SBC functionality with all APIs of each single domain has to be used.

4.2 Functional Requirements

Compatibility overview on supported FlexRay CC features

Feature	FlexRay 2.1	FlexRay 3.0
NM Vector update	supported	added: early update
WUDOP in SymbolWindow	N/A	not supported
POC: deferred ready	N/A	not supported
POC: deferred halt	N/A	supported
FIFO support	supported	supported
2nd absolute timer	N/A	supported
Message ID filtering	supported	supported
TT-E, TT-M	N/A	supported
repetition cycle filtering values	1,2,4,8,16,32,64	1,2,4,5,8,10,16,20,32,40,50,64
Static slot multiplexing	N/A	supported
Baud rate	10	2.5, 5, 10
number of valid start-up frames	N/A	supported

4.2.1 General requirements

[SRS_Fr_05000] Synchronous SW Modules shall be supported ⌈

Description:	The FlexRay-related BSW modules of the AUTOSAR communication stack shall support applications (or software modules) running synchronously to the FlexRay global time.
Rationale:	An application (or software module) running synchronously to the FlexRay global time (i.e. it is driven by the FlexRay global time) shall be able to efficiently communicate via FlexRay, since an application (or software module) running synchronously to the FlexRay global time allows an optimal adaptation of the data processing schedule to the schedule of the data transmission via FlexRay. The sychronisation point of a sw module with FR can be set on each Macrotick within the flexraycycle

▽

△

Use Case:	For some applications (e.g. chassis function as regulator) it is advantageous to run the application synchronously to sensors and actuators connected to the FlexRay bus, hence to the FlexRay global time. It shall be possible to start an SW Module at a specific Flexray time within the Flexray cycle
Dependencies:	[SRS_Fr_05001] Support of Asynchronous SW Modules
Supporting Material:	–

⌋

[SRS_Fr_05001] Asynchronous SW Modules shall be supported ⌈

Description:	The FlexRay-related BSW modules of the AUTOSAR communication stack shall support applications (or software modules) running asynchronously to the FlexRay global time.
Rationale:	An application (or software module) running asynchronously to the FlexRay global time (i.e. it has no "knowledge" about the FlexRay global time) shall be able to communicate via FlexRay, since an application (or software module) running asynchronously to the FlexRay global time allows a flexible software design.
Use Case:	For some applications (e.g. body or gateway) it is advantageous to run the application event-triggered, hence asynchronously to the FlexRay global time.
Dependencies:	[SRS_Fr_05000] Support of Synchronous SW Modules
Supporting Material:	–

⌋

[SRS_Fr_05002] FlexRay Interface and FlexRay Driver shall operated synchronized to the global time ⌈

Description:	The FlexRay-related BSW modules of the AUTOSAR communication stack shall encapsulate all synchronous services to enable a completely asynchronous usage of the FlexRay-related BSW modules by all upper-layer SW modules. Therefore, the FlexRay Interface shall operate synchronized to the FlexRay global time(s) of all involved FlexRay Communication Controllers by using services of the respective FlexRay Driver.

▽

△

Rationale:	The FlexRay Interface and the FlexRay Driver shall be synchronous to the FlexRay bus to support synchronous communication and to avoid buffering and queuing. Also, each upper-layer SW module shall have the possibility to run asynchronously to the FlexRay global time. If [SRS_Fr_05001] applies to all other SW modules (i.e. all upper-layer software modules run completely asynchronously to any FlexRay global time), the FlexRay Interface and the FlexRay Driver(s) it controls are the only synchronous SW modules. Note that (for example in case of asynchronous gateways) the global time of multiple FlexRay Controllers in a single ECU might be different, since they are connected to different FlexRay Clusters. In this case, the respective operations for each single Communication Controller are to be performed synchronously to this Communication Controller's global time.
Use Case:	An ECU with all SW modules running asynchronously to the FlexRay global time.
Dependencies:	[SRS_Fr_05000], [SRS_Fr_05001]
Supporting Material:	–

⌋

[SRS_Fr_05003] Slot/Cycle Multiplexing shall be supported ⌈

Description:	The FlexRay-related BSW modules of the AUTOSAR communication stack shall support the FlexRay Slot/Cycle Multiplexing mechanism which is used to use a FlexRay Slot on a Channel more efficiently by alternating the Frames being sent in this Slot from Cycle to Cycle. In the static segment, the FlexRay Slot/Cycle Multiplexing mechanism allows an ECU to transmit different Frame contents in the same Slot in different Cycles. This is called "Single Sender Slot Multiplexing". In the dynamic segment, the FlexRay Cycle Multiplexing mechanism allows several ECUs to share the same Slot by uniquely assigning certain Cycles (to be accurate: FlexRay Cells of the same Slot) to each ECU. This is called "Multiple Sender Slot Multiplexing". Both kinds of Slot/Cycle Multiplexing mechanisms shall be supported by the FlexRay-related BSW modules.
Rationale:	Optimal uses of the FlexRay bandwidth can only be reached with an extensive and effective use of the Cycle Multiplexing mechanism.
Use Case:	In the following example, an ECU sends two different Frame contents in the same Slot and thus effectively uses this resource.
Dependencies:	–
Supporting Material:	FlexRay Protocol Specification[3]

⌋

[SRS_Fr_05169] Timer Interrupts during Start-up shall be avoided ⌈

Description:	The FlexRay-related BSW modules of the AUTOSAR communication stack shall ensure that no timer interrupt raised from a FlexRay Communication Controller will be forwarded to other software modules during the start-up phase of the ECU.
Rationale:	Interrupts occurring before the entire ECU software has been initialized may cause problems.
Use Case:	–
Dependencies:	[SRS_Fr_05055] Avoid Timer Interrupts during Shutdown
Supporting Material:	–

⌋

[SRS_Fr_05055] Timer Interrupts during Shutdown shall be avoided ⌈

Description:	The FlexRay-related BSW modules of the AUTOSAR communication stack shall ensure that no timer interrupt raised from a FlexRay Communication Controller will be forwarded to other software modules during the shutdown phase of the ECU.
Rationale:	Interrupts occurring while the ECU software is being shut down may cause problems.
Use Case:	–
Dependencies:	[SRS_Fr_05169] Avoid Timer Interrupts during Start-up
Supporting Material:	–

⌋

[SRS_Fr_05216] An API to query the FlexRay Rate and Offset Correction shall be created ⌈

Description:	1. It shall be possible to query rate and offset correction of a specific FlexRay Communication Controller calculated by the clock synchronization algorithm 2. It shall be possible to send rate and offset correction on the bus
Rationale:	It could be useful to know the current drift rate and the offset of every FlexRay controller. The valid feature is needed to fulfill the safety requirement OSR107 AUTOSAR shall provide ECU hardware monitoring and testing to detect faults in the following components: • CPU • memory • peripheral devices • communication components • address, data and control buses

▽

△

Use Case:	The monitoring of 'drift' of dedicated FlexRay controllers could shorten the error analysis. The monitoring of 'drift' of dedicated FlexRay controllers could allow the prediction of hardware errors.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05217] An API to report FlexRay Status Data for number of Sync Frames shall be created ⌈

Description:	It shall be possible to report FlexRay Status Data for number of Sync Frames by an API
Rationale:	Status data contains number of sync frames received or transmitted for each channel.
Use Case:	Necessary for Diagnostic purposes to detect persistent asymmetry between applied clock correction terms of different nodes. Useful to set DTC. Allows to service the nodes that are missing.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05218] An API to report FlexRay Status Data for list of Sync Frame IDs shall be created ⌈

Description:	It shall be possible to report FlexRay Status Data for list of Sync Frame IDs by an API
Rationale:	Status data contains list of Sync Frame IDs received or transmitted for each channel.
Use Case:	Necessary for Diagnostic purposes to detect persistent asymmetry between applied clock correction terms of different nodes. Useful to set DTC. Allows to service the nodes that are missing.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05219] FlexRay Key Slot Mode shall be supported ⌈

Description:	FlexRay includes a "Key Slot Mode" mode that can be configured for use as an extension of the startup process. During integration all FlexRay nodes only transmit one frame in the static segment (identified as the "key slot"). They do this to minimize the number of frames they might corrupt if their timing is incorrect and they transmit in the slots of other nodes. Once they integrate there are two possibilities. Nodes that do NOT use the Key Slot Mode allow their other frames (static and dynamic) to be automatically enabled as soon as they integrate. The underlying assumption is that the mere fact that they integrated is sufficient confirmation of their conformance to the schedule to allow these transmissions to be enabled. Nodes that use the Key Slot Mode must enable the transmission of their other frames with an explicit host command. The underlying assumption in this case is that the host will first perform some sort of explicit confirmation that its transmitted frames are properly timed before it executes the command. This covers the scenario where only the transmission timing is flawed - in a way that does not impact the ability to integrate. The likely way to perform this confirmation is to configure another node to explicitly confirm the timing of the Key Slot frame(s) with a flag in a frame that it transmits. It simply sends a specified frame with the confirmation flag set if it receives the single frame from the node whose timing it is supervising. If the host sees receives this confirmation flag, it issues the command to the controller to disable Key Slot Mode and this causes all frames to be enabled.
Rationale:	FlexRay provides support for Key Slot Mode.
Use Case:	Active NM only after controller transitions from Key Slot Mode to normal communication (i.e., all slots are activated). Key Slot Mode is used for Diagnostics to limit transmission of all slots until node is sure that it is not interfering with transmission of normal communication messages by other nodes, i.e., the node transitions from Key Slot Mode when it is sure that it is in a compatible system mode.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05223] The number of Startup Frames shall be available ⌈

Description:	It shall be possible to detect the situation that no startup frames are present but the FlexRay is still operating because of still available sync frames.
Rationale:	Every network consists of at least 2 Coldstart nodes and these coldstart nodes send startup frames. All Coldstart nodes are shut down if no startup frame is present thus it is impossible for any node to integrate himself into the network without a previous shutdown of all nodes. The indication could be used to "force a shutdown of all nodes" to allow for a restart with all nodes.
Use Case:	A network with 3 coldstart nodes and 2 sync nodes. If the 3 coldstart nodes fall asleep/perform a restart because of an error (e.g. reset or low voltage) it is necessary to restart the FlexRay to allow the reintegration of all FlexRay nodes.

▽

△

Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05220] The buffer shall be immediatly accessible for read operations ⌈

Description:	The FlexRay related BSW modules of the AUTOSAR communication stack shall provide a method to allow immediate buffer access for read operations.
Rationale:	If a PDU is received by the communication controller in every cycle, but where the task is only acting on it occasionally, it does not make sense, that the FlexRay Driver and Interface transfers this PDU every time, but only on request.
Use Case:	To introduce functionality for direct read access to the CC is especially useful, if the FlexRay is solely used as a high-bandwidth replacement for an event-driven protocol, e.g. CAN.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05225] The CHI Command shall be called at any time ⌈

Description:	It shall be possible to call the CHI Command at an arbitrary point in time
Rationale:	Influencing factor of the start-up behavior of a FlexRay Cluster
Use Case:	The start-up time of an ECU can be taken into consideration to ensure that start-up attempts are not missed on receiver side.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05224] Payload Preamble Indicator Bit shall be supported ⌈

Description:	It shall be possible to switch the Payload Preamble Indicator Bit
Rationale:	The payload preamble indicator indicates whether or not an optional vector is contained within the payload segment of the frame transmitted
Use Case:	If the frame is transmitted in the static segment the payload preamble indicator indicates the presence of a network management vector at the beginning of the payload. If the frame is transmitted in the dynamic segment the payload preamble indicator indicates the presence of a message ID at the beginning of the payload. If the frame is transmitted in the static segment the payload preamble indicator indicates the presence of a network management vector at the beginning of the payload.
Dependencies:	–

▽

△

Supporting Material:	–

⌋

[SRS_Fr_05221] The Job list Execution shall be triggered by a synchronized task
⌈

Description:	The FlexRay related BSW modules of the AUTOSAR communication stack shall provide a method to trigger the functionality, usually implemented in the interrupt service routine by an OS-task, instead of using the FlexRay interrupt. Therefore, if the OS provides a method to synchronize a task to the FlexRay cycle, e.g. by using a Synchronized Rate Monotonous Scheduler, it can trigger the FlexRay protocol stack without wasting crucial interrupt resources.
Rationale:	AUTOSAR allows letting event-driven tasks act on the time-triggered FlexRay system. This is done by synchronizing the FlexRay stack to the FlexRay cycle (FlexRay Interrupt Service Routine, JobList, ...). In future systems, not only the FlexRay stack, but also the applications will be synchronized to use all advantages of time-triggered systems. When this happens, all transmit and receive operations by the task will already be synchronized to the FlexRay cycle. The internal synchronization mechanism of the FlexRay stack is then not necessary any more. The resource consumption (Memory, Processing Power, Runtime, Application jitter through interrupt load), currently a significant factor, can be dramatically reduced, when the Job List Execution can be triggered by a synchronized task directly. Therefore the synchronization using an interrupt service routine, as currently required by the AUTOSAR FlexRay documents (Requirements, Interface, Driver, ...), shall not remain the only solution. There shall be an option to provide the synchronization using a synchronized OS-task.
Use Case:	Systems, where not only the FlexRay stack, but also the applications are synchronized. The method will reduce interrupt resource consumption.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.1 FlexRay Interface

4.2.1.1.1 General Requirement

[SRS_Fr_05004] The FlexRay Interface shall provide a PDU-based data API to all upper layers. ⌈

Description:	The FlexRay Interface shall provide a PDU-based data API to all upper layers.
Rationale:	A PDU-based data API is needed in order to fulfill [SRS_Fr_05003] in an optimal way. In order to fulfill [SRS_Fr_05002], the FlexRay Interface needs to abstract the synchronous features of the FlexRay protocol. In order to fulfill [SRS_Fr_05003], a FlexRay Slot-based API is not possible. On the other hand, a FlexRay Cell-based API needs a lot of resources (one local memory space per FlexRay Cell) and performance (e.g. for the identification of the most recently received value). Moreover, in the static segment of the FlexRay Cycle, all FlexRay Cells always have the same length, independent of the Cell contents. To obtain an optimal use of the FlexRay bandwidth, the FlexRay Interface shall allow transferring PDUs with different transmission periods in the same FlexRay Cell.
Use Case:	The following series-relevant example: An ECU sends the following PDUs using a 5ms Cycle: • Acceleration (A) with a 2,5ms period • Status (S) with a 5ms period • Controller Status (C) with a 20ms period • Fault Status (F) with a 80ms period Conclusion: Only half the bandwidth is required if multiple PDUs are scheduled per FlexRay Cell.
Dependencies:	[SRS_Fr_05002], [SRS_Fr_05003]
Supporting Material:	–

⌋

[SRS_Fr_05010] Each PDU shall have one PDU-ID ⌈

Description:	The FlexRay Interface shall identify each PDU with a unique PDU-ID, based on the operation to be carried out with this PDU.
Rationale:	Unique PDU-IDs are being used in each BSW module throughout the AUTOSAR Com stack to identify each single PDU and to decide on the handling of the respective PDU. Within the FlexRay Interface, independent sets of PDU-IDs may be used for independent types of operation (i.e. sending & receiving). Unlike with other communication systems - as for example CAN - the Frame-ID is not sufficient for the identification of an PDU
Use Case:	Interaction of the FlexRay Interface with higher layer BSW modules, e.g. the PDU-Router.

▽

△

Dependencies:	[SRS_Fr_05004]
Supporting Material:	–

⌋

[SRS_Fr_05126] PDU Update/Valid Information shall be handled ⌈

Description:	The FlexRay Interface shall be configurable at system configuration time to generate, transmit, and (upon reception) handle PDU-based update/valid information where necessary. Per PDU, this information shall not consume more than one bit in the FlexRay Frame. This feature shall be configurable at system configuration time independently for each FlexRay Frame.
Rationale:	In certain configurations of the FlexRay Interface and the FlexRay CC, this information is necessary in order for the receiving FlexRay Interface to be able to decide whether a specific PDU contained in a received FlexRay Frame is up-to-date or outdated.
Use Case:	Assume two PDUs to be packed into the same FlexRay Frame. Assume further that only one of the two PDUs has been updated before the scheduled sending of this Frame, and the other PDU has not. The receiver needs to be able to recognize which of the PDUs is valid (i.e. contains updated data) and which is not (i.e. contains invalid or old data). Therefore, this update information has to be generated by the sending FlexRay Interface and transmitted via the FlexRay bus to the receiving FlexRay Interface which evaluates it and acts according to this evaluation.
Dependencies:	–
Supporting Material:	Signal update information of COM

⌋

[SRS_Fr_05097] The FlexRay Interface shall be able to communicate with at least four FlexRay Drivers ⌈

Description:	The FlexRay Interface shall be able to communicate with at least four FlexRay Drivers. The actual number of FlexRay Drivers serviced by the FlexRay Interface shall be defined at system configuration time and shall be less or equal to the number of FlexRay Drivers this FlexRay Interface supports.
Rationale:	Required in order to fulfill [SRS_Fr_05007] for four different types of FlexRay Communication Controllers, each of them requiring their own FlexRay Driver.
Use Case:	–
Dependencies:	[SRS_Fr_05007], [SRS_Fr_05065]
Supporting Material:	–

⌋

[SRS_Fr_05007] The FlexRay Interface shall be able to communicate with at least four FlexRay CCs via the appropriate FlexRay Driver(s) ⌈

Description:	The FlexRay Interface shall be able to communicate with at least four FlexRay CCs via the appropriate FlexRay Driver(s). The actual number of CCs operated by the FlexRay Interface shall be defined at system configuration time and shall be less or equal to the number of CCs this FlexRay Interface supports.
Rationale:	Future network topologies might demand more than one FlexRay CC on one ECU.
Use Case:	An ECU connected to two FlexRay Clusters.
Dependencies:	[SRS_Fr_05065], [SRS_Fr_05097]
Supporting Material:	–

⌋

[SRS_Fr_05130] The FlexRay Interface shall support PDU transmission buffer queues ⌈

Description:	The FlexRay Interface shall support PDU transmission buffer queues (FIFOs) of configurable size located in a higher BSW module by queuing the transmit requests originating from this BSW module.
Rationale:	A higher BSW module might use a FIFO to queue PDUs to be transmitted, but might forward the transmission request to the FlexRay Interface faster (within a limited time interval) than this PDU can be transmitted via FlexRay.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05215] The FlexRay Interface shall provide an API that cancels transmit request ⌈

Description:	The FlexRay Interface shall provide an API that cancels transmit request.
Rationale:	Cancellation API is a functionality to remove transmit request from CC's buffer which is realized by calling an API provided by FlexRay Driver.
Use Case:	When transmission of L-PDU is suspended due to the congested condition in dynamic segment and its containing data becomes outdated, such a request should be cancelled by this API.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05060] Scheduling of Copy Operation into/from FlexRay CC shall be possible ⌈

Description:	The FlexRay Interface shall be configured at system configuration time to copy, by means of the corresponding FlexRay Driver, the data into/from the FlexRay CC's transmit/receive buffers at the desired point in the FlexRay global time. This copy operation shall be carried out in a task configured (i.e. scheduled) at system configuration time. This task has to be synchronous to the communication schedule of the involved FlexRay CC. Together with the copy operation, a reconfiguration of a transmit/receive buffer during normal active mode may be carried out, if supported by the FlexRay CC. The configuration of the copy operation (i.e. the schedule) shall be changeable by a flashing process.
Rationale:	FlexRay Controllers have to be serviced synchronously to the FlexRay communication schedule.
Use Case:	–
Dependencies:	[SRS_Fr_05058], [SRS_Fr_05034]
Supporting Material:	–

⌋

[SRS_Fr_15060] FlexRay Network Management Scheduling Timing Window Relief shall be available ⌈

Description:	The timing window to schedule the NM task when using the Static segment NM Vector hardware service is too constrained and a timing window relief is desired. Particularly, there is a dependence between when a NM task can be scheduled and when the NM message in transmitted in a slot in the same cycle. A timing window relief could aim at the following two windows: • The NM Task could be scheduled anywhere in the Communication Cycle. • The NM Task could be scheduled anywhere in the Static segment of the Communication Cycle.
Use Case:	1. Network Management need not run every FlexRay communication cycle saving processing time and resources. 2. Less stringent constraints on scheduling Network Management main function and transmission slots.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05061] NM Vector feature shall be supported ⌈

Description:	The NM Vector feature provided by the FlexRay Communication Controller shall be supported
Rationale:	The NM Vector can be used by the FlexRay Network Management
Use Case:	Shutting down a FlexRay cluster synchronously

▽

△

Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05096] Communication controllers shall be assigned to FlexRay Driver. ⌈

Description:	The configuration of the FlexRay Interface shall specify which FlexRay Communication Controller is to be served by which FlexRay Driver. Thus a mapping between used FlexRay Drivers and available FlexRay Controllers is made though the interface configuration at system configuration time.
Rationale:	This information has to be available to the FlexRay Interface in order to invoke the correct Driver for a given FlexRay Controller on an ECU.
Use Case:	–
Dependencies:	[SRS_Fr_05056] Configuration of the FlexRay Interface at System Configuration Time
Supporting Material:	–

⌋

4.2.1.1.2 Initialization

[SRS_Fr_05013] The local Memory Space shall be initialized ⌈

Description:	The FlexRay Interface shall provide a software interface to initialize the local memory space used to store the PDU data and their corresponding properties.
Rationale:	The initialization of the local memory space is necessary to store the transferred data from the FlexRay Controller. Usually, variables (e.g. pointers) have to be initialized before they are being used during runtime.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05031] A FlexRay CC shall be initialized and configured ⌈

Description:	The FlexRay Interface shall provide a software interface to initialize and configure a specific FlexRay CC. Thereby, the transmit/receive buffers and the low level parameters of the FlexRay CC shall be configured.
Rationale:	–

▽

△

Use Case:	Initial configuration.
Dependencies:	[SRS_Fr_05116], [SRS_Fr_05011], [SRS_Fr_05012]
Supporting Material:	–

⌋

[SRS_Fr_05034] The configuration data shall be modifiable by a Flashing Process ⌈

Description:	All configuration data of the FlexRay Interface defined at system configuration time shall be modifiable by a flashing process. These modifiable configuration data encompass (amongst other configuration items) the scheduling of the copy operation into/from the FlexRay CCs' transmit/receive buffers as well as the configuration of those buffers and the FlexRay Interface's local memory space. The modification shall be carried out in a specific mode (e.g. in the garage) and not during run-time (i.e. while the car is being driven).
Rationale:	Re-flashing of a changed communication schedule.
Use Case:	If e.g. the only modification in a configuration is the Frame-ID of a specific PDU (i.e. the FlexRay Frame in which this PDU is being sent), this shall not automatically implicate a valid compilation of the complete ECU code! Only the altered parameters need to be modified.
Dependencies:	[SRS_Fr_05060], [SRS_Fr_05012], [SRS_Fr_05013]
Supporting Material:	–

⌋

[SRS_Fr_05042] The FlexRay Interface shall allow switching from one configuration to another one in Normal Active Mode ⌈

Description:	The FlexRay Interface shall allow switching from one configuration of a specific FlexRay Communication Controller's transmit/receive buffers and the local memory space to another one, both stored in the ECU's memory space. This switching shall be possible during runtime, i.e. while the CC is in normal active mode, if supported by this FlexRay CC. It shall be ensured that none of the selectable configurations violate the rules for Slot Multiplexing set forth in [FR_PROT_SPEC]. The switching of the configuration includes a reconfiguration of the FlexRay Communication Controller's transmit/receive buffers. Therefore the following restrictions apply: • In the dynamic segment any FlexRay Communication Controller's transmit/receive buffer shall be used maximally once per Cycle. This limits the reconfiguration possibilities and thus restricts the number of transmittable (sent and received) Frames per dynamic segment to the accumulated

▽

▽

△

	△ number (over all CCs on one ECU) of transmit/receive buffers connected to one Cluster. The FlexRay Interface shall support at least 4 different configurations. The actual number of configurations stored in the ECU's memory space shall be defined at system configuration time and shall be less or equal to the number of configurations this FlexRay Interface supports. For safety reasons it shall be possible to completely deactivate this feature pre compile time. For safety reason such a switching operation shall not be carried out while the car is being driven, but only in the garage in a specific mode.
Rationale:	–
Use Case:	Specific flash mode for the gateway: • In order to increase the throughput of the gateway, and thus speed up the flashing process, transmit/receive buffers normally used for application data transmission could temporarily be used for the flashing process.
Dependencies:	[SRS_Fr_05012], [SRS_Fr_05013]
Supporting Material:	–

⌋

4.2.1.1.3 Start-up Operation

[SRS_Fr_05015] The FlexRay Interface shall provide a software interface to start-up a specific FlexRay CC ⌈

Description:	The FlexRay Interface shall provide a software interface to start-up a specific FlexRay CC.
Rationale:	Start-up of a specific FlexRay Controller after proper configuration.
Use Case:	–
Dependencies:	[SRS_Fr_05109], [SRS_Fr_05031]
Supporting Material:	–

⌋

[SRS_Fr_05018] The FlexRay Interface shall provide a software interface to send a wake-up pattern on a channel or CC ⌈

Description:	The FlexRay Interface shall provide a software interface to send a wake-up pattern on a specific FlexRay Channel of a specific FlexRay CC. The FlexRay Specification allows only a wake-up on one Channel at a time. However, the FlexRay Interface shall support the sending of a wake-up pattern on any one of the two FlexRay Channels. The FlexRay Interface shall observe the restrictions set forth in the FlexRay Protocol Specification concerning reception of a wake-up pattern or a Frame header during own wake-up pattern sending attempts.
Rationale:	Enable waking up a FlexRay Cluster.
Use Case:	–
Dependencies:	[SRS_Fr_05117]
Supporting Material:	[FR_PROT_SPEC]

⌋

[SRS_Fr_05158] The wake-up reason of a specific FlexRay Transceiver device shall be available ⌈

Description:	The FlexRay Interface shall provide a software interface to get the wake-up reason of a specific FlexRay Transceiver device.
Rationale:	Provide EcuM with a possibility to find out the wake-up reason.
Use Case:	–
Dependencies:	[SRS_Fr_05144] Get FlexRay Transceiver Wake-up Reason
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05159] The FlexRay Interface shall provide a software interface to enable the wake-up indication of a specific FlexRay Transceiver device ⌈

Description:	The FlexRay Interface shall provide a software interface to enable the wake-up indication of a specific FlexRay Transceiver device.
Rationale:	EcuM might be interested in a wake-up event via FlexRay.
Use Case:	–
Dependencies:	[SRS_Fr_05160] Disable FlexRay Transceiver Wake-up Indication
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05160] The FlexRay Interface shall provide a software interface to disable the wake-up indication of a specific FlexRay Transceiver device. ⌈

Description:	The FlexRay Interface shall provide a software interface to disable the wake-up indication of a specific FlexRay Transceiver device.
Rationale:	EcuM might not be interested in a wake-up event via FlexRay.
Use Case:	–
Dependencies:	[SRS_Fr_05159] Enable FlexRay Transceiver Wake-up Indication
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05161] Pending Wake-up Events of a Transceiver shall be cleared if necessary ⌈

Description:	The FlexRay Interface shall provide a software interface to clear all pending wake-up events of a specific FlexRay Transceiver device.
Rationale:	EcuM needs to reset the wake-up events of FlexRay Transceiver devices.
Use Case:	Wake-Up via FlexRay Communications Bus
Dependencies:	–
Supporting Material:	[FR_EPL_SPEC]

⌋

4.2.1.1.4 FlexRay Specific Information

This section includes all the FlexRay-specific requirements.

[SRS_Fr_05022] FlexRay CC POC Status shall be available ⌈

Description:	The FlexRay Interface shall provide a software interface to get the CC POC status of a specific FlexRay CC. All information contained in the vPOC structure as defined in chapter "2.2.1.3 POC status" of the [FR_PROT_SPEC] shall be returned.
Rationale:	Software modules like Mode Manager, Network Management, or DCM might be interested in the FlexRay CC POC status.
Use Case:	–
Dependencies:	[SRS_Fr_05120] Get FlexRay CC POC Status
Supporting Material:	–

⌋

[SRS_Fr_05039] The Operation Mode of a FlexRay Transceiver shall be set ⌈

Description:	The FlexRay Interface shall provide a software interface to set the operation mode of a specific FlexRay Transceiver device. According to [FR_EPL_SPEC], at least these operation modes shall be supported: • BD_Normal • BD_Standby • BD_Receive_only • BD_Sleep
Rationale:	Provide ComM with a possibility to set the operation mode of a specific FlexRay Transceiver device.
Use Case:	–
Dependencies:	[SRS_Fr_05166], [SRS_Fr_05157]
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05157] The Operation Mode of a FlexRay Transceiver shall be available ⌈

Description:	The FlexRay Interface shall provide a software interface to get the operation mode of a specific FlexRay Transceiver device. According to [FR_EPL_SPEC], at least these operation modes shall be supported: • BD_Normal • BD_Standby • BD_Receive_only • BD_Sleep
Rationale:	Provide FlexRay StateManager with a possibility to get the operation mode of a specific FlexRay Transceiver device.
Use Case:	–
Dependencies:	[SRS_Fr_05167], [SRS_Fr_05039]
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05174] The FlexRay Interface shall provide services to handle interrupts of a FlexRay Communication Controller. ⌈

Description:	The FlexRay Interface shall provide services to handle interrupts of a FlexRay Communication Controller. At least the following functionality shall be provided: • get interrupt source • service interrupt • enable interrupt • disable interrupt • acknowledge interrupt.
Rationale:	The FlexRay Interface might use an alarm (absolute timer interrupt) of a FlexRay Communication Controller to run synchronously to the FlexRay global time. This interrupt has to be serviced.
Use Case:	–
Dependencies:	[SRS_Fr_05125] Interrupt Handling
Supporting Material:	–

⌋

4.2.1.1.5 Normal Operation

[SRS_Fr_05170] PDUs received via the FlexRay communication system shall be retrieved ⌈

Description:	The FlexRay Interface shall retrieve PDUs received via the FlexRay communication system and indicate the reception to the appropriate higher-layer BSW module, allowing this module to retrieve the PDU data. Depending on static/dynamic configuration of the FlexRay Interface, it might be possible that one FlexRay Frame contains more than one PDU. In this case, the FlexRay Interface shall disassemble the FlexRay Frame and indicate the reception of each PDU to the appropriate higher BSW module. If the use of PDU-based update/valid information has been configured for a FlexRay Frame, the reception indication shall consider this information.
Rationale:	As explained in the valid requirement, the FlexRay Interface API shall be PDU-based. The access via the PDU is the only possibility to access the FlexRay data.
Use Case:	An application needs to receive information from the FlexRay network.
Dependencies:	[SRS_Fr_05004], [SRS_Fr_05010], [SRS_Fr_05126]
Supporting Material:	–

⌋

[SRS_Fr_05027] A PDU shall be transmitted via the FlexRay communication system ⌈

Description:	The FlexRay Interface shall provide a software interface to transmit a PDU via the FlexRay communication system Depending on static configuration of the FlexRay Interface, it might be possible that one FlexRay Frame contains more than one PDU. In this case, the FlexRay Interface shall assemble the FlexRay Frame of the corresponding PDUs before the Frame is transmitted. If the use of PDU-based update/valid information has been configured for a FlexRay Frame, the FlexRay Interface shall correctly generate this information.
Rationale:	As explained in the valid requirement the FlexRay Interface API shall be PDU-based. The access via the PDU is the only possibility to access the FlexRay data.
Use Case:	An applications needs to send information over the FlexRay network.
Dependencies:	[SRS_Fr_05004], [SRS_Fr_05010], [SRS_Fr_05126]
Supporting Material:	–

⌋

[SRS_Fr_05171] A PDU Transmit Confirmation shall be provided ⌈

Description:	The FlexRay Interface shall be configurable to provide the appropriate higher-layer BSW module with a transmit confirmation for a PDU this BSW module has requested to be sent. At system configuration time it shall be configurable per PDU whether a transmit confirmation shall be provided.
Rationale:	In the dynamic segment, the transmission of a Frame is not guaranteed. Therefore, the application needs information whether a PDU has been sent.
Use Case:	–
Dependencies:	[SRS_Fr_05004] PDU-Based API
Supporting Material:	–

⌋

4.2.1.1.6 Shutdown Operation

[SRS_Fr_05016] A FlexRay CC Communication shall be aborted when wanted ⌈

Description:	The FlexRay Interface shall provide a software interface to immediately abort the communication of a specific FlexRay CC by setting this FlexRay CC into the "HALT" state. Thus, the FlexRay CC will stop participating in the communication and lose synchronicity with the FlexRay global time.
Rationale:	–
Use Case:	Error confinement
Dependencies:	[SRS_Fr_05114] Abortion of FlexRay CC Communication

▽

△

Supporting Material:	–

⌋

[SRS_Fr_05063] A FlexRay CC Communication shall be halted when wanted ⌈

Description:	The FlexRay Interface shall provide a software interface to halt the communication of a specific FlexRay CC at the end of the current Communication Cycle by setting this FlexRay CC into the "HALT" state at the end of the current Communication Cycle. Thus, the FlexRay CC will stop participating in the communication and lose synchronicity with the FlexRay global time.
Rationale:	–
Use Case:	Error confinement
Dependencies:	[SRS_Fr_05115], [SRS_Fr_05016]
Supporting Material:	[FR_PROT_SPEC]

⌋

4.2.1.1.7 Fault Operation

[SRS_Fr_05175] The Error Informations shall be provided ⌈

Description:	The FlexRay Interface shall provide the DEM with error-related information using the aggregated channel status that is only known to the FlexRay Interface.
Rationale:	Some error-related information is only known to the FlexRay Interface.
Use Case:	The following error-related information shall be derived using the using the aggregated channel status: • SyntaxError: Flag that reports a syntax error within a slot. • ContentError: Flag that reports a content error within a slot. • TxConflict: Flag that reports a transmission conflict within a slot. • Bviolation: Flag that reports a boundary violation within a slot.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05200] The Error Information in Transmitted Frame shall be available ⌈

Description:	The FlexRay Interface shall call API of [BSW05207] periodically that detects error information in transmitted frames. Pre-compile configuration parameter shall determine whether this functionality is activated.
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	Applications can use this information as an indicator of network quality.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05201] The Error Information in Received Frame shall be available ⌈

Description:	The FlexRay Interface shall call API of [BSW05208] periodically that detects error information in received frames. Pre-compile configuration parameter shall determine whether this functionality is activated.
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	Applications can be implemented with an indirect-NM functionality when the error information mentioned above is provided.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05202] The Error Information in NIT shall be available ⌈

Description:	The FlexRay Interface shall call API of [BSW05209] periodically that detects error information in NIT. Pre-compile configuration parameter shall determine whether this functionality is activated.
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	Application can use this information as an indicator of network quality.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05203] The Error Information in Bus Driver shall be available ⌈

Description:	The FlexRay Interface shall call API of [BSW05212] periodically that detects error information in BD. Pre-compile configuration parameter shall determine whether this functionality is activated.
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	Applications could take actions to recover the failure cause like resetting the modules when they receive this error information.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05204] A Checking Function of L-PDU Length shall be customizabled ⌈

Description:	In the FlexRay Interface, pre-compile configuration parameter shall define a mode in checking function of L-PDU length. In a loose mode, L-PDU with expected length or longer length than expected should be accepted. In a strict mode, only L-PDU with expected length should be accepted.
Rationale:	When using a dynamic segment, receivers would likely get L-PDUs with unexpected length. There are two requirements like follows on the behavior for such a case. • Longer L-PDU should be rejected because such an L-PDU would likely be an illegal one that has been mistakenly transmitted from other nodes. • Longer L-PDU should be accepted. This mode is necessary when a user assumes that the length of L-PDU will be expanded in the future version of ECU's software. To satisfy these use-cases, there should be two modes in the functionality's behavior.
Use Case:	(See the description in Rationale)
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.2 FlexRay Driver

4.2.1.2.1 Valid Requirements

[SRS_Fr_05064] Abstraction of FlexRay CC-specific Implementation shall be provided ⌈

Description:	The FlexRay Driver shall provide an abstraction from the FlexRay CC-specific implementation as well as a generic FlexRay Driver API to the upper software layer (FlexRay Interface).
Rationale:	All FlexRay Drivers should provide the same API semantics/signature. Upper layers should not depend on the concrete implementation of specific features (e.g. optional features) of the CC. They need to be hardware-independent.
Use Case:	Several (different) FlexRay CCs (e.g. from different manufacturers) handled by one FlexRay Interface.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05065] The FlexRay Driver shall be able to communicate with at least four FlexRay CCs of the same type ⌈

Description:	The FlexRay Driver shall be able to communicate with at least four FlexRay CCs of the same type. The actual number of CCs operated by one FlexRay Driver shall be defined at system configuration time and shall be less or equal to the number of CCs this FlexRay Driver supports.
Rationale:	Future network topologies might demand more than one FlexRay CC on one ECU.
Use Case:	An ECU connected to two FlexRay Clusters.
Dependencies:	[SRS_Fr_05007], [SRS_Fr_05097]
Supporting Material:	–

⌋

[SRS_Fr_05005] The CC Hardware FIFO Mechanism shall be supported ⌈

Description:	The FlexRay Driver shall support the CC hardware FIFO mechanism, if supported by the FlexRay CC, and abstract its usage because this mechanism is optional.
Rationale:	In the future, a FlexRay CC could have a low number of transmit/receive buffers to reduce its cost. Then the high number of different Slot-IDs - with slow communication requirements - in the dynamic segment requires a FIFO on the reception side.
Use Case:	The hardware FIFO could be configured for the dynamic segment and / or the static segment of the FlexRay Communication Cycle.

▽

△

Dependencies:	–
Supporting Material:	• FlexRay Protocol Specification 2.1 • FlexRay Protocol Specification 3.0

⌋

[SRS_Fr_15006] Hardware Message ID Filter Mechanism according to the FlexRay Protocol Specification 3.0 shall be supported ⌈

Description:	Hardware filtering mechanism provided by the FlexRay Communication Controller that can be used for selecting receive buffers based on a message ID within the dynamic segment shall be supported
Rationale:	• Avoidance of unnecessary receive indications to upper layers • Dynamic bandwidth allocation
Use Case:	Optimized usage of FlexRay Transport Protocol
Dependencies:	–
Supporting Material:	FlexRay Protocol Specification 3.0

⌋

[SRS_Fr_15007] Time Triggered Master mode provided by FlexRay 3.0 Communication Controller shall be available ⌈

Description:	It shall be possible to use the Time Triggered Master mode provided by FlexRay 3.0 Communication Controller. I. e. support of: • TT-L Time Triggered Local Master Sync • TT-E Time Triggered External Sync It is also possible to set up a TT-L/TT-E cluster with more than one master for increased robustness
Rationale:	Setup up a network with two or more time synchronized clusters (e.g. a Backbone is the timing source for multiple clusters, cascaded clusters)
Use Case:	Several Topologies
Dependencies:	–
Supporting Material:	FlexRay Protocol Specification 3.0

⌋

[SRS_Fr_05008] FlexRay 2.1 and 3.0 Hardware shall be supported ⌈

Description:	FlexRay 2.1 and 3.0 Communication Controllers and Transceivers shall be supported
Rationale:	Valid FlexRay Hardware feature Support is required by several OEMs
Use Case:	[SRS_Fr_05005], [SRS_Fr_15006], [SRS_Fr_15007]
Dependencies:	–

▽

△

Supporting Material:	• FlexRay Protocol Specification 3.0 • FlexRay Electrical Physical Layer 3.0 • FlexRay Protocol Specification 2.1 Rev. A • FlexRay Electrical Physical Layer 2.1 Rev. A

⌋

[SRS_Fr_15009] Dynamic LPdu payload length shall be supported ⌈

Description:	It shall be possible to transmit and receive dynamic LPdu payload length in the dynamic segment of a FlexRay Communication Cycle.
Rationale:	This feature is required by the FlexRay Transport Protocol and can also be used by the AUTOSAR Communication
Use Case:	Usage of ISO FlexRay Transport Protocol and Support of dynamic Signal Length sent by a SW-C
Dependencies:	–
Supporting Material:	AUTOSAR FlexRay Transport Protocol

⌋

[SRS_Fr_05024] The software interface of the Driver shall be independent of the CC buffers' configuration ⌈

Description:	The software interface of the FlexRay Driver shall be independent of the FlexRay CC transmit/receive buffers' configuration (transmit, receive, filtering ...) and their access.
Rationale:	The FlexRay transmit/receive buffer configuration and properties shall not have any influence on the upper-layer software.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05066] L-SDU-Based API shall be available ⌈

Description:	The FlexRay Driver shall provide an L-SDU-based data handling to all upper layers. Based on the unique L-SDU identifier (this is NOT the Frame-ID according to [FR_PROT_SPEC]) and according to rules defined at system configuration time, the FlexRay Interface assembles L-SDU for sending or extracts them for reception, resp. Therefore, the FlexRay Driver shall provide an L-SDU-based data handling to all upper layers.
Rationale:	Payload contained in L-SDU is usually transferred into/out of a FlexRay Cell via transmit/receive buffers allocated in the FlexRay Controller.
Use Case:	–

▽

△

Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05205] The FlexRay Driver shall provide an API that cancels transmit request ⌈

Description:	The FlexRay Driver shall provide an API that cancels transmit request.
Rationale:	Cancellation API is a functionality to remove transmit request from CC's buffer.
Use Case:	When transmission of L-PDU is suspended due to the congested condition in dynamic segment and its containing data becomes outdated, such a request should be cancelled by this API.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05206] FlexRay Driver shall provide a method that reinitializes CC's functionality ⌈

Description:	FlexRay Driver shall provide a method that reinitializes CC's functionality
Rationale:	When trouble occurs in the hardware level, it's likely to fix the cause by resetting the hardware.
Use Case:	This function shall be executed when so many errors are detected in FlexRay modules.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.2.2 Configuration

[SRS_Fr_05058] The configuration of the FlexRay Driver shall be defined at system configuration time. ⌈

Description:	The configuration of the FlexRay Driver shall be defined at system configuration time, i.e. before the execution of the FlexRay Driver code.
Rationale:	The purpose of the FlexRay Driver is to carry out data processing that has been configured at system configuration time, i.e. before the execution of the FlexRay Driver code. It shall not decide or calculate during runtime any communication parameters like the assignment of Slots to ECU or the packaging of PDUs into FlexRay Frames.

▽

△

Use Case:	–
Dependencies:	[SRS_Fr_05034] Configuration Modifiable by a Flashing Process
Supporting Material:	–

⌋

[SRS_Fr_05059] The Driver shall be configure the CC's transmit/receive buffers ⌈

Description:	The FlexRay Driver shall configure the FlexRay CC's transmit/receive buffers according to the configuration of the FlexRay Driver defined at system configuration time.
Rationale:	The FlexRay CC's transmit/receive buffers need to be configured before the FlexRay CC can be used for communication.
Use Case:	–
Dependencies:	[SRS_Fr_05058] Configuration of FlexRay Driver at System Configuration Time
Supporting Material:	–

⌋

4.2.1.2.3 Initialization

[SRS_Fr_05116] Initialization of FlexRay CC shall be available ⌈

Description:	The FlexRay Driver shall provide a software interface to initialize and configure a specific FlexRay CC. Thereby, the transmit/receive buffers and the low level parameters of the FlexRay CC shall be configured.
Rationale:	–
Use Case:	Initial configuration.
Dependencies:	[SRS_Fr_05031], [SRS_Fr_05011], [SRS_Fr_05012]
Supporting Material:	–

⌋

[SRS_Fr_05011] Initialization of the Low-Level Parameters shall be available ⌈

Description:	The FlexRay Driver shall provide a software interface to initialize the FlexRay low-level parameters. The low-level parameters cover all the configurable parameters defined in the [FR_PROT_SPEC] and also the FlexRay CC-specific parameters or CC-specific services (e.g. concerning interrupts).
Rationale:	The initialization of the low-level parameters is necessary to communicate on the bus with a FlexRay Controller. The set of CC parameters is FlexRay CC-specific.

▽

△

Use Case:	Initialization of the FlexRay communication system.
Dependencies:	[SRS_Fr_05116] Initialization of FlexRay CC
Supporting Material:	–

⌋

[SRS_Fr_05012] Initialization of the FlexRay CC Transmit/Receive Buffers shall be available ⌈

Description:	The FlexRay Driver shall provide a software interface to initialize the transmit/receive buffers of a specific FlexRay CC with a default configuration.
Rationale:	The initialization of the transmit/receive buffers is necessary to transfer data with a FlexRay Controller. The configuration of the transmit/receive buffers and filtering mechanisms is FlexRay CC-specific.
Use Case:	–
Dependencies:	[SRS_Fr_05116] Initialization of FlexRay CC
Supporting Material:	–

⌋

4.2.1.2.4 Start-up Operation

[SRS_Fr_05109] The FlexRay Driver shall provide a software interface to start-up a specific FlexRay CC ⌈

Description:	The FlexRay Driver shall provide a software interface to start-up a specific FlexRay CC.
Rationale:	Start-up of a specific FlexRay Controller after proper configuration.
Use Case:	–
Dependencies:	[SRS_Fr_05015], [SRS_Fr_05114], [SRS_Fr_05115]
Supporting Material:	–

⌋

[SRS_Fr_05117] A Wake-Up Pattern shall be sent on a specific channel of a CC ⌈

Description:	The FlexRay Driver shall provide a software interface to send a wake-up pattern on a specific FlexRay Channel of a specific FlexRay CC. The FlexRay Specification allows only a wake-up on one Channel at a time. However, the FlexRay Driver shall support the sending of a wake-up pattern on any one of the two FlexRay Channels. The FlexRay Driver shall observe the restrictions set forth in the FlexRay Protocol Specification concerning reception of a wake-up pattern or a Frame header during own wake-up pattern sending attempts.
Rationale:	Enable waking up a FlexRay Cluster.
Use Case:	–
Dependencies:	[SRS_Fr_05018] Sending of a Wake-Up Pattern
Supporting Material:	[FR_PROT_SPEC]

⌋

4.2.1.2.5 FlexRay Specific Information

This section includes all the FlexRay-specific status information needed by the upper-layer Software e.g. synchronization information, etc...

[SRS_Fr_05120] FlexRay CC POC Status shall be provided ⌈

Description:	The FlexRay Driver shall provide a software interface to get the CC POC status of a specific FlexRay CC. All information contained in the vPOC structure as defined in chapter "2.2.1.3 POC status" of the [FR_PROT_SPEC] shall be returned.
Rationale:	Software modules like Mode Manager, Network Management, or DCM might be interested in the FlexRay CC POC status.
Use Case:	–
Dependencies:	[SRS_Fr_05022] Get FlexRay CC POC Status
Supporting Material:	–

⌋

[SRS_Fr_05121] FlexRay CC Sync State shall be provided ⌈

Description:	The FlexRay Driver shall provide a software interface to get the sync state ("controller synchronous"/"controller asynchronous") of a specific FlexRay CC.
Rationale:	In order to be able to communicate via FlexRay, it is crucial that the FlexRay CC be synchronous to the FlexRay Cluster's global time. Software modules like Mode Manager, Network Management, or DCM might be interested in the sync state of a specific FlexRay CC.

▽

△

Use Case:	A distributed control might only be started once the communication via the FlexRay bus in possible. Therefore, knowledge about the sync state of the FlexRay CC is crucial
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.2.6 Normal Operation

[SRS_Fr_05106] The Buffer of a specific CC in Normal Active Mode shall be reconfigurable ⌈

Description:	The FlexRay Driver shall be able to reconfigure a specific transmit/receive buffer of a specific FlexRay Communication Controller in normal active mode, if supported by the FlexRay CC. The reconfiguration of a transmit/receive buffer of a FlexRay CC in normal active mode allows providing to the FlexRay Interface a higher number of (virtual) transmit/receive buffers than this FlexRay CC actually has. The configuration tool of the FlexRay Driver shall schedule these reconfiguration actions whenever it is necessary before/after the FlexRay Interface accesses a transmit/receive buffer of a FlexRay CC. The reconfiguration of a transmit/receive buffer of a FlexRay CC in normal active mode means that the same buffer is being used for different buffer configurations (i.e. Frame-ID and filter configuration, transmitting/sending, Channel A/B) without leaving normal active mode (i.e. without setting the FlexRay CC into configuration mode). In the dynamic segment, each transmit/receive buffer of a FlexRay CC shall be used maximally once per Cycle. This limits the reconfiguration possibilities and thus restricts the number of transmittable (sent and received) Frames per dynamic segment to the accumulated number (over all CCs on one ECU) of transmit/receive buffers connected to one Cluster. For safety reasons it shall be possible to completely deactivate this feature pre compile time.
Rationale:	The number of transmit/receive buffers affects the HW cost, therefore, the supplier offers low-cost controllers with a small number of transmit/receive buffers. And by using only dedicate transmit/receive buffer configurations, each small application would need a lot of buffer. Therefore, the buffer reconfiguration during normal active mode shall be supported by the FlexRay Driver.

▽

△

Use Case:	For example, this can be used to send many PDUs with slow communication requirements in the dynamic segment. Non-safety-critical use, in gateways, etc. Another example (in the static segment): PDU with a period of 1.25ms in a 2.5 ms Cycle length scheduled in the Static Slot 5 and 45. The same buffer should be use to send (or receive) the Slot with the Frame-ID 5 and 45 with the buffer reconfiguration in normal active mode.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_15106] A CC Buffer reconfiguration shall be possible at runtime ⌈

Description:	It shall be possible to perform a CC Buffer reconfiguration at runtime (i.e. it is not being defined during system configuration time) The set of hardware-independent configuration parameters consists of the following items: • Identifier of the CC Buffer • Direction: Transmission or Reception • FlexRay Slot Number • Cycle Counter Offset • Cycle Counter Repetition • FlexRay Channel • Payload Length of FlexRay Frame • FlexRay Header CRC
Rationale:	Dynamic bandwidth assignment to the XCP slaves by the XCP master.
Use Case:	Manipulation of parameters for adjustment purpose (e.g. chassis suspension, motor management,..)
Dependencies:	–
Supporting Material:	http://www.asam.net/doc_int/getfile/getfile.php?id=376

⌋

[SRS_Fr_05125] The FlexRay Driver shall provide services to handle interrupts of a FlexRay Communication Controller. ⌈

Description:	The FlexRay Driver shall provide services to handle interrupts of a FlexRay Communication Controller. At least the following functionality shall be provided: • get interrupt source • service interrupt • enable interrupt • disable interrupt • acknowledge interrupt.
Rationale:	Only the FlexRay Driver has knowledge on how to handle (e.g. enable/disable) a CC-specific interrupt.
Use Case:	–
Dependencies:	[SRS_Fr_05174] Interrupt Handling
Supporting Material:	–

⌋

4.2.1.2.7 Shutdown Operation

[SRS_Fr_05114] A FlexRay CC Communication shall be aborted when wanted ⌈

Description:	The FlexRay Driver shall provide a software interface to immediately abort of the communication of a specific FlexRay CC by setting this FlexRay CC into the "HALT" state. Thus, the FlexRay CC will stop participating in the communication and lose synchronicity with the FlexRay global time.
Rationale:	–
Use Case:	Error confinement
Dependencies:	[SRS_Fr_05016] Abortion of a FlexRay CC Communication
Supporting Material:	–

⌋

[SRS_Fr_05115] The FlexRay CC Communication shall be halted when wanted ⌈

Description:	The FlexRay Driver shall provide a software interface to halt the communication of a specific FlexRay CC at the end of the current Communication Cycle by setting this FlexRay CC into the "HALT" state at the end of the current Communication Cycle. Thus, this FlexRay CC will stop participating in the communication and lose synchronicity with the FlexRay global time.
Rationale:	–
Use Case:	Error confinement
Dependencies:	[SRS_Fr_05063] Halt of a FlexRay CC Communication

⌋

△

Supporting Material:	–

⌋

[SRS_Fr_05207] An API to Detect Error Information in Transmitted Frame shal be provided ⌈

Description:	The FlexRay Driver shall provide an API that detects the following error status in transmitted frames and notifies them to application level. • vSS!SyntaxError • vSS!ContentError • vSS!BViolation
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05208] An API to Detect Error Information in Received Frame shall be provided ⌈

Description:	The FlexRay Driver shall provide an API that detects the following error information in received frames and notifies them to application level. • vSS!SyntaxError • vSS!ContentError • vSS!BViolation
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05209] An API to Detect Error Information in NIT shall be provided ⌈

Description:	The FlexRay Driver shall provide an API that detects the following error information in NIT and notifies them to application level. • vSS!SyntaxError • vSS!BViolation
Rationale:	The FlexRay modules should provide information that only the modules can detect.

▽

△

Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05210] Criteria on Validness of Received Frame shall be provided ⌈

Description:	The FlexRay Driver shall decide whether the received frame is valid in accordance with a criteria setting in pre-compiled parameters. When the criteria is set to a loose mode, the following state would be checked. The frame would be acknowledged as a valid one if ValidFrame is true. • vSS!ValidFrame When the criteria is set to a strict mode, the following states would be checked. The frame would be acknowledged as a valid one if ValidFrame is true and all the other error statuses are false. • vSS!ValidFrame • vSS!SyntaxError • vSS!ContentError • vSS!BViolation
Rationale:	The FlexRay Interface should provide a method to strictly check the error information in frames.
Use Case:	This ensures the network reliability.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.2.8 Fault Operation

[SRS_Fr_05211] Hardware Checking Function on CC shall be available ⌈

Description:	FlexRay Driver shall check whether CC is working properly at its initialization phase and notify error to application when illegal status is detected.
Rationale:	This functionality ensures that the hardware is working as expected.
Use Case:	Improvement of hardware reliability.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05222] The Transceiver Rx-Only Mode via FR State Manager shall be supported ⌈

Description:	This mode change should be visible to the Mode Management, but not to the application. The application should still get FullCommunication indicated as if the ECU is able to send frames on the bus. TxConfirmation for send requests shall still be performed during ECU "silent" mode. The "silent" mode should persist even when the bus is shut down and woken up again. After a reset the ECU should be able to send frames on the bus again.
Rationale:	It should be possible to switch off the Tx path of an ECU to send in its slots without having any collision on the bus.
Use Case:	Set a faulty ECU to "silent" to have the possibility to simulate the Tx path of the ECU while the connected actuators and sensors are still working.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.3 Transport Layer FlexRay

4.2.1.3.1 Configuration

[SRS_Fr_05077] Each N-SDU shall have a unique identifier ⌈

Description:	The FlexRay Transport Layer shall identify each N-SDU with a unique identifier, so the upper layer can address an N-SDU without any assumption on the addressing mode configuration of the FlexRay Transport Layer. Furthermore, a symbolic name may be assigned for each N-SDU identifier value to simplify usage of the API.
Rationale:	Independence of upper layer with the FlexRay Transport Layer addressing particularities, optimization. The PDU-Router routes all N-SDUs (FlexRay, CAN and LIN) regardless of the addressing mode particularities of the underlying protocols.
Use Case:	FlexRay - CAN TP gateway
Dependencies:	–
Supporting Material:	[ISO 15765-2] specification.

⌋

[SRS_Fr_05226] Transport Connection Properties "ISO 10681-2" ⌈

Description:	The FlexRay Transport Layer configuration shall be defined at system configuration time and shall define the following properties individually for each N-SDU (i.e. for each connection): • Associated N-PDU identifiers • Size of the associated N-PDU(s) • Acknowledge type • Transmit cancellation ON/OFF • Default values for Bandwidth Control Parameters • Target address • Source address • Connection type (1:1 or 1:n) • Timeout parameters according to ISO 10681-2
Rationale:	At runtime the FlexRay Transport module must have all the information required to manage a Transport Layer connection.
Use Case:	This information can be used at generation time to check the network configuration with a FlexRay Transport Layer point of view.
Dependencies:	–
Supporting Material:	ISO 10681-2

⌋

4.2.1.3.2 Initialization

[SRS_Fr_05088] FlexRay Transport Layer's variables shall be initialized ⌈

Description:	The FlexRay Transport Layer shall implement an interface for initialization in order to initialize all global variables of the FlexRay Transport Layer module and set all Transport Layer connections in a default state (Idle).
Rationale:	Basic functionality.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05089] The FlexRay Transport Layer services shall not be operational before initializing the module. ⌈

Description:	Before using the transmission capabilities of the Flex-Ray Transport Layer, it shall be initialized. If this is not the case, the services have to raise an error in development mode.

▽

△

Rationale:	Basic functionality.
Use Case:	To avoid usage of the module without a complete initialization this could cause the transmission of corrupted Frames.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.3.3 Normal Operation

[SRS_Fr_05090] The FlexRay Transport Layer shall support per connection the ISO 10681-2 / ISO 15765-2 service N_ChangeParameter ⌈

Description:	The FlexRay Transport Layer shall support per connection (i.e. per N-SDU) the ISO 10681-2 / ISO 15765-2 service N_ChangeParameter. Thus, it is possible to adapt the BC to the receiver's needs.
Rationale:	In the case of Inter-ECU communication it could be necessary to adapt the BandwidthControl parameter to the individual needs of the receiver.
Use Case:	–
Dependencies:	–
Supporting Material:	ISO 10681-2 and ISO 15765-2 specifications.

⌋

[SRS_Fr_05093] A cancellation service of transmission shal be provided at any time ⌈

Description:	The FlexRay Transport Layer shall provide to the sender a cancellation service of transmission at any point in time It shall be configurable at system configuration time whether this feature is provided by the FlexRay Transport Layer.
Rationale:	Cancellation of pending transmission.
Use Case:	The higher layer can cancel a pending transmission by using this service.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05095] The FlexRay Transport Layer shall support the dynamic bandwidth control mechanism ⌈

Description:	The FlexRay Transport Layer shall support the dynamic bandwidth control mechanism as described in the ISO 10681-2
Rationale:	Adapting the bandwidth control parameters as used in ISO-TP to FlexRay circumstances.
Use Case:	–
Dependencies:	–
Supporting Material:	ISO10681-2specification.

⌋

4.2.1.4 FlexRay Time Service

4.2.1.4.1 Valid Requirements

[SRS_Fr_05033] Tick Conversion shall be provided ⌈

Description:	The FlexRay software modules shall provide a software interface to carry out a conversion between FlexRay macroticks and nanoseconds in both directions.
Rationale:	The upper-layer SW module does not know the duration of the FlexRay specific macrotick. Therefore, the FlexRay software modules shall provide an API to carry out the conversion.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05053] The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific Cluster ⌈

Description:	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms (according to chapter 8.6.5 of [FR_PROT_SPEC]) to a specific FlexRay Cluster, i.e. to all FlexRay Communication Controllers connected to this Cluster.
Rationale:	–
Use Case:	An application may need to maintain the synchronicity between two FlexRay Clusters.
Dependencies:	[SRS_Fr_05156] Controller External Clock Synchronization
Supporting Material:	–

⌋

[SRS_Fr_05156] The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific CC ⌈

Description:	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms (according to chapter 8.6.5 of [FR_PROT_SPEC]) to a specific FlexRay Communication Controller
Rationale:	–
Use Case:	An application may need to maintain the synchronicity between two FlexRay Clusters.
Dependencies:	[SRS_Fr_05053] Cluster External Clock Synchronization
Supporting Material:	–

⌋

4.2.1.4.2 Configuration

[SRS_Fr_05044] CC's Absolute Timer shall be provided ⌈

Description:	The FlexRay software modules shall provide a software interface to set a FlexRay Communication Controller's absolute timer in order to program a timer interrupt ("alarm") at a defined point in (the FlexRay global) time specified by a pair of cycle time (in units of macroticks) and cycle count.
Rationale:	–
Use Case:	Absolute alarms can be used to synchronize software modules to the global time of a FlexRay node. The scheduler of a time driven operating system can be triggered by an absolute alarm, thereby synchronizing the operating system to the global time of a FlexRay node.
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.4.3 Normal Operation

[SRS_Fr_05046] Absolute Alarms of a CC shall be enabled ⌈

Description:	The FlexRay software modules shall provide a software interface to enable the absolute alarm(s) of a FlexRay Communication Controller.
Rationale:	–
Use Case:	Higher software layers may want to (re)-enable an absolute alarm.
Dependencies:	–

▽

△

Supporting Material:	–

⌋

[SRS_Fr_05047] Absolute Alarms of a CC shall be disabled ⌈

Description:	The FlexRay software modules shall provide a software interface to disable the absolute alarm(s) of a FlexRay Communication Controller.
Rationale:	–
Use Case:	Higher software layers may want to disable an absolute alarm temporarily.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05048] Absolute Alarms of a CC shall be acknowledged ⌈

Description:	The FlexRay software modules shall provide a software interface to acknowledge the absolute alarm(s) of a FlexRay Communication Controller, i.e. to clear the interrupt condition.
Rationale:	If in case of level-sensitive interrupts, an alarm is not acknowledged within the ISR (i.e. the interrupt request flag is not being reset) an interrupt service will be requested immediately again after leaving the ISR.
Use Case:	Higher software layers may use this function to reset the absolute alarm condition.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05049] Relative Alarms of a CC shall be enabled ⌈

Description:	The FlexRay software modules shall provide a software interface to enable the relative alarm(s) of a FlexRay Communication Controller, if supported by the FlexRay CC.
Rationale:	–
Use Case:	Higher software layers may want to (re)-enable a relative alarm.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05050] Relative Alarms of a CC shall be disabled ⌈

Description:	The FlexRay software modules shall provide a software interface to disable the relative alarm(s) of a FlexRay Communication Controller, if supported by the FlexRay CC.
Rationale:	–
Use Case:	Higher software layers may want to disable a relative alarm temporarily.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05051] Relative Alarms of a CC shall be acknowledged ⌈

Description:	The FlexRay software modules shall provide a software interface to acknowledge the relative alarm(s) of a FlexRay Communication Controller, if supported by the FlexRay CC, i.e. to clear the interrupt condition.
Rationale:	If in case of level-sensitive interrupts, an alarm is not acknowledged within the ISR (i.e. the interrupt request flag is not being reset) an interrupt service will be requested immediately again after leaving the ISR.
Use Case:	Higher software layers may use this function to reset the relative alarm condition.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05052] Cycle Length in Macroticks shall be provided ⌈

Description:	The FlexRay software modules shall provide a software interface to get the length of a Communication Cycle (in macroticks) of a specific FlexRay Cluster.
Rationale:	–
Use Case:	This is for example required to set up a (2n Cycles) repetitive relative alarm with the repetition rate of one Cycle.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05019] FlexRay Global Time shall be provided ⌈

Description:	The FlexRay software modules shall provide a software interface to get the FlexRay global time from a specific FlexRay CC, specified by a pair of cycle time (in units of macroticks) and cycle count.
Rationale:	The FlexRay global time may be used for the synchronization of the upper-layer software. It can also be read by an application running synchronously or asynchronously to the FlexRay global time.

▽

△

Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.4.4 Fault Operation

[SRS_Fr_05071] The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is halted or aborted ⌈

Description:	If a time services function is called for a specific FlexRay CC after the communication of this FlexRay CC has been halted or aborted (i.e. the FlexRay CC is in "HALT" state), the FlexRay Driver shall raise an error.
Rationale:	Time service functions only make sense when the FlexRay global time is available, thus when the FlexRay CC is not in "HALT" state.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05072] The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is Out of Sync ⌈

Description:	If a time services function is called for a specific FlexRay CC which is not synchronous to its Cluster, the FlexRay Driver shall raise an error.
Rationale:	Time service functions only make sense when the FlexRay global time is available, thus when the FlexRay CC is synchronous.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.5 FlexRay Transceiver Driver

4.2.1.5.1 Configuration

[SRS_Fr_05131] The transceiver driver package shall include a description file with the basic information needed to configure the driver for a given bus and the supported notifications. ⌈

Description:	This file shall clarify whether the following features are supported: • Single and/or multiple instances of a given transceiver device • Maximally supported baud rate of each bus to enable the detection of configuration errors • Wake-up by bus • Control of power supply possible by the transceiver • List of errors • Transceiver control via SPI or port pin • Which notifications are supported • Call context of the notification functions (ISR, polling) to enable detection of necessary data consistency mechanisms during configuration time • Transceiver supports bus error detection (if not, a state query will always return <good>)
Rationale:	Configuration
Use Case:	Basic functionality for transceiver configuration.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05132] The FlexRay Transceiver Driver shall support the configuration for more than one transceiver type as well as for more than one Cluster ⌈

Description:	The driver shall be able to support multiple FlexRay Clusters on the ECU. It must be possible to configure the used transceiver device independently for each Cluster. This includes also mixed systems with e.g. two FlexRay Clusters using different bus physics.
Rationale:	Systems with two FlexRay Clusters.
Use Case:	Basic functionality for transceiver configuration.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05134] The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack. ⌈

Description:	To start the ECU from power-up or reset, a fix 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 a lot of requirements, partly dependent on the FlexRay controller and the power supply concept.
Rationale:	Correct ECU startup behavior
Use Case:	Basic functionality for transceiver configuration.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05136] The FlexRay Transceiver Driver shall support the compile time configuration of one notification to a higher layer for change notification for "wake-up by bus" events. ⌈

Description:	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.
Rationale:	Efficient coupling between FlexRay Transceiver Driver and higher layer.
Use Case:	See [SRS_Fr_05147]
Dependencies:	[SRS_Fr_05147]
Supporting Material:	–

⌋

4.2.1.5.2 Initialization

[SRS_Fr_05137] The FlexRay Transceiver Driver shall provide an API to initialize the driver internally. ⌈

Description:	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.
Rationale:	Set FlexRay Transceivers and FlexRay Transceiver Driver in a pre-defined and known state
Use Case:	Basic functionality for transceiver control.

▽

△

Dependencies:	SPI and DIO driver initialization. [SRS_Fr_05144] The FlexRay Transceiver Driver setup information must provide the necessary configuration data to enable the generation tool to select the appropriate control mechanism (e.g. SPI, I/O ports) and to guarantee the correct allocation of the necessary communication resources and initialization sequences.
Supporting Material:	–

⌋

4.2.1.5.3 Normal Operation

[SRS_Fr_05138] The FlexRay Transceiver Driver API shall be synchronous. ⌈

Description:	The FlexRay Transceiver Driver API shall execute the requested action immediately and shall deliver the result state immediately to the caller. This will ease up the implementation of wake-up and sleep concepts within the AUTOSAR BSW stack.
Rationale:	Better usage of transceiver functionality in the complex AUTOSAR BSW environment.
Use Case:	Atomic transition to other operation mode; easier and better abstraction for higher layers like the ECU state manager or ComManager. Improved testability compared to asynchronous handling.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05166] It shall be possible to set the FlexRay Transceiver Operation Mode ⌈

Description:	The FlexRay Transceiver Driver shall provide a software interface to set the operation mode of a specific FlexRay Transceiver device. According to [FR_EPL_SPEC], at least these operation modes shall be supported: • BD_Normal • BD_Standby • BD_Receive_only • BD_Sleep
Rationale:	Provide ComM with a possibility to set the operation mode of a specific FlexRay Transceiver device.
Use Case:	–
Dependencies:	[SRS_Fr_05039] Set FlexRay Transceiver Operation Mode

▽

△

Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05167] The FlexRay Transceiver Operation Mode shall be provided ⌈

Description:	The FlexRay Transceiver Driver shall provide a software interface to get the operation mode of a FlexRay Transceiver device. According to [FR_EPL_SPEC], at least these operation modes shall be supported: • BD_Normal • BD_Standby • BD_Receive_only • BD_Sleep
Rationale:	Provide ComM with a possibility to get the operation mode of a specific FlexRay Transceiver device.
Use Case:	–
Dependencies:	[SRS_Fr_05157] Get FlexRay Transceiver Operation Mode
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05144] The FlexRay Transceiver Wake-up Reason shall be provided ⌈

Description:	The FlexRay Transceiver Driver shall provide a software interface to get the wake-up reason of a specific FlexRay Transceiver device. The FlexRay Transceiver Driver shall be able to store the local view "who has requested the wake-up: bus or internally". • Bus: The bus has caused the wake-up. • Internally: The wake-up has been caused by an internal request to the driver. • Sleep: The transceiver is in operation mode sleep and no wake-up has been occurred. • Wake pin: An edge on the wake pin of the transceiver (if present) has caused the wakeup. The wake-up reason should be "sleep" when the operation mode is not Normal and no wake-up has been occurred. When a wake-up has occurred, the API must always return the first detected wake-up reason (e.g. if a wake-up by bus occurs and than nearly at the same time an internal wake-up, the wake-up reason is "bus".). After leaving the operation mode Normal, the wake-up reason shall be set to "sleep" again.

▽

△

Rationale:	Detection of wake-up reason during development and via diagnostic command. May also be used by the NM or ECU state manager.
Use Case:	–
Dependencies:	[SRS_Fr_05158] Get FlexRay Transceiver Wake-up Reason
Supporting Material:	–

⌋

[SRS_Fr_05147] The FlexRay Transceiver Driver shall support a notification to inform higher layers about the wake-up by bus. ⌈

Description:	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!). Due to the delay from wake-up detection till the start of the necessary actions have a large influence to 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 to 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. Assumed 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. See [SRS_Fr_05136] for details, too.
Dependencies:	DIO and SPI driver for notification, one of (bus specific) NM, diagnostics or ECU state manager as client. [SRS_Fr_05136]
Supporting Material:	–

⌋

[SRS_Fr_05148] The FlexRay Transceiver Driver shall support situations where a wake-up by bus occurs at the same moment the transition to standby/sleep is executed by the driver. ⌈

Description:	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.
Rationale:	Safe wake-up and sleep handling.
Use Case:	All busses with a wake-up by bus are affected.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05149] The FlexRay Transceiver Driver shall support an API to enable and disable the wake-up notification for each bus separately. ⌈

Description:	To enable higher layers to command the FlexRay Transceiver device safe into its standby and/or sleep state, an additional API to disable and enable the wake-up notification is necessary. If the notification is disabled, driver shall not perform the notification but store the event internally until the notification is enabled again. The notification shall then be processed immediately. It shall be possible to clear a pending wake-up event. If no further wake-up event occurs, no notification shall be performed after enabling the notification again. If a further wake-up event occurs it shall be notified.
Rationale:	Safe wake-up and sleep handling.
Use Case:	All busses with a wake-up by bus are affected.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05135] Active FlexRay Stars shall be supported ⌈

Description:	Active Stars are part of valid FlexRay topologies.
Use Case:	1. Active stars are required for electrical isolation of Branches. 2. Device drivers should read out status information to support diagnostics and selective control (and possibly enabling and disabling) of transceivers.
Supporting Material:	–

⌋

4.2.1.5.4 Shutdown Operation

[SRS_Fr_05150] The FlexRay Transceiver Driver shall support the AUTOSAR ECU state manager in a way that a safe system startup and shutdown is possible. ⌈

Description:	The ECU state manager is under development in parallel to this specification, therefore this requirement is added to check during implementation and system test for correct interaction between the FlexRay Transceiver Driver and the ECU state manager. In valid, for startup the FlexRay Transceivers must not be enabled until the power supply is available and stable to prevent errors on the bus. Also the communication hardware and driver must not be enabled until the transceiver is configured into its normal operation mode. For shutdown, the communication must be stopped according to the AUTOSAR NM algorithm, the FlexRay drivers must be stopped and then the transceivers may be set to standby/sleep, too. The correct sequence depends on the wake-up/sleep concept of AUTOSAR.
Rationale:	System startup and shutdown together with ECU state manager.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

4.2.1.5.5 Fault Operation

[SRS_Fr_05151] The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness. ⌈

Description:	Depending on the supported transceiver device, the driver shall check the correctness of the executed control communication and the operation mode a transceiver is in.
Rationale:	Diagnostics and trouble shooting
Use Case:	1. Detection of defect or misbehaving transceiver hardware 2. Detection of corrupted SPI communication The check shall only be applied 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. "Errors" caused by the "outer world" (e.g. disturbed communication Channels or ground offsets) are not in the scope of this API.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05168] FlexRay Transceiver Error State shall be indicated (modify according to Monitoring Concept and Concept Reliability) ⌈

Description:	The FlexRay Transceiver Driver shall indicate error states of a FlexRay Transceiver device to the DEM.
Rationale:	The Transceiver error state is necessary for error handling.
Use Case:	–
Dependencies:	–
Supporting Material:	[FR_EPL_SPEC]

⌋

[SRS_Fr_05212] The Errors in Bus Driver shall be detected and notified ⌈

Description:	The FlexRay Transceiver Driver shall provide an API that detects errors in bus driver and notify them to application level.
Rationale:	The FlexRay modules should provide information that only the modules can detect.
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05213] The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly ⌈

Description:	The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly.
Rationale:	This functionality ensures that the hardware is working as expected.
Use Case:	Improvement of hardware reliability.
Dependencies:	–
Supporting Material:	–

⌋

[SRS_Fr_05214] FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality ⌈

Description:	FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality
Rationale:	When trouble occurs in the hardware level, it's likely to fix the cause by resetting the hardware.
Use Case:	This function shall be executed when so many errors are detected in FlexRay modules.
Dependencies:	–

▽

△

Supporting Material:	–

⌋

4.3 Non-Functional Requirements (Qualities)

4.3.1 FlexRay Interface

4.3.1.1 Abstraction Requirements

[SRS_Fr_05006] Abstraction of FlexRay-Specific Features shall be provided ⌈

Description:	For data reception and sending, the FlexRay Interface shall provide to the upper software layers an abstraction from the specific features of the FlexRay communication system as well as a PDU-based API.
Rationale:	All bus interfaces supported by the AUTOSAR BSW (e.g. CAN, FlexRay, LIN) should provide the same API semantics/signature.
Use Case:	Several bus interfaces (e.g. CAN, FlexRay, and LIN) on one ECU.
Dependencies:	–
Supporting Material:	–

⌋

4.3.1.2 Resource Usage

[SRS_Fr_05009] The FlexRay Interface shall allocate the needed memory space only once for a PDU sent multiple times in the FlexRay matrix ⌈

Description:	The FlexRay Interface shall allocate the needed memory space only once for a PDU sent multiple times in the FlexRay matrix.
Rationale:	The use of only one local memory space for each PDU, instead of one local memory space for each occurrence of the PDU in the FlexRay matrix, implicates a much better performance.
Use Case:	–
Dependencies:	[SRS_Fr_05004] PDU-Based API
Supporting Material:	–

⌋

4.3.1.3 Configuration

[SRS_Fr_05056] Configuration of the FlexRay Interface shall be done at System Configuration Time ⌈

Description:	The FlexRay Interface configuration shall be done at system configuration time, i.e. the configuration parameters have to be defined before the execution of the FlexRay Interface code (even prior to executing the initialization code).
Rationale:	The purpose of the FlexRay Interface is to execute data processing that has been configured at system configuration time, i.e. before the execution of the FlexRay Interface code. It shall not decide or calculate during runtime any communication parameters like the assignment of Slots to ECU or the packaging of PDUs into FlexRay Frames.
Use Case:	–
Dependencies:	[SRS_Fr_05042] Switch Configuration during Runtime
Supporting Material:	–

⌋

4.3.2 Transport Layer FlexRay

[SRS_Fr_05172] ISO 10681-2 or ISO 15765-2 or both Specifications shall be selected ⌈

Description:	It shall be possible to select at pre-compile time which AUTOSAR Transport Protocol Module shall be used. Either the AUTOSAR FlexRay Transport Layer compliant with ISO 10681-2 specification shall be selected (i.e. FlexRay ISO TP), or the AUTOSAR FlexRay Transport Layer compliant to ISO 15765-2 regarding PCI layout shall be selected (i.e. FlexRay AUTOSAR TP), or both protocol shall be selected.(i.e using different sets of PDU#s)
Rationale:	Reuse of existing standards for AUTOSAR BSW.
Use Case:	Flashing, ECU diagnosis and transmission of large data blocks
Dependencies:	[SRS_Fr_05073]
Supporting Material:	[ISO 15765-2] and ISO 10681-2 specifications.

⌋

[SRS_Fr_05098] ISO 15765-2 specifications shall be used ⌈

Description:	It shall be possible to select the AUTOSAR FlexRay Transport Layer pre-compile time to be compliant with ISO 15765-2 regarding the PCI layout. It shall be possible to configure this independently for each concurrently active connection (i.e. for each N-SDU).

▽

△

Rationale:	Reuse of existing standards for AUTOSAR BSW. The ISO 15765-2 specifications are the mostly used Transport Layer in automotive area. Although ISO 15765-2 have several drawbacks from the functional point of view (e.g., no acknowledgement on the last block of data), ISO compliance is an important issue for the sake of interoperability with legacy ECUs. - Thus the AUTOSAR FlexRay Transport Layer shall at least provide an ISO compliant mode that can be enforced at system configuration time via a configuration switch.
Use Case:	–
Dependencies:	[SRS_Fr_05073] shall be enabled per ECU at the same time.
Supporting Material:	[ISO 15765-2] specification.

⌋

[SRS_Fr_05073] The FlexRay Transport Layer shall be configured to be compliant with the ISO 10681-2 specification ⌈

Description:	It shall be possible to configure the AUTOSAR FlexRay Transport Layer at system configuration time to be compliant with the ISO 10681-2 specification per ECU
Rationale:	Usage of an ISO standard for Flexray
Use Case:	–
Dependencies:	Either [SRS_Fr_05073] or [SRS_Fr_05098] shall be enabled at the same time.
Supporting Material:	[ISO 15765-2] and ISO 10681-2 specifications.

⌋

[SRS_Fr_05074] The FlexRay Transport Layer software module shall be located between the PDU Router and the FlexRay Interface ⌈

Description:	The FlexRay Transport Layer software module shall be located between the PDU Router and the FlexRay Interface for FlexRay PDUs requiring Transport Protocol functionalities. The FlexRay Transport Layer is used by the PDU Router to transmit and receive FlexRay PDUs coming from the Diagnostic Communication Manager (or from a SW-Component).
Rationale:	Design of AUTOSAR Software Architecture.
Use Case:	–
Dependencies:	–
Supporting Material:	AUTOSAR Layered Software Architecture [LSA]

⌋

[SRS_Fr_05075] The FlexRay Transport Layer implementation shall be independent of the network configuration ⌈

Description:	The FlexRay Transport Layer implementation shall be independent of the network configuration represented by the FlexRay communication matrix. The API just deals with universal identifiers and data units (N-SDU) properties.
Rationale:	Design of AUTOSAR Software Architecture.
Use Case:	–
Dependencies:	–
Supporting Material:	AUTOSAR Layered Software Architecture [LSA]

⌋

[SRS_Fr_05123] The Configuration shall be modifiable by a Flashing Process ⌈

Description:	All post-builld configuration parameters of the FlexRay Transport Layer, defined at system configuration time shall be modifiable by a flashing process. This encompasses (amongst other configuration items) all connection-specific parameters and also all global parameters required for the configuration of the FlexRay Transport Layer.
Rationale:	Change of FlexRay Transport Layer parameters without recompiling of the FlexRay Transport Layer software.
Use Case:	–
Dependencies:	[SRS_Fr_05034] Configuration Modifiable by a Flashing Process
Supporting Material:	–

⌋

[SRS_Fr_05076] The FlexRay Transport Layer shall support at least 32 logical FlexRay Transport Layer active connections being used concurrently ⌈

Description:	The FlexRay Transport Layer shall support at least 32 logical FlexRay Transport Layer active connections being used concurrently. The number of logical active connections shall be configurable at system configuration time.
Rationale:	Enable multiple concurrent FlexRay Transport Layer connections.
Use Case:	The FlexRay Transport Layer configuration may be specific for each Transport Layer connection.
Dependencies:	–
Supporting Material:	–

⌋

4.3.3 FlexRay Transceiver Driver

The FlexRay Transceiver Driver has strong relations to the AUTOSAR NM and to the AUTOSAR ECU state manager since they control and use this BSW component. The

requirements and needs of these two BSW components may have strong influence on this specification and maybe vice versa.

The FlexRay Transceiver Driver itself will use the drivers for ports (DIO) or SPI. Therefore, the FlexRay Transceiver Driver will be a user for their services as it is. If the FlexRay Transceiver Driver uses other drivers for e.g. initialization access, they must be available before the FlexRay Transceiver initialization is executed!

4.3.3.1 Timing Requirements

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

Description:	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. Small times may be implemented as a wait loop inside the driver. Disadvantages are that this time is lost for the other software and the wait time depends on the used μC and e.g. system clock. Large wait times (e.g. >200μs) may require an asynchronous API of the FlexRay Transceiver Driver. Disadvantage is then that the complete API and usage will be different for such a hardware device.
Rationale:	Correct handling of used transceiver
Use Case:	–
Dependencies:	–
Supporting Material:	–

⌋

5 References

[1] Standardization Template
AUTOSAR_FO_TPS_StandardizationTemplate

[2] Glossary
AUTOSAR_FO_TR_Glossary

[3] FlexRay Protocol Specification
http://www.flexray.com

A Change history of AUTOSAR traceable items

A.1 Traceable item history of this document according to AUTOSAR Release R22-11

A.1.1 Added Traceables in R22-11

Number	Heading
[SRS_Fr_05000]	Synchronous SW Modules shall be supported
[SRS_Fr_05001]	Asynchronous SW Modules shall be supported
[SRS_Fr_05002]	FlexRay Interface and FlexRay Driver shall operated synchronized to the global time
[SRS_Fr_05003]	Slot/Cycle Multiplexing shall be supported
[SRS_Fr_05004]	The FlexRay Interface shall provide a PDU-based data API to all upper layers.
[SRS_Fr_05005]	The CC Hardware FIFO Mechanism shall be supported
[SRS_Fr_05006]	Abstraction of FlexRay-Specific Features shall be provided
[SRS_Fr_05007]	The FlexRay Interface shall be able to communicate with at least four FlexRay CCs via the appropriate FlexRay Driver(s)
[SRS_Fr_05008]	FlexRay 2.1 and 3.0 Hardware shall be supported
[SRS_Fr_05009]	The FlexRay Interface shall allocate the needed memory space only once for a PDU sent multiple times in the FlexRay matrix
[SRS_Fr_05010]	Each PDU shall have one PDU-ID
[SRS_Fr_05011]	Initialization of the Low-Level Parameters shall be available
[SRS_Fr_05012]	Initialization of the FlexRay CC Transmit/Receive Buffers shall be available
[SRS_Fr_05013]	The local Memory Space shall be initialized
[SRS_Fr_05015]	The FlexRay Interface shall provide a software interface to start-up a specific FlexRay CC
[SRS_Fr_05016]	A FlexRay CC Communication shall be aborted when wanted
[SRS_Fr_05018]	The FlexRay Interface shall provide a software interface to send a wake-up pattern on a channel or CC
[SRS_Fr_05019]	FlexRay Global Time shall be provided
[SRS_Fr_05022]	FlexRay CC POC Status shall be available
[SRS_Fr_05024]	The software interface of the Driver shall be independent of the CC buffers' configuration
[SRS_Fr_05027]	A PDU shall be transmitted via the FlexRay communication system
[SRS_Fr_05031]	A FlexRay CC shall be initialized and configured
[SRS_Fr_05033]	Tick Conversion shall be provided
[SRS_Fr_05034]	The configuration data shall be modifiable by a Flashing Process
[SRS_Fr_05039]	The Operation Mode of a FlexRay Transceiver shall be set
[SRS_Fr_05042]	The FlexRay Interface shall allow switching from one configuration to another one in Normal Active Mode

▽

△

Number	Heading
[SRS_Fr_05044]	CC's Absolute Timer shall be provided
[SRS_Fr_05046]	Absolute Alarms of a CC shall be enabled
[SRS_Fr_05047]	Absolute Alarms of a CC shall be disabled
[SRS_Fr_05048]	Absolute Alarms of a CC shall be acknowledged
[SRS_Fr_05049]	Relative Alarms of a CC shall be enabled
[SRS_Fr_05050]	Relative Alarms of a CC shall be disabled
[SRS_Fr_05051]	Relative Alarms of a CC shall be acknowledged
[SRS_Fr_05052]	Cycle Length in Macroticks shall be provided
[SRS_Fr_05053]	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific Cluster
[SRS_Fr_05055]	Timer Interrupts during Shutdown shall be avoided
[SRS_Fr_05056]	Configuration of the FlexRay Interface shall be done at System Configuration Time
[SRS_Fr_05058]	The configuration of the FlexRay Driver shall be defined at system configuration time.
[SRS_Fr_05059]	The Driver shall be configure the CC's transmit/receive buffers
[SRS_Fr_05060]	Scheduling of Copy Operation into/from FlexRay CC shall be possible
[SRS_Fr_05061]	NM Vector feature shall be supported
[SRS_Fr_05063]	A FlexRay CC Communication shall be halted when wanted
[SRS_Fr_05064]	Abstraction of FlexRay CC-specific Implementation shall be provided
[SRS_Fr_05065]	The FlexRay Driver shall be able to communicate with at least four FlexRay CCs of the same type
[SRS_Fr_05066]	L-SDU-Based API shall be available
[SRS_Fr_05071]	The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is halted or aborted
[SRS_Fr_05072]	The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is Out of Sync
[SRS_Fr_05073]	The FlexRay Transport Layer shall be configured to be compliant with the ISO 10681-2 specification
[SRS_Fr_05074]	The FlexRay Transport Layer software module shall be located between the PDU Router and the FlexRay Interface
[SRS_Fr_05075]	The FlexRay Transport Layer implementation shall be independent of the network configuration
[SRS_Fr_05076]	The FlexRay Transport Layer shall support at least 32 logical FlexRay Transport Layer active connections being used concurrently
[SRS_Fr_05077]	Each N-SDU shall have a unique identifier
[SRS_Fr_05088]	FlexRay Transport Layer's variables shall be initialized
[SRS_Fr_05089]	The FlexRay Transport Layer services shall not be operational before initializing the module.
[SRS_Fr_05090]	The FlexRay Transport Layer shall support per connection the ISO 10681-2 / ISO 15765-2 service N_ChangeParameter

▽

△

Number	Heading
[SRS_Fr_05093]	A cancellation service of transmission shal be provided at any time
[SRS_Fr_05095]	The FlexRay Transport Layer shall support the dynamic bandwidth control mechanism
[SRS_Fr_05096]	Communication controllers shall be assigned to FlexRay Driver.
[SRS_Fr_05097]	The FlexRay Interface shall be able to communicate with at least four FlexRay Drivers
[SRS_Fr_05098]	ISO 15765-2 specifications shall be used
[SRS_Fr_05106]	The Buffer of a specific CC in Normal Active Mode shall be reconfigurable
[SRS_Fr_05109]	The FlexRay Driver shall provide a software interface to start-up a specific FlexRay CC
[SRS_Fr_05114]	A FlexRay CC Communication shall be aborted when wanted
[SRS_Fr_05115]	The FlexRay CC Communication shall be halted when wanted
[SRS_Fr_05116]	Initialization of FlexRay CC shall be available
[SRS_Fr_05117]	A Wake-Up Pattern shall be sent on a specific channel of a CC
[SRS_Fr_05120]	FlexRay CC POC Status shall be provided
[SRS_Fr_05121]	FlexRay CC Sync State shall be provided
[SRS_Fr_05123]	The Configuration shall be modifiable by a Flashing Process
[SRS_Fr_05125]	The FlexRay Driver shall provide services to handle interrupts of a FlexRay Communication Controller.
[SRS_Fr_05126]	PDU Update/Valid Information shall be handled
[SRS_Fr_05130]	The FlexRay Interface shall support PDU transmission buffer queues
[SRS_Fr_05131]	The transceiver driver package shall include a description file with the basic information needed to configure the driver for a given bus and the supported notifications.
[SRS_Fr_05132]	The FlexRay Transceiver Driver shall support the configuration for more than one transceiver type as well as for more than one Cluster
[SRS_Fr_05134]	The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack.
[SRS_Fr_05135]	Active FlexRay Stars shall be supported
[SRS_Fr_05136]	The FlexRay Transceiver Driver shall support the compile time configuration of one notification to a higher layer for change notification for "wake-up by bus" events.
[SRS_Fr_05137]	The FlexRay Transceiver Driver shall provide an API to initialize the driver internally.
[SRS_Fr_05138]	The FlexRay Transceiver Driver API shall be synchronous.
[SRS_Fr_05144]	The FlexRay Transceiver Wake-up Reason shall be provided
[SRS_Fr_05147]	The FlexRay Transceiver Driver shall support a notification to inform higher layers about the wake-up by bus.
[SRS_Fr_05148]	The FlexRay Transceiver Driver shall support situations where a wake-up by bus occurs at the same moment the transition to standby/sleep is executed by the driver.

▽

△

Number	Heading
[SRS_Fr_05149]	The FlexRay Transceiver Driver shall support an API to enable and disable the wake-up notification for each bus separately.
[SRS_Fr_05150]	The FlexRay Transceiver Driver shall support the AUTOSAR ECU state manager in a way that a safe system startup and shutdown is possible.
[SRS_Fr_05151]	The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness.
[SRS_Fr_05152]	The FlexRay Transceiver Driver shall handle the transceiver-specific timing requirements internally.
[SRS_Fr_05156]	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific CC
[SRS_Fr_05157]	The Operation Mode of a FlexRay Transceiver shall be available
[SRS_Fr_05158]	The wake-up reason of a specific FlexRay Transceiver device shall be available
[SRS_Fr_05159]	The FlexRay Interface shall provide a software interface to enable the wake-up indication of a specific FlexRay Transceiver device
[SRS_Fr_05160]	The FlexRay Interface shall provide a software interface to disable the wake-up indication of a specific FlexRay Transceiver device.
[SRS_Fr_05161]	Pending Wake-up Events of a Transceiver shall be cleared if necessary
[SRS_Fr_05166]	It shall be possible to set the FlexRay Transceiver Operation Mode
[SRS_Fr_05167]	The FlexRay Transceiver Operation Mode shall be provided
[SRS_Fr_05168]	FlexRay Transceiver Error State shall be indicated (modify according to Monitoring Concept and Concept Reliability)
[SRS_Fr_05169]	Timer Interrupts during Start-up shall be avoided
[SRS_Fr_05170]	PDUs received via the FlexRay communication system shall be retrieved
[SRS_Fr_05171]	A PDU Transmit Confirmation shall be provided
[SRS_Fr_05172]	ISO 10681-2 or ISO 15765-2 or both Specifications shall be selected
[SRS_Fr_05174]	The FlexRay Interface shall provide services to handle interrupts of a FlexRay Communication Controller.
[SRS_Fr_05175]	The Error Informations shall be provided
[SRS_Fr_05200]	The Error Information in Transmitted Frame shall be available
[SRS_Fr_05201]	The Error Information in Received Frame shall be available
[SRS_Fr_05202]	The Error Information in NIT shall be available
[SRS_Fr_05203]	The Error Information in Bus Driver shall be available
[SRS_Fr_05204]	A Checking Function of L-PDU Length shall be customizabled
[SRS_Fr_05205]	The FlexRay Driver shall provide an API that cancels transmit request
[SRS_Fr_05206]	FlexRay Driver shall provide a method that reinitializes CC's functionality
[SRS_Fr_05207]	An API to Detect Error Information in Transmitted Frame shal be provided
[SRS_Fr_05208]	An API to Detect Error Information in Received Frame shall be provided
[SRS_Fr_05209]	An API to Detect Error Information in NIT shall be provided
[SRS_Fr_05210]	Criteria on Validness of Received Frame shall be provided

▽

△

Number	Heading
[SRS_Fr_05211]	Hardware Checking Function on CC shall be available
[SRS_Fr_05212]	The Errors in Bus Driver shall be detected and notified
[SRS_Fr_05213]	The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly
[SRS_Fr_05214]	FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality
[SRS_Fr_05215]	The FlexRay Interface shall provide an API that cancels transmit request
[SRS_Fr_05216]	An API to query the FlexRay Rate and Offset Correction shall be created
[SRS_Fr_05217]	An API to report FlexRay Status Data for number of Sync Frames shall be created
[SRS_Fr_05218]	An API to report FlexRay Status Data for list of Sync Frame IDs shall be created
[SRS_Fr_05219]	FlexRay Key Slot Mode shall be supported
[SRS_Fr_05220]	The buffer shall be immediatly accessible for read operations
[SRS_Fr_05221]	The Job list Execution shall be triggered by a synchronized task
[SRS_Fr_05222]	The Transceiver Rx-Only Mode via FR State Manager shall be supported
[SRS_Fr_05223]	The number of Startup Frames shall be available
[SRS_Fr_05224]	Payload Preamble Indicator Bit shall be supported
[SRS_Fr_05225]	The CHI Command shall be called at any time
[SRS_Fr_05226]	Transport Connection Properties "ISO 10681-2"
[SRS_Fr_15006]	Hardware Message ID Filter Mechanism according to the FlexRay Protocol Specification 3.0 shall be supported
[SRS_Fr_15007]	Time Triggered Master mode provided by FlexRay 3.0 Communication Controller shall be available
[SRS_Fr_15009]	Dynamic LPdu payload length shall be supported
[SRS_Fr_15060]	FlexRay Network Management Scheduling Timing Window Relief shall be available
[SRS_Fr_15106]	A CC Buffer reconfiguration shall be possible at runtime

Table A.1: Added Traceables in R22-11

A.1.2 Changed Traceables in R22-11

none

A.1.3 Deleted Traceables in R22-11

none

A.2 Traceable item history of this document according to AUTOSAR Release R23-11

A.2.1 Added Requirements in R23-11

Number	Heading
[SRS_Fr_05000]	Synchronous SW Modules shall be supported
[SRS_Fr_05001]	Asynchronous SW Modules shall be supported
[SRS_Fr_05002]	FlexRay Interface and FlexRay Driver shall operated synchronized to the global time
[SRS_Fr_05003]	Slot/Cycle Multiplexing shall be supported
[SRS_Fr_05004]	The FlexRay Interface shall provide a PDU-based data API to all upper layers.
[SRS_Fr_05005]	The CC Hardware FIFO Mechanism shall be supported
[SRS_Fr_05006]	Abstraction of FlexRay-Specific Features shall be provided
[SRS_Fr_05007]	The FlexRay Interface shall be able to communicate with at least four FlexRay CCs via the appropriate FlexRay Driver(s)
[SRS_Fr_05008]	FlexRay 2.1 and 3.0 Hardware shall be supported
[SRS_Fr_05009]	The FlexRay Interface shall allocate the needed memory space only once for a PDU sent multiple times in the FlexRay matrix
[SRS_Fr_05010]	Each PDU shall have one PDU-ID
[SRS_Fr_05011]	Initialization of the Low-Level Parameters shall be available
[SRS_Fr_05012]	Initialization of the FlexRay CC Transmit/Receive Buffers shall be available
[SRS_Fr_05013]	The local Memory Space shall be initialized
[SRS_Fr_05015]	The FlexRay Interface shall provide a software interface to start-up a specific FlexRay CC
[SRS_Fr_05016]	A FlexRay CC Communication shall be aborted when wanted
[SRS_Fr_05018]	The FlexRay Interface shall provide a software interface to send a wake-up pattern on a channel or CC
[SRS_Fr_05019]	FlexRay Global Time shall be provided
[SRS_Fr_05022]	FlexRay CC POC Status shall be available
[SRS_Fr_05024]	The software interface of the Driver shall be independent of the CC buffers' configuration
[SRS_Fr_05027]	A PDU shall be transmitted via the FlexRay communication system
[SRS_Fr_05031]	A FlexRay CC shall be initialized and configured
[SRS_Fr_05033]	Tick Conversion shall be provided
[SRS_Fr_05034]	The configuration data shall be modifiable by a Flashing Process
[SRS_Fr_05039]	The Operation Mode of a FlexRay Transceiver shall be set
[SRS_Fr_05042]	The FlexRay Interface shall allow switching from one configuration to another one in Normal Active Mode
[SRS_Fr_05044]	CC's Absolute Timer shall be provided

△

Number	Heading
[SRS_Fr_05046]	Absolute Alarms of a CC shall be enabled
[SRS_Fr_05047]	Absolute Alarms of a CC shall be disabled
[SRS_Fr_05048]	Absolute Alarms of a CC shall be acknowledged
[SRS_Fr_05049]	Relative Alarms of a CC shall be enabled
[SRS_Fr_05050]	Relative Alarms of a CC shall be disabled
[SRS_Fr_05051]	Relative Alarms of a CC shall be acknowledged
[SRS_Fr_05052]	Cycle Length in Macroticks shall be provided
[SRS_Fr_05053]	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific Cluster
[SRS_Fr_05055]	Timer Interrupts during Shutdown shall be avoided
[SRS_Fr_05056]	Configuration of the FlexRay Interface shall be done at System Configuration Time
[SRS_Fr_05058]	The configuration of the FlexRay Driver shall be defined at system configuration time.
[SRS_Fr_05059]	The Driver shall be configure the CC's transmit/receive buffers
[SRS_Fr_05060]	Scheduling of Copy Operation into/from FlexRay CC shall be possible
[SRS_Fr_05061]	NM Vector feature shall be supported
[SRS_Fr_05063]	A FlexRay CC Communication shall be halted when wanted
[SRS_Fr_05064]	Abstraction of FlexRay CC-specific Implementation shall be provided
[SRS_Fr_05065]	The FlexRay Driver shall be able to communicate with at least four FlexRay CCs of the same type
[SRS_Fr_05066]	L-SDU-Based API shall be available
[SRS_Fr_05071]	The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is halted or aborted
[SRS_Fr_05072]	The FlexRay Driver shall raise an error if the FlexRay Time Services function is called after the communication of the CC is Out of Sync
[SRS_Fr_05073]	The FlexRay Transport Layer shall be configured to be compliant with the ISO 10681-2 specification
[SRS_Fr_05074]	The FlexRay Transport Layer software module shall be located between the PDU Router and the FlexRay Interface
[SRS_Fr_05075]	The FlexRay Transport Layer implementation shall be independent of the network configuration
[SRS_Fr_05076]	The FlexRay Transport Layer shall support at least 32 logical FlexRay Transport Layer active connections being used concurrently
[SRS_Fr_05077]	Each N-SDU shall have a unique identifier
[SRS_Fr_05088]	FlexRay Transport Layer's variables shall be initialized
[SRS_Fr_05089]	The FlexRay Transport Layer services shall not be operational before initializing the module.
[SRS_Fr_05090]	The FlexRay Transport Layer shall support per connection the ISO 10681-2 / ISO 15765-2 service N_ChangeParameter
[SRS_Fr_05093]	A cancellation service of transmission shal be provided at any time

▽

△

Number	Heading
[SRS_Fr_05095]	The FlexRay Transport Layer shall support the dynamic bandwidth control mechanism
[SRS_Fr_05096]	Communication controllers shall be assigned to FlexRay Driver.
[SRS_Fr_05097]	The FlexRay Interface shall be able to communicate with at least four FlexRay Drivers
[SRS_Fr_05098]	ISO 15765-2 specifications shall be used
[SRS_Fr_05106]	The Buffer of a specific CC in Normal Active Mode shall be reconfigurable
[SRS_Fr_05109]	The FlexRay Driver shall provide a software interface to start-up a specific FlexRay CC
[SRS_Fr_05114]	A FlexRay CC Communication shall be aborted when wanted
[SRS_Fr_05115]	The FlexRay CC Communication shall be halted when wanted
[SRS_Fr_05116]	Initialization of FlexRay CC shall be available
[SRS_Fr_05117]	A Wake-Up Pattern shall be sent on a specific channel of a CC
[SRS_Fr_05120]	FlexRay CC POC Status shall be provided
[SRS_Fr_05121]	FlexRay CC Sync State shall be provided
[SRS_Fr_05123]	The Configuration shall be modifiable by a Flashing Process
[SRS_Fr_05125]	The FlexRay Driver shall provide services to handle interrupts of a FlexRay Communication Controller.
[SRS_Fr_05126]	PDU Update/Valid Information shall be handled
[SRS_Fr_05130]	The FlexRay Interface shall support PDU transmission buffer queues
[SRS_Fr_05131]	The transceiver driver package shall include a description file with the basic information needed to configure the driver for a given bus and the supported notifications.
[SRS_Fr_05132]	The FlexRay Transceiver Driver shall support the configuration for more than one transceiver type as well as for more than one Cluster
[SRS_Fr_05134]	The FlexRay Transceiver Driver shall support the configuration sequence of the AUTOSAR stack.
[SRS_Fr_05135]	Active FlexRay Stars shall be supported
[SRS_Fr_05136]	The FlexRay Transceiver Driver shall support the compile time configuration of one notification to a higher layer for change notification for "wake-up by bus" events.
[SRS_Fr_05137]	The FlexRay Transceiver Driver shall provide an API to initialize the driver internally.
[SRS_Fr_05138]	The FlexRay Transceiver Driver API shall be synchronous.
[SRS_Fr_05144]	The FlexRay Transceiver Wake-up Reason shall be provided
[SRS_Fr_05147]	The FlexRay Transceiver Driver shall support a notification to inform higher layers about the wake-up by bus.
[SRS_Fr_05148]	The FlexRay Transceiver Driver shall support situations where a wake-up by bus occurs at the same moment the transition to standby/sleep is executed by the driver.
[SRS_Fr_05149]	The FlexRay Transceiver Driver shall support an API to enable and disable the wake-up notification for each bus separately.

▽

△

Number	Heading
[SRS_Fr_05150]	The FlexRay Transceiver Driver shall support the AUTOSAR ECU state manager in a way that a safe system startup and shutdown is possible.
[SRS_Fr_05151]	The FlexRay Transceiver Driver shall check the control communication to the transceiver and the reaction of the transceiver for correctness.
[SRS_Fr_05152]	The FlexRay Transceiver Driver shall handle the transceiver-specific timing requirements internally.
[SRS_Fr_05156]	The FlexRay software modules shall provide a software interface to apply rate and offset correction terms to a specific CC
[SRS_Fr_05157]	The Operation Mode of a FlexRay Transceiver shall be available
[SRS_Fr_05158]	The wake-up reason of a specific FlexRay Transceiver device shall be available
[SRS_Fr_05159]	The FlexRay Interface shall provide a software interface to enable the wake-up indication of a specific FlexRay Transceiver device
[SRS_Fr_05160]	The FlexRay Interface shall provide a software interface to disable the wake-up indication of a specific FlexRay Transceiver device.
[SRS_Fr_05161]	Pending Wake-up Events of a Transceiver shall be cleared if necessary
[SRS_Fr_05166]	It shall be possible to set the FlexRay Transceiver Operation Mode
[SRS_Fr_05167]	The FlexRay Transceiver Operation Mode shall be provided
[SRS_Fr_05168]	FlexRay Transceiver Error State shall be indicated (modify according to Monitoring Concept and Concept Reliability)
[SRS_Fr_05169]	Timer Interrupts during Start-up shall be avoided
[SRS_Fr_05170]	PDUs received via the FlexRay communication system shall be retrieved
[SRS_Fr_05171]	A PDU Transmit Confirmation shall be provided
[SRS_Fr_05172]	ISO 10681-2 or ISO 15765-2 or both Specifications shall be selected
[SRS_Fr_05174]	The FlexRay Interface shall provide services to handle interrupts of a FlexRay Communication Controller.
[SRS_Fr_05175]	The Error Informations shall be provided
[SRS_Fr_05200]	The Error Information in Transmitted Frame shall be available
[SRS_Fr_05201]	The Error Information in Received Frame shall be available
[SRS_Fr_05202]	The Error Information in NIT shall be available
[SRS_Fr_05203]	The Error Information in Bus Driver shall be available
[SRS_Fr_05204]	A Checking Function of L-PDU Length shall be customizabled
[SRS_Fr_05205]	The FlexRay Driver shall provide an API that cancels transmit request
[SRS_Fr_05206]	FlexRay Driver shall provide a method that reinitializes CC's functionality
[SRS_Fr_05207]	An API to Detect Error Information in Transmitted Frame shal be provided
[SRS_Fr_05208]	An API to Detect Error Information in Received Frame shall be provided
[SRS_Fr_05209]	An API to Detect Error Information in NIT shall be provided
[SRS_Fr_05210]	Criteria on Validness of Received Frame shall be provided
[SRS_Fr_05211]	Hardware Checking Function on CC shall be available
[SRS_Fr_05212]	The Errors in Bus Driver shall be detected and notified

▽

△

Number	Heading
[SRS_Fr_05213]	The FlexRay Transceiver Driver's initialization function shall check error status in BD to ensure the hardware is working properly
[SRS_Fr_05214]	FlexRay Transceiver Driver shall provide a method that reinitializes BD's functionality
[SRS_Fr_05215]	The FlexRay Interface shall provide an API that cancels transmit request
[SRS_Fr_05216]	An API to query the FlexRay Rate and Offset Correction shall be created
[SRS_Fr_05217]	An API to report FlexRay Status Data for number of Sync Frames shall be created
[SRS_Fr_05218]	An API to report FlexRay Status Data for list of Sync Frame IDs shall be created
[SRS_Fr_05219]	FlexRay Key Slot Mode shall be supported
[SRS_Fr_05220]	The buffer shall be immediatly accessible for read operations
[SRS_Fr_05221]	The Job list Execution shall be triggered by a synchronized task
[SRS_Fr_05222]	The Transceiver Rx-Only Mode via FR State Manager shall be supported
[SRS_Fr_05223]	The number of Startup Frames shall be available
[SRS_Fr_05224]	Payload Preamble Indicator Bit shall be supported
[SRS_Fr_05225]	The CHI Command shall be called at any time
[SRS_Fr_05226]	Transport Connection Properties "ISO 10681-2"
[SRS_Fr_15006]	Hardware Message ID Filter Mechanism according to the FlexRay Protocol Specification 3.0 shall be supported
[SRS_Fr_15007]	Time Triggered Master mode provided by FlexRay 3.0 Communication Controller shall be available
[SRS_Fr_15009]	Dynamic LPdu payload length shall be supported
[SRS_Fr_15060]	FlexRay Network Management Scheduling Timing Window Relief shall be available
[SRS_Fr_15106]	A CC Buffer reconfiguration shall be possible at runtime

Table A.2: Added Requirements in R23-11

A.2.2 Changed Requirements in R23-11

none

A.2.3 Deleted Requirements in R23-11

none

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

A.3.1 Added Requirements in R24-11

none

A.3.2 Changed Requirements in R24-11

Number	Heading
[SRS_Fr_05003]	Slot/Cycle Multiplexing shall be supported
[SRS_Fr_05076]	The FlexRay Transport Layer shall support at least 32 logical FlexRay Transport Layer active connections being used concurrently

Table A.3: Changed Requirements in R24-11

A.3.3 Deleted Requirements in R24-11

none

A.4 Traceable item history of this document according to AUTOSAR Release R25-11

A.4.1 Added Requirements in R25-11

none

A.4.2 Changed Requirements in R25-11

none

A.4.3 Deleted Requirements in R25-11

none

A.5 Traceable item history of this document according to AUTOSAR Release R26-11

A.5.1 Added Requirements in R26-11

none

A.5.2 Changed Requirements in R26-11

none

A.5.3 Deleted Requirements in R26-11

none