Document Title	Specification of FlexRay State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	254
Document Classification	Standard

Document Version	1.1.0
Document Status	Final
Part of Release	3.1
Revision	5

Document Change History

Date	Version	Changed by	Change Description
15.09.2010	1.1.0	AUTOSAR Administration	• Added notification for FrNm in case of a long term synchronization loss • StartupRepetitions made optional to allow for unlimited repetition of startup • Introduction of CANSM_RX_PDU_INIT and CANSM_TX_PDU_INIT, update of Com_IpduGroupStart • Legal disclaimer revised
23.06.2008	1.0.2	AUTOSAR Administration	Legal disclaimer revised
01.02.2008	1.0.1	AUTOSAR Administration	Chapter 8 API Spelling harmonized
20.11.2007	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

This specification 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 specification.

The material contained in this specification is protected by copyright and other types of Intellectual Property Rights. The commercial exploitation of the material contained in this specification requires a license to such Intellectual Property Rights.

This specification 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 specification may be utilized or reproduced, in any form or by any means, without permission in writing from the publisher.

The AUTOSAR specifications have been developed for automotive applications only. They have neither been developed, nor tested for non-automotive applications.

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

This specification specifies the functionality, API and the configuration of the AUTOSAR Basic Software module "FlexRay State Manager".

In the AUTOSAR Layered Software Architecture, the FlexRay State Manager belongs to the ECU Abstraction Layer, or more precisely, to the Communication Hardware Abstraction.

The main task of the FlexRay State Manager can be summarized as follows:

FrSm010:
The FlexRay State Manager shall provide an abstract interface to the AUTOSAR Communication Manager to startup or shutdown the communication on a FlexRay cluster.

FrSm012:
The FlexRay State Manager does not directly access the FlexRay hardware (FlexRay Communication Controller and FlexRay Transceiver), but by means of the FlexRay Interface. The FlexRay Interface redirects the request to the appropriate driver module.

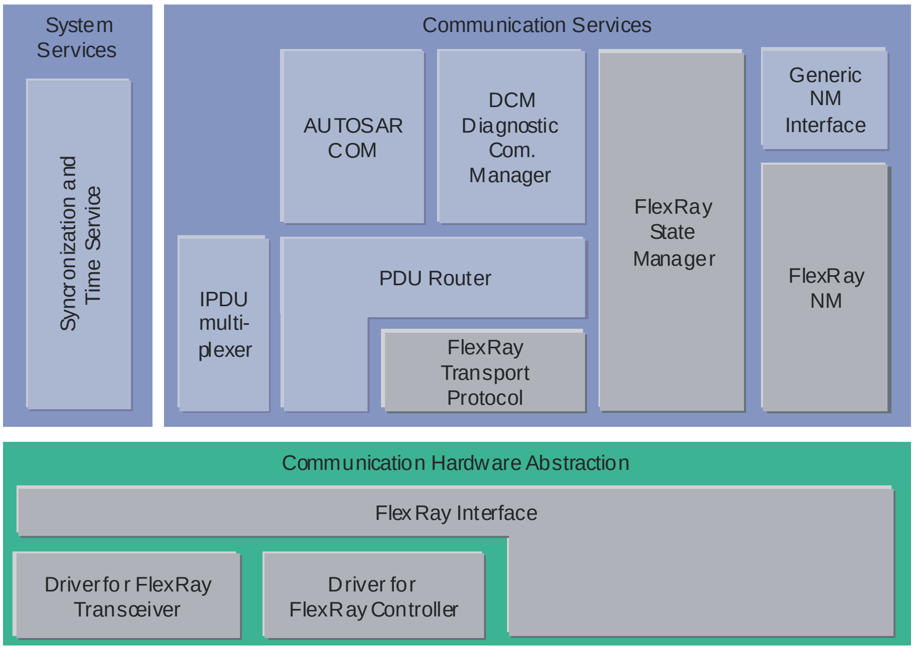

System Services
Syncronization and Time Service
Communication Services
AUTOSAR COM
DCM Diagnostic Com. Manager
FlexRay State Manager
Generic NM Interface
IPDU multi-plexer
PDU Router
FlexRay Transport Protocol
FlexRay NM
Communication Hardware Abstraction
FlexRay Interface
Driver for FlexRay Transceiver
Driver for FlexRay Controller

Figure 1 Software Architecture Overview

2 Acronyms and abbreviations

Acronym:	*Description:*
API	Application Program Interface
CC	Communication Controller
CHI	Controller Host Interface
FrIf	FlexRay Interface (AUTOSAR BSW module)
POC	Protocoll Operation Control
POCState	Actual CC internal state of the POC. This state might differ from vPOC!State in certain cases, e.g. after FREEZE command invocation (see [11] for details).
WUP	Wake-Up Pattern
ComM	AUTOSAR Communication Manager
vPOC	Data structure provided from the CC to the host at the CHI, which contains the actual POC status of the CC.
vPOC!Freeze	vPOC!Freeze denotes the Freeze bit that is part of the vPOC data structure. The Freeze bit is used by the CC to indicate that the HALT state has been entered due to an error condition.
FrSm	FlexRay State Manager

Abbreviation:	*Description:*
i.e.	[lat.] id est = [eng.] that is
e.g.	[lat.] exempli gratia = [eng.] for example
N/A	Not applicable

Term:	*Description:*
Active wake-up	Wake-up caused by the ECU e.g. by a sensor.
Passive wake-up	Wakeup caused by another ECU and propagated (e.g. by bus or wakeup-line) to the ECU currently in focus.
Remote wake-up	A passive wake-up received by the FlexRay bus.

3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_BasicSoftwareModules

[2] Layered Software Architecture
AUTOSAR_LayeredSoftwareArchitecture

[3] General Requirements on Basic Software Modules
AUTOSAR_SRS_General

[4] Specification of ECU Configuration
AUTOSAR_ECU_Configuration

[5] Specification of Communication Stack Types
AUTOSAR_SWS_ComStackTypes

[6] Requirements on FlexRay
AUTOSAR_SRS_FlexRay

[7] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRay_Interface

[8] Specification of FlexRay Driver
AUTOSAR_SWS_FlexRay_Driver

[9] Specification of Communication Manager
AUTOSAR_SWS_ComManager

[10] AUTOSAR_SRS_ModeManagement.doc
AUTOSAR_SRS_ModeManagement

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

3.2 Related standards and norms

[12] FlexRay Communications System Protocol Specification Version 2.1 Rev A

4 Constraints and assumptions

4.1 Limitations

This specification only defines the straightforward case for starting and stopping the communication on a FlexRay cluster.

The following items are not supported by the current version of this specification.

- Wake-up forwarding on a dual-channel FlexRay cluster is left open.
- The handling of single-slot mode is left open.
- The case of multiple CC of one ECU assigned to one FlexRay cluster is not fully defined as the error handling is missing. Thus, the current version can only be used for the case of one CC per cluster.

4.2 Applicability to car domains

The FlexRay Communication stack can be used wherever high data rates and fault tolerant communication (in conjunction with [11]) is required. Furthermore, it enables the synchronized operation of several ECUs within a car.
The FlexRay State Manager can be used for all domain applications which use
the FlexRay Protocol.

5 Dependencies to other modules

5.1 AUTOSAR BSW Scheduler

The BSW Scheduler calls the main functions of the FrSm, which are necessary for the cyclic processes of the FrSm.

5.2 Communication Manager

The ComM requests network communication modes and is notified by the FrSm when a communication mode is reached.

5.3 AUTOSAR FlexRay Interface

The FrSm uses the API of the FrIf to initialize the FlexRay Communication Hardware and to control the operating modes of the FlexRay Controllers and FlexRay Transceivers assigned to the FlexRay Networks.

5.4 AUTOSAR Com

The FrSm uses the API of COM to control FlexRay related I-PDU groups.

5.5 AUTOSAR Development Error Tracer

In order to be able to report development errors, the FlexRay State Manager has to have access to the error hook of the Development Error Tracer.

5.6 AUTOSAR Diagnostic Event Manager

In order to be able to report production errors the FlexRay State Manager has to have access to the Diagnostic Event Manager.

5.7 File structure

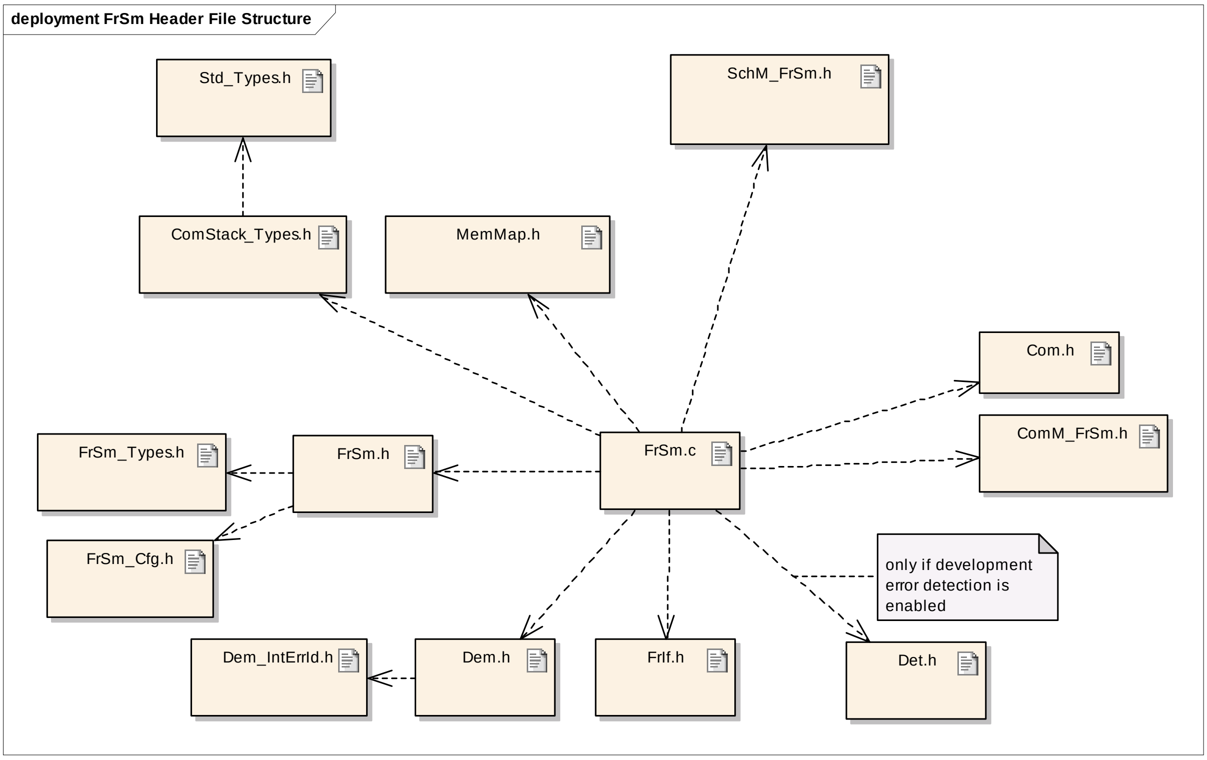
5.7.1 Code file structure

FrSm051:
The code file structure shall not be defined within this specification completely. At this point it shall be pointed out that the code-file structure shall include the following files named:

- FrSm_Lcfg.c – for link time configurable parameters and
- FrSm_PBcfg.c – for post build time configurable parameters.

These files shall contain all link time and post-build time configurable parameters.

5.7.2 Header file structure



FrSm052:
The header file FrSm.h exports the API of the FrSm. This file includes further header files and declares the function prototypes, which are supposed to be referenced by user modules.
FrSm053:
The header file FrSm_Cfg.h shall contain the pre-compile parameters of the module and the declarations of FrSm_Lcfg.c and FrSm_Pbcfg.c

FrSm054:
The header file FrSm_Types.h exports the FrSm specific types.
FrSm055:
The FrSm implementation (FrSm.c) references its header file FrSm.h to get access to its own API declaration and to its configuration parameters.
FrSm056:
The FrSm needs to report development errors if development errors are enabled by configuration. Therefore, it includes the header file Det.h.
FrSm057:
The FrSm includes the header file MemMap.h in order to map its code and data into specific memory sections.
FrSm058:
The FrSm implementation (FrSm.c) references the API of the FrIf. Therefore, it includes the header file FrIf.h.
FrSm059:
The module shall include the Dem.h file. By this inclusion the APIs to report errors as well as the required Event Id symbols are included. This specification defines the name of the Event Id symbols which are provided by XML to the DEM configuration

tool. The DEM configuration tool assigns ECU dependent values to the Event Id symbols and publishes the symbols in Dem_IntErrId.h.

FrSm106:

The FrSm implementation (FrSm.c) references the API of COM. Therefore, it includes the header file Com.h.

6 Requirements traceability

Document: AUTOSAR requirements on Basic Software, general

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	FrSm051
[BSW00404] Reference to post build time configuration	FrSm051
[BSW00405] Reference to multiple configuration sets	FrSm013
[BSW00345] Pre-compile-time configuration	FrSm053
[BSW159] Tool-based configuration	FrSm064
[BSW167] Static configuration checking	FrSm065
[BSW171] Configurability of optional functionality	FrSm066
[BSW170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable
[BSW00380] Separate C-Files for configuration parameters	FrSm051
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Not applicable
[BSW00381] Separate configuration header file for pre-compile time parameters	FrSm013
[BSW00412] Separate H-File for configuration parameters	FrSm053
[BSW00383] List dependencies of configuration files	FrSm070 FrSm071
[BSW00384] List dependencies to other modules	See chapter 5
[BSW00387] Specify the configuration class of callback function	Not applicable
[BSW00388] Introduce containers	See chapter 10.2
[BSW00389] Containers shall have names	See chapter 10.2
[BSW00390] Parameter content shall be unique within the module	See chapter 10.2
[BSW00391] Parameter shall have unique names	See chapter 10.2
[BSW00392] Parameters shall have a type	See chapter 10.2
[BSW00394] Specify the scope of the parameters	See chapter 10.2
[BSW00395] List the required parameters (per parameter	See chapter 10.2
[BSW00396] Configuration classes	See chapter 10.2
[BSW00397] Pre-compile-time parameters	See chapter 10.2
[BSW00398] Link-time parameters	See chapter 10.2
[BSW00399] Loadable Post-build time parameters	See chapter 10.2
[BSW00400] Selectable Post-build time parameters	See chapter 10.2
[BSW00438] Post Build Configuration Data Structure	FrSm013, FrSm014
[BSW00402] Published information	See 10.3

Document: AUTOSAR_SRS_ModeManagement

Requirement	***Satisfied by***
[BSW09090] User-to-channel relationship	Not applicable
[BSW09133] Assigning physical channels to the Communication Manager	Not applicable
[BSW09132] Assigning Network Management to physical channels	Not applicable
[BSW09141] Configuration of physical channel wake-up	Not applicable
[BSW09078] Coordinating communication requests	Not applicable
[BSW049] Initiating wake-up and keeping awake physical channels	Not applicable
[BSW09080] Physical channel independency	Not applicable
[BSW09081] API for requesting communication	FrSm020
[BSW09083] Support of different communication modes	See chapter 7.2
[BSW09084] API for querying the current communication	FrSm024
[BSW09085] Indication of communication mode changes	See chapter 7.2
[BSW09168] Pseudo-channel for local	Not applicable
[BSW09071] Limit Communication Manager modes	Not applicable
[BSW09157] Revoke Communication Manager mode limitation	Not applicable
[BSW09087] Proxy communication request after wake-up	Not applicable
[BSW09088] Handling of different physical channel types	Not applicable
[BSW09089] Preventing waking up physical channels	Not applicable
[BSW09155] Counting of inhibited communication requests	Not applicable
[BSW09156] API to retrieve the number of inhibited "Full Communication" mode requests	Not applicable
[BSW09079] Transparent relationship between software components and physical channels	Not applicable

Document: AUTOSAR_SRS_FlexRay
Not applicable

7 Functional specification

7.1 Background & Rationale

FlexRay start-up is a complex process that is completely different from CAN. E.g. on CAN every message can wakeup the bus, on FlexRay a special wakeup pattern is needed. In order to make the FlexRay start-up process as reliable as possible, it has to be controlled by a BSW module with in-depth FlexRay knowledge. As the AUTOSAR Communication Manager has a completely abstracted bus view, it is the task of the FlexRay State Manager to map this abstracted view to the states of the FlexRay POC and to the CHI commands to change these states.

7.2 State Machine of the FlexRay State Manager

7.2.1 General

FrSm030:
The FlexRay State Manager shall have one state machine for each FlexRay cluster.

The states of this state machine are to some extend derived from the POC states of the FlexRay CC. This document is based on the assumption that there is always a unique POC state for every FlexRay cluster (see Limitations in section 4.1).

FrSm031:
The state machine of each cluster shall be processed in the main function FrSm_MainFunction_<Cluster Id> assigned to that cluster (see section 8.5.1). However, as defined in section 8.3.2 , some transitions of the state machine shall also be processed in the context of the FrSm_RequestComMode function in order achieve a deterministic behaviour for shutdown.

7.2.2 States

FrSm032:
The state machine shall comprise the following states:

FrSm Cluster State	***Mapped FlexRay CC state***
FRSM_READY	POC:ready or (transitional) POC:default config or (transitional) POC:config or (transitional) POC:halt
FRSM_WAKEUP	POC:wake-up
FRSM_STARTUP	POC:start-up
FRSM_HALT_REQ	POC:normal active or POC:normal passive
FRSM_ONLINE	POC:normal active
FRSM_ONLINE_PASSIVE	POC:normal passive

7.2.3 Variables

FrSm033:
In addition to its state, the state machine shall comprise the following variables.

FrSm Variable	***Type***	***Description***
`reqComMode`	ComM_ModeType	The communication mode that has been requested by the ComM. The communication modes are abbreviated in this document as follows: NoCom: COMM_NO_COMMUNICATION SilentCom:COMM_SILENT_COMMUNICATION FullCom: COMM_FULL_COMMUNICATION According to the definition of ComM_ModeType these modes are ordered as follows: NoCom < SilentCom < FullCom
`startupCounter`	uint8	The number of startup attempts that have been performed

7.2.4 State Machine Configuration

FrSm034:
The state machine uses the following configuration parameters that are defined in chapter 10.2.

FrSm Configuration Parameter	***Type***	***Description***
`IsWakeupECU`	Boolean	See chapter 10.2
`IsColdstartECU`	Boolean	See chapter 10.2
`StartupRepetitionsWithWakeup`	uint8	The number of times an ECU may repeat the startup procedure including a wakeup for a FlexRay cluster. If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value ∞
`StartupRepetitions`	uint8	Determines how often the ECU can repeat the startup procedure by reinitializing the FlexRay CC, see chapter 10.2. This value must not be smaller than StartupRepetitionsWithWakeup. If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value ∞

7.2.5 Conditions

The state machine description uses the following conditions that are evaluated during runtime for each FlexRay cluster:

FrSm Condition	***Type***	***Description***
`t3_IsActive`	Boolean	Evaluates to true if t3 has been started and has not expired yet, otherwise to false

7.2.6 Timers

Timer	***Description***
`t1`	The timer t1 models the duration of time between finishing wakeup and calling FrIf_AllowColdstart in the FlexRay State Manager state machine. The duration of this timer can be statically configured with the configuration parameter FrSmDurationT1.
`t2`	The timer t2 models the time difference after which the FlexRay State Manager will repeat the startup of the FlexRay cluster. The duration of this timer can be statically configured with the configuration parameter FrSmDurationT2.
`t3`	The timer t3 supervises the transition to FullCom. The duration of this timer can be statically configured with the configuration parameter FrSmDurationT3.

FrSm103: If the configuration parameter FrSMDurationT1 is set to 0, timer t1 shall not be started. Instead, the call of FrIf_AllowColdstart shall immediately follow the call of FrIf_StartCommunication.

FrSm037: If the duration FrSMDurationT2 of timer t2 is set to 0, the startup of the FlexRay cluster shall not be supervised.

FrSm125: If the duration FrSmDurationT3 of timer t3 is set to 0, the transition to FullCom shall not be supervised.

FrSm038: The duration of timer t1, t2 and t3 shall always be multiples of the cycle time of the main function.

Note, that no assumption is made whether timer t1, t2 or t3 are implemented in software or hardware.

7.2.7 Functional Elements

FrSm039:

The functionality being performed in the transitions of the state machine is partitioned into the following functional elements.

Functional Element	***Description***
FE_WAKEUP	Call FrIf_SendWUP for one controller of the FlexRay cluster. Because of the limitations defined in chapter 4.1, the case of multiple controllers per cluster is not in the scope of this document.
FE_CONFIG	Call FrIf_ControllerInit for each controller of the FlexRay cluster.
FE_START	Call FrIf_StartCommunication for each controller of the FlexRay cluster.
FE_ALLOW_COLDSTART	Call FrIf_AllowColdstart for each controller of the FlexRay cluster if the configuration parameter IsColdstartECU is true.
FE_HALT	Call FrIf_HaltCommunication for each controller of the FlexRay cluster.
FE_TRCV_STANDBY	Call FrIf_SetTransceiverMode(FRTRCV_TRCV-MODE_STANDBY) and FrIf_EnableTransceiver-Wakeup for each transceiver of the FlexRay cluster.
FE_TRCV_NORMAL	Call FrIf_SetTransceiverMode(FRTRCV_TRCV-MODE_NORMAL), FrIf_DisableTransceiverWakeup and FrIf_ClearTransceiverWakeup for each transceiver of the FlexRay cluster.
FE_START_FRIF	Set the FrIf state to ONLINE by calling FrIf_SetState(FRIF_GOTO_ONLINE) for the cluster.
FE_STOP_FRIF	Set the FrIf state to OFFLINE by calling FrIf_SetState(FRIF_GOTO_OFFLINE) for the cluster.
FE_DEM_STATUS_FAILED	Report status of production error FRSM_E_CLUSTER_STARTUP as failed
FE_DEM_STATUS_PASSED	Report status of production error FRSM_E_CLUSTER_STARTUP as passed
FE_FULL_COM_IND	Indicate to the ComM that FullCom has been reached by calling ComM_FrSm_ModeIndication(FullCom)
FE_NO_COM_IND	Indicate to the ComM that FullCom has been left by calling ComM_FrSm_ModeIndication(NoCom).
FE_START_COM_RX	Start the I-PDU-group for reception that is referenced by FrSmRxIPduGroupRef by calling Com_IPduGroupStart with the parameter Initialize set to FrSmRxPduInit.
FE_STOP_COM_RX	Stop the I-PDU-group for reception that is referenced by FrSmRxIPduGroupRef by calling Com_IPduGroupStop.
FE_START_COM_TX	Start the I-PDU-group for transmission that is referenced by FrSmTxIPduGroupRef by calling Com_IPduGroupStart with the parameter Initialize set to FrSmTxPduInit.
FE_STOP_COM_TX	Stop the I-PDU-group for transmission that is referenced by FrSmTxIPduGroupRef by calling Com_IPduGroupStop.
FE_STARTUP_ERROR_IND	Call FrNm_StartupError.

7.2.8 Transitions

FrSm093: The following diagram defines the transitions of the FrSm state machine.

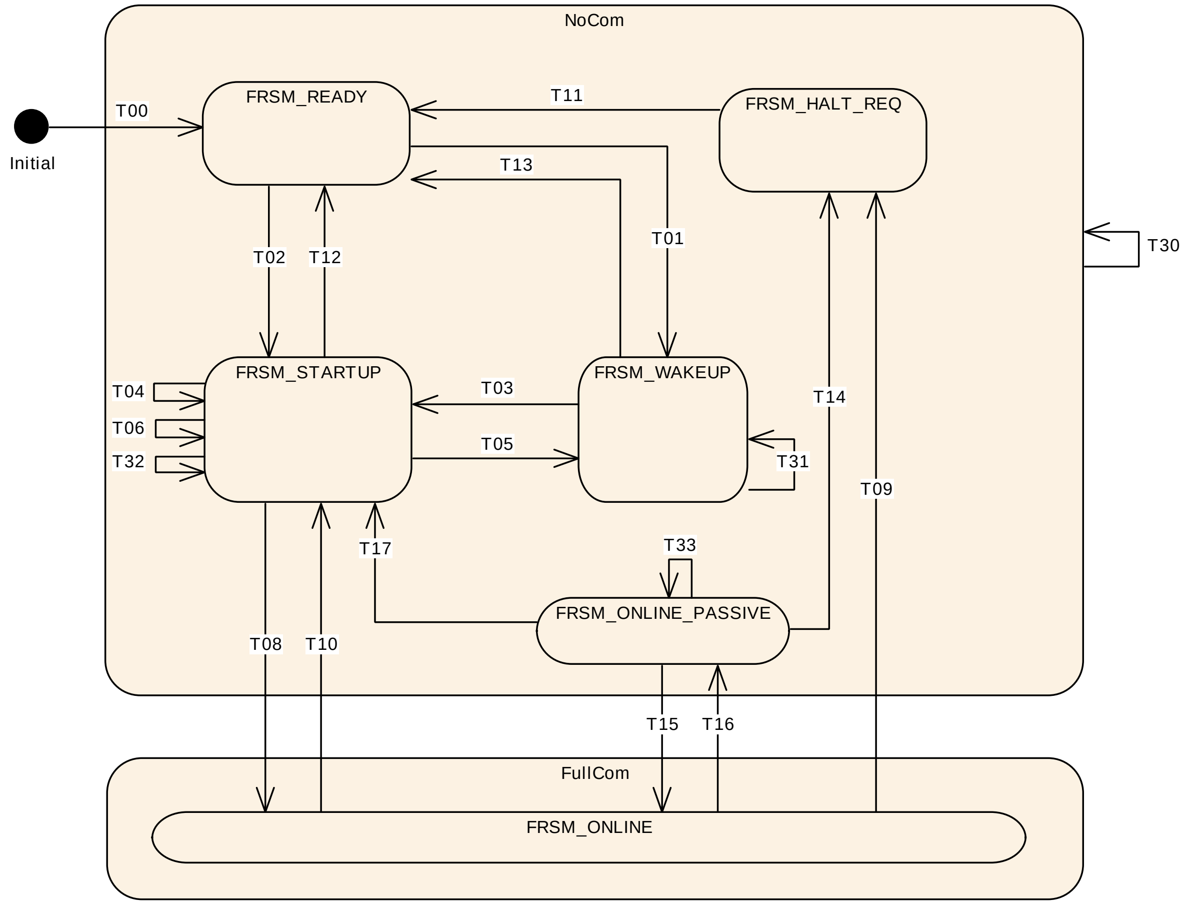


Figure 2 State machine of the FlexRay State Manager

FrSm104:
The following table defines the events and conditions that trigger the transitions of FrSm state machine and the actions that are executed within the transitions.

FrSm105:
The FrSm shall execute the actions of the transition in the order that is defined in the table.

ID	***Tran-sition***	***Event [Condition]***	***Actions***
FrSm119:	T00		FE_CONFIG
FrSm072:	T01	[reqComMode = FullCom ∧ IsWakeupECU]	FE_TRCV_NORMAL startupCounter := 1 FE_WAKEUP
FrSm073:	T02	[reqComMode = FullCom ∧ ¬ IsWakeupECU]	FE_TRCV_NORMAL startupCounter := 1 FE_START FE_ALLOW_COLDSTART start t2
FrSm074:	T03	[vPOC!State = Ready ∧ ¬ vPOC!Freeze ∧ reqComMode = FullCom]	FE_START start t1
FrSm075:	T04	t1 [reqComMode = FullCom]	FE_ALLOW_COLDSTART start t2
FrSm076:	T05	t2 [startupCounter <= StartupRepetitionsWithWakeup ∧ reqComMode = FullCom]	FE_CONFIG FE_WAKEUP startupCounter := startupCounter + 1
FrSm077:	T06	t2 [StartupRepetitionsWithWakeup < startupCounter ∧ startupCounter <= StartupRepetitions ∧ reqComMode = FullCom]	FE_CONFIG FE_START FE_ALLOW_COLDSTART startupCounter := startupCounter + 1 start t2
FrSm079:	T08	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze]	cancel t1 cancel t2 FE_START_COM_RX FE_START_FRIF FE_START_COM_TX FE_DEM_STATUS_PASSED FE_FULL_COM_IND
FrSm080:	T09	[reqComMode = NoCom]	FE_STOP_COM_TX FE_STOP_FRIF FE_STOP_COM_RX FE_HALT FE_NO_COM_IND
FrSm081:	T10	[vPOC!State = Halt ∨ vPOC!Freeze]	FE_STOP_COM_TX FE_STOP_FRIF FE_STOP_COM_RX FE_NO_COM_IND FE_CONFIG FE_START startupCounter := 1 start t2
FrSm083:	T11	[vPOC!State = Halt ∨ vPOC!Freeze]	FE_TRCV_STANDBY FE_CONFIG
FrSm084:	T12	[reqComMode = NoCom]	cancel t1 cancel t2 FE_TRCV_STANDBY FE_CONFIG

ID	Tran-sition	Event [Condition]	Actions
FrSm085:	T13	[reqComMode = NoCom]	FE_TRCV_STANDBY FE_CONFIG
	T14	[reqComMode = NoCom]	FE_STOP_FRIF FE_HALT
FrSm086:	T15	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze]	FE_START_COM_RX FE_START_COM_TX FE_FULL_COM_IND
FrSm087:	T16	[vPOC!State = Normal Passive ∧ ¬ vPOC!Freeze]	FE_STOP_COM_TX FE_STOP_COM_RX FE_NO_COM_IND
FrSm117:	T17	[vPOC!State = Halt ∨ vPOC!Freeze]	FE_STOP_FRIF FE_CONFIG FE_START startupCounter := 1 start t2
FrSm121	T30	t3	FE_DEM_STATUS_FAILED FE_STARTUP_ERROR_IND
FrSm122	T31	[¬ t3_IsActive]	FE_STARTUP_ERROR_IND
FrSm123	T32	[¬ t3_IsActive]	FE_STARTUP_ERROR_IND
FrSm124	T33	[¬ t3_IsActive]	FE_STARTUP_ERROR_IND

Legend:	∧ *AND*	start t: *start timer t*
	∨ *OR*	cancel t*: stop timer t*
	¬ *NOT*	*[…] guard condition for transition*
	:= *assignment*	*t1 […] t1 has expired*

FrSm092: The transitions T09 and T14 (see FrSm080) shall be executed in the context of the FrSm_RequestComMode function, see section 8.3.2.

FrSm040: If synchronization is lost after FullCom has been reached, the FrSm shall first try to bring the FlexRay CC to the startup state without allowing cold start.
Rationale: The loss of synchronization may be a local problem of the ECU. Thus the ECU should first try to re-integrate without disturbing the cluster.

FrSm062: If resynchronization cannot be achieved before t2 expires (see FrSm076 and FrSm077), the same wakeup and startup procedure as for the initial synchronization shall be used.

Note: If the startup of a FlexRay cluster is not successful (i.e. timer t2 expires), the FrSm module will repeat the startup procedure depending on the value of the counter startupCounter:

- If startupCounter does not exceed the threshold StartupRepetitionsWithWakeup, the startup procedure will be repeated including the wakeup.
- If startupCounter exceeds the threshold StartupRepetitionsWithWakeup but does not exceed the threshold StartupRepetitions, the startup procedure will be repeated without wakeup.

Note: If the counter startupCounter exceeds the threshold StartupRepetitions the FrSm will report the production error FRSM_E_CLUSTER_STARTUP and remain in state FRSM_STARTUP. Thus, if an ECU has been configured as a coldstart node, it will then stop performing coldstart attempts. However, if another ECU performs a coldstart, the ECU will join the coldstart.

Note: If no threshold StartupRepetitions has been configured, an ECU that has been configured as a coldstart node will not stop performing coldstart attempts until either synchronization has been achieved or NoCom is requested.

Rationale: If the RX path of a FlexRay CC is faulty, an ECU performing a wakeup or coldstart could disturb the FlexRay communication as it will not be able to detect any collision. Thus, an unlimited number of coldstart attempts could lead to a continuous disturbance of the FlexRay communication.

Note: An ECU which has been woken up by a passive wake-up does not need to perform a wake-up. However, this version of the FrSm will always try to perform a wake-up, if the ECU has been configured as a wake-up node.

7.3 Configuration description

The FlexRay State Manager configuration tool reads the ECU configuration description of the FlexRay Interface as the mapping of controllers to clusters is contained in the FlexRay Interface configuration description.

7.4 Error classification

FrSm041:
Values for production code Event Ids are assigned externally by the configuration of the Dem. They are published in the file Dem_IntErrId.h and included via Dem.h.

FrSm042::
Development error values are of type uint8.

Type or error	***Relevance***	***Related error code***	***Value [hex]***
Invalid pointer in parameter list	Development	`FRSM_E_INV_POINTER`	`0x01`
Invalid network handle parameter	Development	`FRSM_E_INV_HANDLE`	`0x02`
FrSm module was not initialized	Development	`FRSM_E_NOT_INITIALIZED`	`0x03`
Invalid communication mode requested	Development	`FRSM_E_INV_MODE`	`0x04`

FlexRay startup could not reach the state *normal active* within the configured time.	Production	`FRSM_E_CLUSTER_STARTUP`	`Assigned by DEM`

7.5 Error detection

FrSm043:
The detection of development errors shall be configurable (*ON* / *OFF*) at pre-compile time.
The switch *`FrSmDevErrorDetect`* (see chapter 10) shall activate or deactivate the detection of all development errors.

FrSm044:
If the *`FrSmDevErrorDetect`* switch is enabled API parameter checking is enabled. The detailed description of the detected errors can be found in chapter 7.4 and chapter 8.

[FrSm01202]:
The detection of production code errors cannot be switched off.

7.6 Error notification

FrSm045:
Detected development errors shall be reported to the *`Det_ReportError`* service of the Development Error Tracer (DET) if the pre-processor switch *`FrSmDevErrorDetect`* is set (see chapter 10).

FrSm046:
Production errors shall be reported to Diagnostic Event Manager.

8 API specification

8.1 Imported types

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

FrSm095:

Header file	***Imported Type***
FrTrcv_Types.h	FrTrcv_TrcvModeType
FrNm_Types.h	NetworkHandleType
FrIf_Types.h	FrIf_StateTransitionType
ComStack_Types.h	NetworkHandleType
Fr_Types.h	Fr_ChannelType
	Fr_POCStatusType
Dem_Types.h	Dem_EventIdType
Com_Types.h	Com_PduGroupIdType
Std_Types.h	Std_ReturnType
	Std_VersionInfoType
ComM_Types.h	ComM_ModeType

8.2 Type definitions

8.2.1 FrSm_ConfigType

Name:	FrSm_ConfigType
Type:	Structure
Range:	Implementation specific.
Description:	This type contains the implementation-specific post build time configuration structure that is for FrSm_Init.

8.3 Function definitions

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

8.3.1 FrSm_Init

FrSm013:

Service name:	FrSm_Init

Syntax:	`void FrSm_Init(`
	`    const FrSm_ConfigType* FrSm_ConfigPtr`
	`)`

Service ID[hex]:	0x01
Sync/Async:	Synchronous

Reentrancy:	Non Reentrant	
Parameters (in):	FrSm_ConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the FlexRay State Manager.	

FrSm014: The FrSm_Init function shall

- initialize the state machines for all FlexRay clusters and set them into the state FRSM_READY;
- internally store the configuration data address to enable subsequent API calls to access the configuration data;
- if development error detection is enabled (FrSmDevErrorDetect is ON) the successful initialization shall be remembered internally for other API functions to check for proper module initialization.

FrSm015: If development error detection is enabled (`FrSmDevErrorDetect` is ON) and FrSm_ConfigPtr equals NULL_PTR, the FrSm shall report the error FRSM_E_INV_POINTER to the DET and shall not perform the initialization. However, a value of NULL_PTR for FrSm_ConfigPtr shall not be treated as an error, if a configuration variant (see section 10.1.2) without post-build data is used.

8.3.2 FrSm_RequestComMode

FrSm020:

Service name:	FrSm_RequestComMode	

Syntax:	`Std_ReturnType FrSm_RequestComMode(`
	`NetworkHandleType NetworkHandle,`
	`ComM_ModeType ComM_Mode`
	`)`	

Service ID[hex]:	0x02	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant for different FlexRay clusters	
Parameters (in):	NetworkHandle	This parameter identifies the FlexRay cluster for which a communication mode is requested.
	ComM_Mode	This parameter holds the requested communication mode.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request not accepted
Description:	This API function is used by the ComM to startup or shutdown the communication on a FlexRay cluster.	

FrSm021: The FrSm_RequestComMode function shall store the requested communication mode. The next activation of the FrSm_MainFunction will then process this request when processing the state machine of the corresponding cluster.

Note, that the state machine definition in section 7.2 refers to this stored request as comReq.

FrSm022: If NoCom is requested after FullCom has been reached (i.e. when the FrSm state machine of the corresponding cluster is in state FRSM_ONLINE), the FrSm_RequestComMode function shall immediately process the corresponding transition of the state machine (see section 7.2).
Rationale: This shall ensure that the NoCom request will stop the participation of the ECU in the FlexRay communication at the end of the current FlexRay cycle.

FrSm023: The silent communication mode is not supported on FlexRay; it may not be requested by the ComM.

FrSm018:
If development error detection is enabled and the parameter NetworkHandle has an invalid value the development error code FRSM_E_INV_HANDLE shall be raised and the function shall return E_NOT_OK.
FrSm019:
If development error detection is enabled and the parameter ComM_Mode has an invalid value the development error code FRSM_E_INV_MODE shall be raised and the function shall return E_NOT_OK.
FrSm061:
If development error detection is enabled and the FrSm has not been initialized using FrSm_Init, the development error code FRSM_E_NOT_INITIALIZED shall be raised and the function shall return E_NOT_OK.

8.3.3 FrSm_GetCurrentComMode

FrSm024:

Service name:	FrSm_GetCurrentComMode	

Syntax:	`Std_ReturnType FrSm_GetCurrentComMode(`
	`    NetworkHandleType NetworkHandle,`
	`    ComM_ModeType* ComM_ModePtr`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FlexRay clusters	
Parameters (in):	NetworkHandle	Handle of communication network
Parameters (inout):	None	
Parameters (out):	ComM_ModePtr	Pointer to the memory location where the current communication mode shall be stored
Return value:	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request was not accepted as the FrSm has not been initiliazed using FrSm_Init.
Description:	This API function can be used to determine the current communication mode of a FlexRay cluster.	

FrSm025: The FrSm_GetCurrentComMode function shall write the current communication mode of the corresponding FlexRay cluster into the given memory location.

FrSm026:
The communication mode shall be determined as follows:

- If the cluster state machine is in state FRSM_ONLINE and the FrIf is in state FRIF_ONLINE, the communication mode is COMM_FULL_COMMUNICATION.
- In any other case, the communication mode is COMM_NO_COMMUNICATION.

FrSm027:
If development error detection is enabled and the parameter NetworkHandle has an invalid value the development error code FRSM_E_INV_HANDLE shall be raised and the function shall return E_NOT_OK.
FrSm028:
If development error detection is enabled and the parameter ComM_ModePtr equals NULL_PTR the development error code FRSM_E_INV_POINTER shall be raised and the function shall return E_NOT_OK.
FrSm060:
If development error detection is enabled and the FrSm has not been initialized using FrSm_Init, the development error code FRSM_E_NOT_INITIALIZED shall be raised and the function shall return E_NOT_OK.

8.3.4 FrSm_GetVersionInfo

FrSm029:

Service name:	FrSm_GetVersionInfo	

Syntax:	`void FrSm_GetVersionInfo(`
	`    Std_VersionInfoType* versioninfo`
	`)`	

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	This service returns the version information of this module. The version information includes: - Module Id - Vendor Id - Vendor specific version numbers (BSW00407). This function shall be pre compile time configurable On/Off by the configuration parameter: FRSM_VERSION_INFO_API Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.	

Configuration of FrSm_GetVersionInfo: This function shall be pre compile time configurable `On/Off` by the configuration parameter: `FrSmVersionInfoApi`

8.4 Call-back notifications

The FlexRay State Manager does not provide any call-back API services to other BSW modules. Therefore, the header file FrSm_Cbk.h is not needed.

8.5 Scheduled functions

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

8.5.1 FrSm_MainFunction_<Cluster Id>

FrSm118:

Service name:	FrSm_MainFunction_<Cluster Id>

Syntax:	`void FrSm_MainFunction_<Cluster Id>(`
	`)`

Service ID[hex]:	0x80
Timing:	FIXED_CYCLIC
Description:	--

FrSm114:
The Service ID of FrSm_MainFunction_<Cluster Id> shall be 0x80 + Cluster_Id.

FrSm047:
The FrSm_MainFunction shall determine the POC status of all FlexRay CC that are connected to the corresponding FlexRay cluster. This document is based on the assumption that there is always a unique POC state for every FlexRay cluster (see Limitations in section 4.1).

FrSm048:
After determining the POC status, the FrSm_MainFunction shall process the state machine of the corresponding cluster.

FrSm049:
The FrSm_MainFunction shall be called cyclically with a cycle time that is shorter than or equal to the FlexRay cycle duration.
Rationale: The FrSm_MainFunction should be called at least once per FlexRay cycle. As the POC status only changes once per cycle, multiple invocations per FlexRay cycle have no benefit.

Note: After FullCom has been reached, the invocation of the FrSm_MainFunction can optionally be synchronized to the FlexRay global time to ensure that the FrSm_Main-

Function is activated once per FlexRay cycle. However, this is outside of the scope of this specification.

Note: In case of very short FlexRay cycle times the FrSm_MainFunction can optionally be called with a cycle time that is larger than the FlexRay cycle time. However, this is outside of the scope of this specification as it can lead to increased startup time and to undetected POC status changes.

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

FrSm096:

API function	***Description***
FrIf_SendWUP	Wraps the FlexRay Driver API function Fr_SendWUP().
FrIf_ClearTransceiverWakeup	Wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverWakeup().
Com_IpduGroupStart	Starts a preconfigured I-PDU group. For example, cyclic I-PDUs will be sent out cyclically after the call of Com_IpduGroupStart(). See Chapter 7.4.5 for details. If Initialize is true all I-PDUs of the I-PDU group shall be (re-)initialized before the I-PDU group is started. That is they shall behave like after a start-up of COM, for example the old_value of the filter objects and shadow buffers of signal groups have to be (re-)initialized.
FrIf_SetState	Requests FrIf state machine transition.
FrIf_DisableTransceiverWakeup	Wraps the FlexRay Transceiver Driver API function FrTrcv_DisableTransceiverWakeup().
FrIf_AllowColdstart	Wraps the FlexRay Driver API function Fr_AllowColdstart().
FrIf_StartCommunication	Wraps the FlexRay Driver API function Fr_StartCommunication().
ComM_BusSM_ModeIndication	Indication of the actual bus mode by the corresponding Bus State Manager. ComM shall propagate the indicated state to the users with means of the RTE (see ComM661).
FrIf_EnableTransceiverWakeup	Wraps the FlexRay Transceiver Driver API function FrTrcv_EnableTransceiverWakeup().
FrIf_HaltCommunication	Wraps the FlexRay Driver API function Fr_HaltCommunication().
Com_IpduGroupStop	Stops a preconfigured I-PDU group. For example, cyclic I-PDUs will be stopped after the call of Com_IpduGroupStop(). See Chapter 7.4.5 for details.
FrNm_StartupError	This function is called by the FrSM when synchronization of the FlexRay cluster could not be achieved.
FrIf_ControllerInit	Initialized a FlexRay CC.
FrIf_GetPOCStatus	Wraps the FlexRay Driver API function Fr_GetPOCStatus().
Dem_ReportErrorStatus	Reports errors to the DEM.
FrIf_SetTransceiverMode	Wraps the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode() .

8.6.2 Optional Interfaces

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

FrSm097:

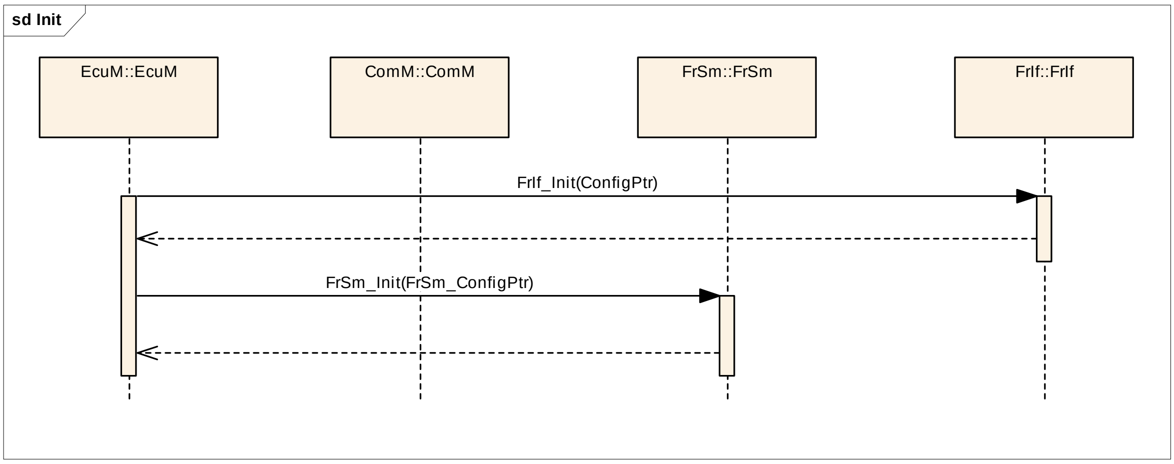
API function	*Description*
Det_ReportError	Service to report development errors.

8.6.3 Configurable interfaces

The FrSm has no configurable interface.

9 Sequence diagrams

9.1 Initialization



9.2 Transition from no communication to full communication

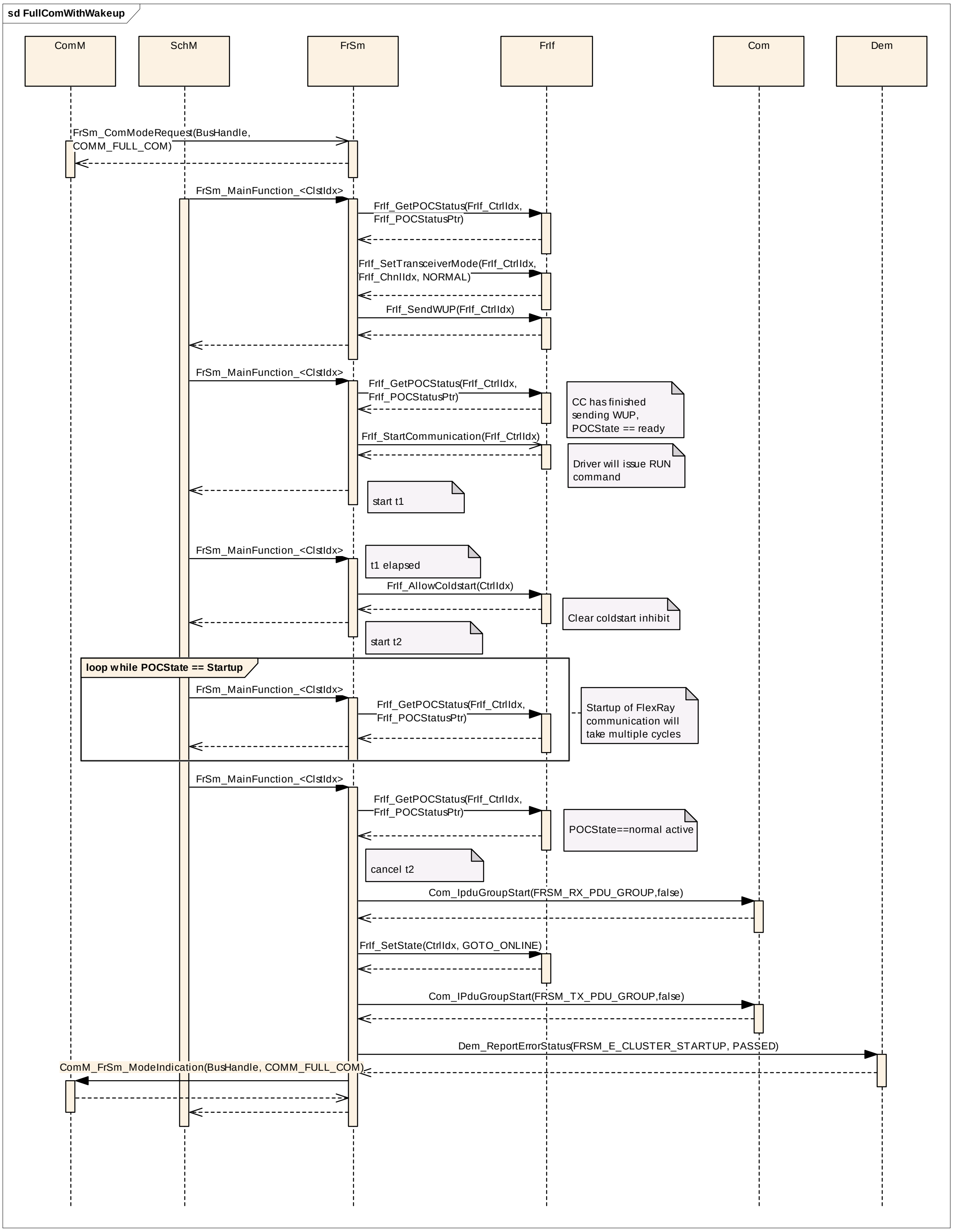
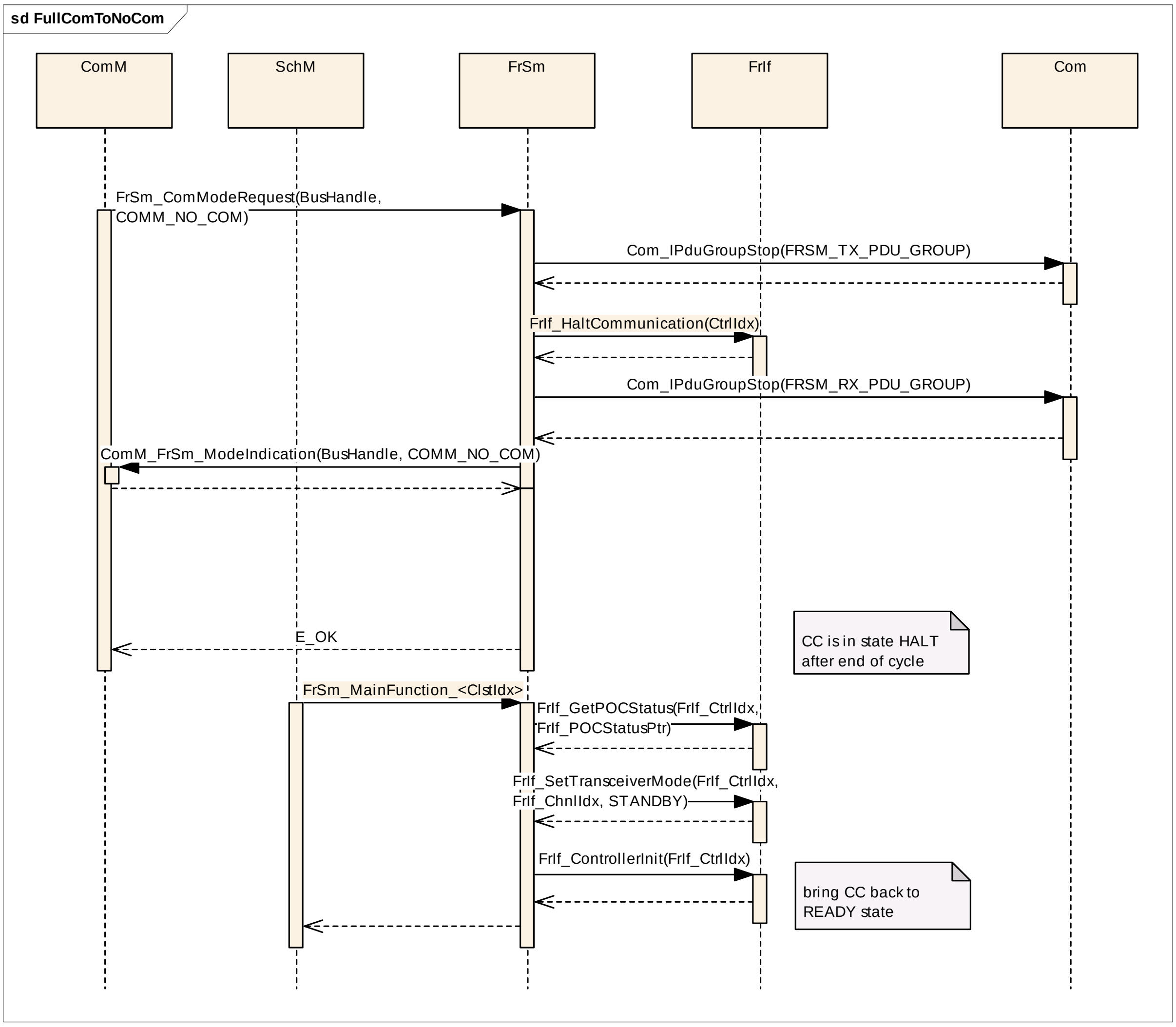

sd FullComWithWakeup
ComM
SchM
FrSm
FrIf
Com
Dem
FrSm_ComModeRequest(BusHandle, COMM_FULL_COM)
FrSm_MainFunction_<ClstIdx>
FrIf_GetPOCStatus(FrIf_CtrlIdx, FrIf_POCStatusPtr)
FrIf_SetTransceiverMode(FrIf_CtrlIdx, FrIf_ChnlIdx, NORMAL)
FrIf_SendWUP(FrIf_CtrlIdx)
FrSm_MainFunction_<ClstIdx>
FrIf_GetPOCStatus(FrIf_CtrlIdx, FrIf_POCStatusPtr)
CC has finished sending WUP, POCState == ready
FrIf_StartCommunication(FrIf_CtrlIdx)
Driver will issue RUN command
start t1
FrSm_MainFunction_<ClstIdx>
t1 elapsed
FrIf_AllowColdstart(CtrlIdx)
Clear coldstart inhibit
start t2
loop while POCState == Startup
FrSm_MainFunction_<ClstIdx>
FrIf_GetPOCStatus(FrIf_CtrlIdx, FrIf_POCStatusPtr)
Startup of FlexRay communication will take multiple cycles
FrSm_MainFunction_<ClstIdx>
FrIf_GetPOCStatus(FrIf_CtrlIdx, FrIf_POCStatusPtr)
POCState==normal active
cancel t2
Com_IpduGroupStart(FRSM_RX_PDU_GROUP,false)
FrIf_SetState(CtrlIdx, GOTO_ONLINE)
Com_IPduGroupStart(FRSM_TX_PDU_GROUP,false)
Dem_ReportErrorStatus(FRSM_E_CLUSTER_STARTUP, PASSED)
ComM_FrSm_ModeIndication(BusHandle, COMM_FULL_COM)

Figure 3 Transition from no communication to full communication for the case of an ECU that is configured as wake-up and coldstart node.

9.3 Transition from full communication to no communication


sd FullComToNoCom
ComM
SchM
FrSm
FrIf
Com
FrSm_ComModeRequest(BusHandle, COMM_NO_COM)
Com_IPduGroupStop(FRSM_TX_PDU_GROUP)
FrIf_HaltCommunication(CtrlIdx)
Com_IPduGroupStop(FRSM_RX_PDU_GROUP)
ComM_FrSm_ModeIndication(BusHandle, COMM_NO_COM)
CC is in state HALT after end of cycle
E_OK
FrSm_MainFunction_<ClstIdx>
FrIf_GetPOCStatus(FrIf_CtrlIdx, FrIf_POCStatusPtr)
FrIf_SetTransceiverMode(FrIf_CtrlIdx, FrIf_ChnlIdx, STANDBY)
FrIf_ControllerInit(FrIf_CtrlIdx)
bring CC back to READY state

10 Configuration specification

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

Chapter 10.2 specifies the structure (containers) and the parameters of the module FlexRay State Manager.

Chapter 10.3 specifies published information of the module FlexRay State Manager.

10.1 How to read this chapter

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

- AUTOSAR Layered Software Architecture [2]
- AUTOSAR ECU Configuration Specification [4]
  This document describes the AUTOSAR configuration methodology and the AUTOSAR configuration metamodel in detail.

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

10.1.1 Configuration and configuration parameters

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

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

10.1.2 Variants

Variants describe sets of configuration parameters. E.g., variant 1: only pre-compile time configuration parameters; variant 2: mix of pre-compile- and post build time-configuration parameters. In one variant a parameter can only be of one configuration class.

10.1.3 Containers

Containers structure the set of configuration parameters. This means:

- *all* configuration parameters are kept in containers.

- (sub-) containers can reference (sub-) containers. It is possible to assign a multiplicity to these references. The multiplicity then defines the possible number of instances of the contained parameters.

10.1.4 Specification template for configuration parameters

The following tables consist of three sections:

- the general section
- the configuration parameter section
- the section of included/referenced containers

Pre-compile time - specifies whether the configuration parameter shall be of configuration class *Pre-compile time* or not

Label	***Description***
X	The configuration parameter shall be of configuration class *Pre-compile time*.
--	The configuration parameter shall never be of configuration class *Pre-compile time*.

Link time - specifies whether the configuration parameter shall be of configuration class *Link time* or not

Label	***Description***
X	The configuration parameter shall be of configuration class *Link time*.
--	The configuration parameter shall never be of configuration class *Link time*.

Post Build - specifies whether the configuration parameter shall be of configuration class *Post Build* or not

Label	***Description***
X	The configuration parameter shall be of configuration class *Post Build* and no specific implementation is required.
L	*Loadable* - the configuration parameter shall be of configuration class *Post Build* and only one configuration parameter set resides in the ECU.
M	*Multiple* - the configuration parameter shall be of configuration class *Post Build* and is selected out of a set of multiple parameters by passing a dedicated pointer to the init function of the module.
--	The configuration parameter shall never be of configuration class *Post Build*.

>

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described Chapters 7 and Chapter 8.
FrSm064: The FrSm shall support tool based configuration.
FrSm065: The configuration tool shall check the consistency of the configuration parameters at system configuration time.

10.2.1 Variants

10.2.1.1 Variant1 (Pre-compile Configuration)

FrSm098: In the pre-compile configuration all parameters below that are marked as Pre-compile configurable shall be configurable in a pre-compile manner, for example as #defines.

The module is most likely delivered as source code.

10.2.1.2 Variant2 (Link-time Configuration)

FrSm099: This configuration includes all configuration options of the “Pre-compile Configuration”. Additionally all parameters defined below, as link-time configurable shall be configurable at link time for example by linking a special configured parameter object file.

The module is most likely delivered as object code.

10.2.1.3 Variant3 (Post-build Configuration)

FrSm100:
This configuration includes all configuration options of the “Link-time configuration”. Additionally all parameters defined below, as post build configurable shall be configurable post build for example by flashing configuration data.

The module is most likely delivered as object code.

10.2.2 FrSm

Module Name	*FrSm*
Module Description	

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
FrSmCluster	1..*	
FrSmGeneral	1	

10.2.3 FrSmGeneral

SWS Item	**FrSm107 :**
Container Name	FrSmGeneral
Description	
Configuration Parameters	

SWS Item	**FrSm066 :**		
Name	FrSmDevErrorDetect {FRSM_DEV_ERROR_DETECT}		
Description	Enables and disables the development error detection and notification mechanism.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrSm108 :**		
Name	FrSmVersionInfoApi {FRSM_VERSION_INFO_API}		
Description	Enables and disables the version info API		
Multiplicity	1		
Type	BooleanParamDef		
Default value	false		
ConfigurationClass	*Pre-compile time*	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.2.4 FrSmCluster

SWS Item	**FrSm067 :**
Container Name	FrSmCluster{FrSmClusterConfiguration} [Multi Config Container]
Description	
Configuration Parameters	

SWS Item	**FrSm102 :**		
Name	FrSmDurationT1 {FrSmDurationT1}		
Description	The duration of timer t1 as multiples of the cycle time of the FrSm main function. A value of 0 shall imply that the timer is not used.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	*Pre-compile time*	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME

	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: Fr/FrController/PMicroPerCycle		

SWS Item	**FrSm89 :**		
Name	FrSmDurationT2 {FrSmDurationT2}		
Description	The duration of timer t2 as multiples of the cycle time of the FrSm main function. A value of 0 shall imply that the timer is not used.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: Fr/FrController/PMicroPerCycle		

SWS Item	**FrSm120 :**		
Name	FrSmDurationT3 {FrSmDurationT3}		
Description	The duration of timer t3 as multiples of the cycle time of the FrSm main function. A value of 0 shall imply that the timer is not used.		
Multiplicity	1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: Fr/FrController/PMicroPerCycle		

SWS Item	**FrSm068 :**		
Name	FrSmIsColdstartEcu {FrSmIsColdstartECU}		
Description	True: The ECU is a coldstart node for this FlexRay cluster. False: The ECU is no coldstart node for this FlexRay cluster.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrSm109 :**		
Name	FrSmIsWakeupEcu {FrSmIsWakeupECU}		
Description	True: FrSm shall perform a wakeup for this cluster. False: FrSm shall never perform a wakeup for this FlexRay cluster.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrSm115 :**
Name	FrSmMainFunctionCycleTime {FRSM_MAIN_FUNCTION_CYCLE_TIME}

Description	This parameter defines the cycle time of the periodic calling of FrSm main function.		
Multiplicity	1		
Type	FloatParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrSm242 :**		
Name	FrSmRxPduInit		
Description	This parameter shall configure the initialize parameter for the configured RX PDU group, when the FrSM module references the API Com_IpduGroupStart in the transition from no communication to full communication.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrSm069 :**		
Name	FrSmStartupRepetitions {FrSmStartupRepetitions}		
Description	The number of times an ECU may repeat the startup procedure for a FlexRay cluster.		
Multiplicity	0..1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module dependency: This value must be greater or equal to FrSmStartupRepetitionsWithWakeup		

SWS Item	**FrSm094 :**		
Name	FrSmStartupRepetitionsWithWakeup {FrSmStartupRepetitionsWithWakeup}		
Description	The number of times an ECU may repeat the startup procedure including a wakeup for a FlexRay cluster.		
Multiplicity	0..1		
Type	IntegerParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrSm243 :**
Name	FrSmTxPduInit
Description	This parameter shall configure the initialize parameter for the configured TX PDU group, when the FrSM module references the API

	Com_IpduGroupStart in the transition from no communication to full communication.		
Multiplicity	1		
Type	BooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrSm116 :**		
Name	FrSmFrIfClusterRef {FrIfClusterRef}		
Description	References the cluster configuration in the FlexRay Interface configuration. Note that the assigned controllers and transceivers are defined in the FrIf configuration and can be accessed via this reference.		
Multiplicity	1		
Type	Reference to FrIfCluster		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: module		

SWS Item	**FrSm070 :**		
Name	FrSmNetworkHandleRef {FrSmNetworkHandle}		
Description	Reference to the unique handle to identify one certain FlexRay network correspond to one of the network handles of the ComM configuration.		
Multiplicity	1		
Type	Reference to ComMChannel		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: Module		

SWS Item	**FrSm088 :**		
Name	FrSmRxIPduGroupRef {RxIpuGroupId}		
Description	ID of the RX-IPDU group that shall be started and stopped by the FrSm.		
Multiplicity	1		
Type	Reference to ComIPduGroup		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module		

SWS Item	**FrSm101 :**		
Name	FrSmTxIPduGroupRef {TxIpuGroupId}		
Description	ID of the TX-IPDU group that shall be started and stopped by the FrSm.		
Multiplicity	1		
Type	Reference to ComIPduGroup		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: Module		

No Included Containers

10.3 Published Information

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

The standard common published information like

vendorId (<Module>_VENDOR_ID),
moduleId (<Module>_MODULE_ID),
arMajorVersion (<Module>_AR_MAJOR_VERSION),
arMinorVersion (<Module>_ AR_MINOR_VERSION),
arPatchVersion (<Module>_ AR_PATCH_VERSION),
swMajorVersion (<Module>_SW_MAJOR_VERSION),
swMinorVersion (<Module>_ SW_MINOR_VERSION),
swPatchVersion (<Module>_ SW_PATCH_VERSION),
vendorApiInfix (<Module>_VENDOR_API_INFIX)

is provided in the BSW Module Description Template (see [11] Figure 4.1 and Figure 7.1).

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

11 Changes during SWS Improvements by Technical Office

11.1 Deleted SWS Items

None

11.2 Replaced SWS Items

None

11.3 Changed SWS Items

None

11.4 Added SWS Items

SWS Item	***Rationale***
FrSM095	UML model linking of imported types
FrSM096	UML model linking of mandatory interfaces
FrSM097	UML model linking of optional interfaces
FrSM098	Definition of configuration variant needs an ID
FrSM099	Definition of configuration variant needs an ID
FrSM100	Definition of configuration variant needs an ID

12 Changes for Revision 3.1.5

12.1 Deleted SWS Items

SWS Item	*Rationale*
FrSm078	Supervision is now done using t3

12.2 Replaced SWS Items

None

12.3 Changed SWS Items

SWS Item	*Rationale*
FrSm037	Removed part of requirement that was redundant to chapter 10
FrSm103	Removed part of requirement that was redundant to chapter 10

12.4 Added SWS Items

SWS Item	*Rationale*
FrSm120	supervision with timer T3
FrSm121	supervision with timer T3
FrSm122	supervision with timer T3
FrSm123	supervision with timer T3
FrSm124	supervision with timer T3
FrSm125	supervision with timer T3