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

Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R23-11

Document Change History

Date	**Release**	**Changed by**	**Description**
2023-11-23	R23-11	AUTOSAR Release Management	• Editorial changes • Naming harmonized • Specification IDs for production errors
2022-11-24	R22-11	AUTOSAR Release Management	• Improved traceability
2021-11-25	R21-11	AUTOSAR Release Management	• No content changes
2020-11-30	R20-11	AUTOSAR Release Management	• Improved Error Handling Sections
2019-11-28	R19-11	AUTOSAR Release Management	• Updated Transitions T03 and T06 • Changed Document Status from Draft to published
2018-10-30	4.4.0	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation
2017-12-08	4.3.1	AUTOSAR Release Management	• Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation

▽

△

2016-11-30	4.3.0	AUTOSAR Release Management	• Added wakeup forwarding for dual channel FlexRay networks • Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation
2015-07-31	4.2.2	AUTOSAR Release Management	• Revised development error handling • Debugging support marked as obsolete • Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation
2014-10-31	4.2.1	AUTOSAR Release Management	• Changed development error checking of FrSM_Init pointer parameter • Editorial Changes
2014-03-31	4.1.3	AUTOSAR Release Management	• Removed Dual Channel Wakeup Echo
2013-10-31	4.1.2	AUTOSAR Release Management	• Added immediate handling of NoCom requests in normal passive mode or key slot only mode • Editorial changes • Removed chapter(s) on change documentation
2013-03-15	4.1.1	AUTOSAR Administration	• FlexRay Transceiver Mode Switch can be delayed • Formal updates
2011-12-22	4.0.3	AUTOSAR Administration	• Short term loss of synchronization is reported to DEM or DET • Number of startup frames can be monitored during normal operation • Revised production error handling
2010-09-30	3.1.5	AUTOSAR Administration	• The amount of wakeup patterns can be configured • Clearing the Coldstart Inhibit Mode can be delayed also for passive wakeup • Removed enabling and disabling of transceiver wakeups

▽

△

2010-02-02	3.1.4	AUTOSAR Administration	• Added support of FlexRay Dual Channel Wakeup • Added support of FlexRay Single Slot Mode • Added support of Passive Mode (Receive only) • Improved timeout supervision of FlexRay startup • Legal disclaimer revised
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised
2008-02-01	3.0.2	AUTOSAR Administration	• Chapter 8 API Spelling harmonized
2007-12-21	3.0.1	AUTOSAR Administration	• Initial Release

Contents

1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module FlexRay State Manager (`FrSM`).

The AUTOSAR BSW stack specifies for each communication bus a bus specific state manager. This module shall implement the control flow for the respective bus. The `FrSM` is a member of the Communication Service Layer. It interacts with the Communication Hardware Abstraction Layer and the System Service Layer.

2 Acronyms and Abbreviations

The glossary below includes acronyms and abbreviations relevant to the FlexRay Manager module that are not included in the [1, AUTOSAR glossary].

Acronym/ Abbrevation	Description:
BswM	Basic Software Mode Manager
CC	Communication Controller
CHI	Controller Host Interface
ComM	AUTOSAR Communication Manager
DCM	Diagnostic Communication Manager
e.g.	[lat.] exempli gratia = [eng.] for example
EcuM	ECU State Manager
Fr	FlexRay Driver
FrIf	FlexRay Interface (AUTOSAR BSW module)
FrSM	FlexRay State Manager
FrTrcv	FlexRay Transceiver Driver
i.e.	[lat.] id est = [eng.] that is
N/A	Not applicable
POC	Protocol 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).
RX	Reception
SchM	Schedule Manager
TX	Transmission
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.
vPOC!SlotMode	vPOC!SlotMode denotes the SlotMode field that is part of the vPOC data structure.
WUP	Wake-Up Pattern

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 or wakeup-line.

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

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

[4] General Requirements on Basic Software Modules
AUTOSAR_CP_SRS_BSWGeneral

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [2, SWS BSW General], which is also valid for FrSM.

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

4 Constraints and assumptions

4.1 Limitations

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

For the case of multiple CC of one ECU assigned to one FlexRay cluster some items are left open for the implementation:

- Which CC is used to transmit the wakeup pattern
- Handling of inconsistent POC states in the CCs

4.2 Applicability to car domains

The FlexRay Communication stack can be used wherever high data rates and fault tolerant communication (in conjunction with [3, FlexRay protocol specification]) is required. Furthermore, it enables the synchronized operation of several ECUs within a car.

The FrSM 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 AUTOSAR 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 Default Error Tracer

In order to be able to report development errors, the FrSM has to have access to the error hook of the Default Error Tracer.

5.5 AUTOSAR Diagnostic Event Manager

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

5.6 AUTOSAR BSW Mode Manager

In order to be able to report state changed the FrSM has to have access to the BSW Mode Manager.

5.7 AUTOSAR FlexRay Network Management

In order to be able to report startup failures the FrSM has to have access to the Flex Ray Network Management.

5.8 File structure

5.8.1 Code file structure

For details refer to the chapter 5.1.6 "Code file structure" in [2, SWS BSW General].

5.8.2 Header file structure

[SWS_FrSM_00139] ⌈The header file FrSM.h shall include a software and specification version number.⌋*()*

[SWS_FrSM_00140] ⌈The `FrSM` module shall perform a consistency check between code files and header files based on pre-process-checking the version numbers of related code files and header files.⌋*(SRS_BSW_00004)*

6 Requirements Tracing

The following tables reference the requirements specified in [4, SRS BSW General] and links to the fulfillment of these. Please note that if column "Satisfied by" is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_FrSM_00140]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_FrSM_00126]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_FrSM_00064]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_FrSM_00065]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_FrSM_00018] [SWS_FrSM_00028] [SWS_FrSM_00168]
[SRS_BSW_00350]	All AUTOSAR Basic Software Modules shall allow the enabling/ disabling of detection and reporting of development errors.	[SWS_FrSM_00018] [SWS_FrSM_00019] [SWS_FrSM_00027] [SWS_FrSM_00028] [SWS_FrSM_00060] [SWS_FrSM_00061] [SWS_FrSM_00141] [SWS_FrSM_00168] [SWS_FrSM_00169] [SWS_FrSM_00179]
[SRS_BSW_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_FrSM_00018] [SWS_FrSM_00028] [SWS_FrSM_00168]
[SRS_BSW_00373]	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	[SWS_FrSM_00118]
[SRS_BSW_00385]	List possible error notifications	[SWS_FrSM_00300] [SWS_FrSM_00301]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_FrSM_00300] [SWS_FrSM_00301]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_FrSM_00013]
[SRS_BSW_00406]	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	[SWS_FrSM_00060] [SWS_FrSM_00061] [SWS_FrSM_00169] [SWS_FrSM_00179]
[SRS_BSW_00407]	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	[SWS_FrSM_00029]
[SRS_BSW_00438]	Configuration data shall be defined in a structure	[SWS_FrSM_00013] [SWS_FrSM_00126] [SWS_FrSM_00127] [SWS_FrSM_00128]
[SRS_BSW_00450]	A Main function of a un-initialized module shall return immediately	[SWS_FrSM_00181]
[SRS_BSW_00458]	Classification of production errors	[SWS_FrSM_00300] [SWS_FrSM_00301]
[SRS_BSW_00483]	BSW Modules shall handle buffer alignments internally	[SWS_FrSM_00026] [SWS_FrSM_00127]

Table 6.1: RequirementsTracing

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 ComM has a completely abstracted bus view, it is the task of the FrSM to map this abstracted view to the states of the FlexRay POC and to the CHI commands to change these states.

7.2 Main Task of the FlexRay State Manager

The main task of the FrSM module can be summarized as follows:

The FrSM module shall provide an abstract interface to the AUTOSAR ComM module to startup or shutdown the communication on a FlexRay cluster. The FrSM module shall not directly access the FlexRay hardware (FlexRay Communication Controller and FlexRay Transceiver), but by means of the FrIf module. FrIf module redirects the request to the appropriate driver module.

7.3 State Machine of the FlexRay State Manager

7.3.1 General

[SWS_FrSM_00030] ⌈FrSM shall implement one state machine for each FlexRay cluster.⌋*()*

The states of this state machine are to some extent 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).

The state machine of each cluster is processed by the main function FrSM_MainFunction() assigned to that cluster. However, some transitions of the state machine are processed in the context of the FrSM_RequestComMode() function in order to achieve a deterministic behavior for shutdown.

7.3.2 States

[SWS_FrSM_00032] ⌈The state machine shall comprise the following states:

- FRSM_READY

- FRSM_WAKEUP
- FRSM_STARTUP
- FRSM_HALT_REQ
- FRSM_ONLINE
- FRSM_ONLINE_PASSIVE
- FRSM_KEYSLOT_ONLY
- FRSM_LOW_NUMBER_OF_COLDSTARTERS

⌋*()*

Table 7.1 provides additional information about the state machine states.

FrSM Cluster State	Mapped FlexRay CC state	Description
FRSM_READY	POC:ready	
FRSM_WAKEUP	POC:wake-up	FrSM performs wake-up
FRSM_STARTUP	POC:start-up	FrSM performs startup
FRSM_HALT_REQ	POC:normal active or POC:normal passive	FrSM performs a shutdown
FRSM_ONLINE	POC:normal active	Full Communication
FRSM_ONLINE_PASSIVE	POC:normal passive	Due to clock synchronization errors no data is transmitted or received.
FRSM_KEYSLOT_ONLY	POC:normal active and vPOCSlotMode $\neq$ AllSlots	Data can only be transmitted in the key slots.
FRSM_LOW_NUMBER_OF_COLDSTARTERS	POC:normal active	Full communication; FlexRay is synchronized based on sync frames only.

Table 7.1: FrSM states and their mapping

[SWS_FrSM_00176] ⌈For controlling the passive mode (receive-only), the state machine shall additionally comprise the following states which concurrent to the states above:

- FRSM_ECU_ACTIVE
- FRSM_ECU_PASSIVE

⌋*()*

Table 7.2 provides more information about the additional states.

FrSM additional State	Description
FRSM_ECU_ACTIVE	When the FrSM is concurrently in state FRSM_READY, the transceivers are in set into mode FRTRCV_TRCVMODE_STANDBY, otherwise into mode FRTRCV_TRCVMODE_NORMAL

▽

△

FRSM_ECU_PASSIVE	When the FrSM is concurrently in state FRSM_READY, the transceivers are in set into mode FRTRCV_TRCVMODE_STANDBY, otherwise into mode FRTRCV_TRCVMODE_RECEIVEONLY.

Table 7.2: FrSM Additional States

[SWS_FrSM_00180] ⌈For reporting these two concurrent states to the BswM, a corresponding value of FrSM_BswM_StateType shall be determined as follows:

- FrSM additional state is FRSM_ECU_ACTIVE then the according FrSM state is reported, e.g. FRSM_ONLINE
- FrSM additional state is FRSM_ECU_PASSIVE then the according FrSM state is reported with the postfix '_ECU_PASSIVE', e.g. FRSM_ONLINE_ECU_PASSIVE

⌋*()*

Table 7.3 provides all possible reportings to BswM

FrSM Cluster State	**FrSM additional State**	**FrSM_BswM_StateType value**
FRSM_READY	FRSM_ECU_ACTIVE	FRSM_READY
FRSM_READY	FRSM_ECU_PASSIVE	FRSM_READY_ECU_PASSIVE
FRSM_WAKEUP	FRSM_ECU_ACTIVE	FRSM_WAKEUP
FRSM_WAKEUP	FRSM_ECU_PASSIVE	FRSM_WAKEUP_ECU_PASSIVE
FRSM_STARTUP	FRSM_ECU_ACTIVE	FRSM_STARTUP
FRSM_STARTUP	FRSM_ECU_PASSIVE	FRSM_STARTUP_ECU_PASSIVE
FRSM_ONLINE	FRSM_ECU_ACTIVE	FRSM_ONLINE
FRSM_ONLINE	FRSM_ECU_PASSIVE	FRSM_ONLINE_ECU_PASSIVE
FRSM_ONLINE_PASSIVE	FRSM_ECU_ACTIVE	FRSM_ONLINE_PASSIVE
FRSM_ONLINE_PASSIVE	FRSM_ECU_PASSIVE	FRSM_ONLINE_PASSIVE_ECU_PASSIVE
FRSM_KEYSLOT_ONLY	FRSM_ECU_ACTIVE	FRSM_KEYSLOT_ONLY
FRSM_KEYSLOT_ONLY	FRSM_ECU_PASSIVE	FRSM_KEYSLOT_ONLY_ECU_PASSIVE
FRSM_HALT_REQ	FRSM_ECU_ACTIVE	FRSM_HALT_REQ
FRSM_HALT_REQ	FRSM_ECU_PASSIVE	FRSM_HALT_REQ_ECU_PASSIVE
FRSM_LOW_NUMBER_OF_COLDSTARTERS	FRSM_ECU_ACTIVE	FRSM_LOW_NUMBER_OF_COLDSTARTERS
FRSM_LOW_NUMBER_OF_COLDSTARTERS	FRSM_ECU_PASSIVE	FRSM_LOW_NUMBER_OF_COLDSTARTERS_ECU_PASSIVE

Table 7.3: FrSM state reporting to BswM

7.3.3 Variables

In addition to its state, the state machine description uses the following variables. Note that these variables are only auxiliary means for improving the clearness and the readability of the specification.

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	Integer	The number of startup attempts that have been performed
wakeupType	Enum	The following values are supported: • SingleChannelWakeup • DualChannelWakeup • DualChannelWakeupForward • NoWakeup
wakeupTransmitted	Boolean	True if vPOC!WakeupStatus = FR_WAKEUP_TRANSMITTED for at least attempt to transmit a wakeup pattern, false otherwise
busTrafficDetected	Boolean	True if vPOC!WakeupStatus = FR_WAKEUP_RECEIVED_HEADER or FR_WAKEUP_RECEIVED_WUP for at least attempt to transmit a wakeup pattern, false otherwise
wakeupCounter	Integer	The number of attempts that have been performed for transmitting a wakeup pattern.

Note that the silent communication mode (SilentCom) is not supported on FlexRay; it may not be requested by the ComM module.

7.3.4 State Machine Configuration

The state machine description uses the following configuration parameters that are defined in subsection 10.2.4 for each FlexRay cluster:

FrSM Configuration Parameter	Type	Description
FrSMIsWakeupEcu	Boolean	Configuration parameter, see subsection 10.2.4.
FrSMCheckWakeupReason	Boolean	Configuration parameter, see subsection 10.2.4.
FrSMIsColdstartEcu	Boolean	Configuration parameter, see subsection 10.2.4.
FrSMIsDualChannelNode	Boolean	This parameter is derived from the FrIf configuration. If the corresponding FrIf cluster is connected to both channels of the FlexRay cluster, this parameter is TRUE. Otherwise, it is FALSE.

▽

△

FrSMStartupRepetitionsWithWakeup	Integer	Configuration parameter, see subsection 10.2.4. If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value ∞
FrSMStartupRepetitions	Integer	Configuration parameter, see subsection 10.2.4. If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value ∞
FrSMNumWakeupPatterns	Integer	Configuration parameter, see subsection 10.2.4.
FrSMDelayStartupWithoutWakeup	Boolean	Configuration parameter, see subsection 10.2.4.
FrSMMinNumberOfColdstarter	Integer	Configuration parameter, see subsection 10.2.4.

7.3.5 Conditions

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

FrSM Condition	**Type**	**Description**
WUReason	Enum	If FrSMCheckWakeupReason is false, WUReason evaluates to NO_WU_BY_BUS. Otherwise if FrSMCheckWakeupReason is true, determine the wakeup reason by calling FrIf_GetTransceiverWUReason for each transceiver of the FlexRay cluster and check for FRTRCV_WU_BY_BUS and evaluate WUReason to • NO_WU_BY_BUS in case no wakeup has been detected. • PARTIAL_WU_BY_BUS in case the ECU is connected to both FlexRay channels of the cluster and wakeup has been detected for exactly one channel • ALL_WU_BY_BUS in case wakeup has been detected for all of the FlexRay channels of the cluster to which the ECU is connected.
t1_IsActive	boolean	Evaluates to true if t1 has been started and has not expired yet, otherwise to false
t3_IsNotActive	boolean	Evaluates to false if t3 is running and has not expired, otherwise to true.
t_TrcvStdbyDelay_IsActive	boolean	Evaluates to true if t_TrcvStdbyDelay has been started and has not expired yet, otherwise to false.

▽

△

wakeupFinished	boolean	Evaluates to false if the wakeup pattern transmission as defined in subsection 7.3.8 is still in progress, otherwise to true.
lowNumberOfColdstarters	boolean	Evaluates to true if FrIf_GetNumOfStartup Frames() is smaller than FrSMMinNumberOfColdstarter, otherwise to false.

7.3.6 Timers

The state machine description uses the following timers for each FlexRay cluster:

Timer	Description
t1	The timer t1 models the delay of clearing the coldstart inhibit mode (i.e. calling FrIf_Allow Coldstart). The duration of this timer can be statically configured with the configuration parameter Fr SMDurationT1.
t2	The timer t2 models the time difference after which the FrSM will repeat the startup of the FlexRay cluster. The duration of this timer can be statically configured with the configuration parameter Fr SMDurationT2.
t3	The timer t3 supervises the transition to FullCom. The duration of this timer can be statically configured with the configuration parameter FrSMDurationT3.
t4	The timer t4 ensures that a dual channel node will eventually clear its coldstart inhibit bit and become a leading coldstarter.
t_TrcvStdbyDelay	The timer t_TrcvStdbyDelay models the time difference after which the FrSM will reinitialize the FlexRay communication controllers and set the transceivers into STANDBY mode when FlexRay communication is stopped.

[SWS_FrSM_00142] ⌈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.⌋*()*

[SWS_FrSM_00143] ⌈If the duration FrSMDurationT2 of timer t2 is set to 0, the startup of the FlexRay cluster shall not be supervised.

Note, that no assumption is made whether any of the timers is implemented in software or hardware.⌋*()*

[SWS_FrSM_00209] ⌈If the configuration parameter FrSMTrcvStdbyDelay is not configured or set to 0, timer t_TrcvStdbyDelay shall not be started. Instead, the transition from state FRSM_HALT_REQ to FRSM_READY shall be executed immediately.⌋*()*

7.3.7 Functional Elements

The functionality being performed in the transitions of the state machine is partitioned into the following functional elements. I.e. the following table contains abbreviations used as actions in the FrSM state machine description, which reference one or more function calls visible at the interfaces of the FrSM module.

Functional Element	Description
FE_WAKEUP	Call FrIf_SendWUP for each controller of the FlexRay cluster.
FE_SET_WU_CHANNEL_INITIAL	In case of a single channel node, do nothing. In case of a dual channel node, call FrIf_SetWakeupChannel for each controller of the FlexRay cluster in order to set the wakeup channel to the channel A.
FE_SET_WU_CHANNEL_FORWARD	In case of a single channel node, do nothing. In case of a dual channel node, call FrIf_SetWakeupChannel for each controller of the FlexRay cluster in order to set the wakeup channel to the channel on which no wakeup has been detected while evaluating WUReason.
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 FrSMIsColdstartEcu is true.
FE_HALT	Call FrIf_HaltCommunication for each controller of the FlexRay cluster.
FE_TRCV_STANDBY	Call FrIf_SetTransceiverMode with FrIf_TrcvMode as FRTRCV_TRCVMODE_STANDBY for each transceiver of the FlexRay cluster.
FE_TRCV_NORMAL	In case the FrSM state machine is in state FRSM_ECU_ACTIVE, call FrIf_SetTransceiverMode with FrIf_TrcvMode as FRTRCV_TRCVMODE_NORMAL and FrIf_ClearTransceiverWakeup for each transceiver of the FlexRay cluster. In case the FrSM state machine is in state FRSM_ECU_PASSIVE, call FrIf_SetTransceiverMode with FrIf_TrcvMode as FRTRCV_TRCVMODE_RECEIVEONLY and FrIf_ClearTransceiverWakeup for each transceiver of the FlexRay cluster.
FE_START_FRIF	Set the FrIf state to ONLINE by calling FrIf_SetState with FrIf_StateTransition as FRIF_GOTO_ONLINE for the cluster.
FE_STOP_FRIF	Set the FrIf state to OFFLINE by calling FrIf_SetState with FrIf_StateTransition as 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_DEM_SYNC_LOSS	Report the status of the production error FRSM_E_CLUSTER_SYNC_LOSS as failed. If the name of an indication function (see subsection 8.6.3) is configured, call the indication function with the parameter SyncLossErrorStatus = true.
FE_DEM_SYNC_LOSS_PASSED	If the name of an indication function (see subsection 8.6.3) is configured, call the indication function with the parameter SyncLossErrorStatus = false. Additionally report the status of the production error FRSM_E_CLUSTER_SYNC_LOSS as passed.
FE_FULL_COM_IND	Indicate to the ComM that FullCom has been reached by calling ComM_BusSM_ModeIndication (FullCom)
FE_NO_COM_IND	Indicate to the ComM that FullCom has been left by calling ComM_BusSM_ModeIndication (NoCom).
FE_STARTUP_ERROR_IND	Call FrNm_StartupError.

7.3.8 Wakeup Pattern Transmission

[SWS_FrSM_00208] ⌈The FrSM shall repeat the transmission of wakeup patterns according to the configuration parameter FrSMNumWakeupPatterns. I.e. the FlexRay State Manager shall perform the following actions while being in state FRSM_WAKEUP:

- Set counter wakeupCounter to 1 when the state FRSM_WAKEUP is entered
- While wakeupCounter $<$ FrSMNumWakeupPatterns and busTrafficDetected = false:
  - Wait until the FlexRay controllers of the FlexRay cluster are in state FR_READY
  - When the FlexRay controllers are in state FR_READY, check vPOC!Wakeup Status of the FlexRay controllers and act as follows:
    - if vPOC!WakeupStatus = FR_WAKEUP_RECEIVED_HEADER $\vee$ FR_WAKEUP_RECEIVED_WUP: busTrafficDetected := true
    - else if vPOC!WakeupStatus = FR_WAKEUP_TRANSMITTED: wakeupTransmitted := true
    - else: wakeupTransmitted := false
- If busTrafficDetected = false and wakeupCounter < FrSMNumWakeupPatterns, execute FE_WAKEUP
- Increment the wakeupCounter

If any of the FlexRay controllers enters the HALT state due to an error condition, the wakeup pattern transmission shall be aborted and the wakeupFinished condition shall evaluate to true.⌋*()*

7.3.9 Transitions

The following diagram shows the FrSM state machine.

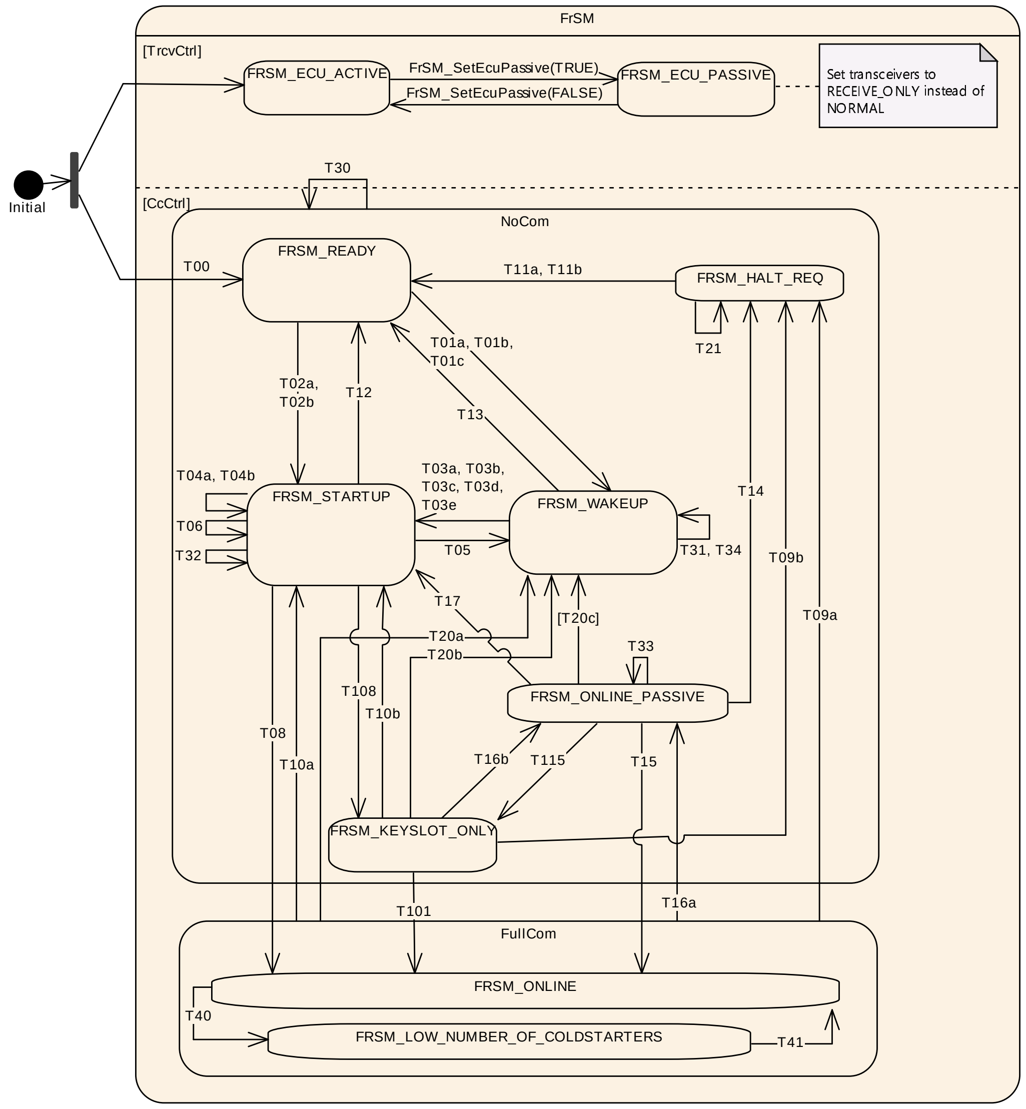


Figure 7.1: FrSM state machine

Note that the states are described in subsection 7.3.2.

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. Each row of the table contains a requirement which should be interpreted as follows. If the FrSM module is in the source state of the transition in column "Transition" as defined in [SWS_FrSM_00093] and when the condition in column "Event [Condition]" holds and if the event in column "Event [Condition]" occurs, then the actions in column "Actions" shall be executed and afterwards the FrSM module shall change its state to the target state of the transition in column "Transition" as defined in [SWS_FrSM_00093].

In case different actions have to be performed in a transition T, there can be multiple rows in the table. The rows are denoted as T (a), T (b) etc. in this case. Note that the conditions ensure that only one of the possibilities matches.

[SWS_FrSM_00093] ⌈The FrSM shall execute the actions of the transition in the order that is defined in the table within [SWS_FrSM_00105].⌋*()*

[SWS_FrSM_00145] ⌈After every transition to a different state, the FrSM shall inform the BswM by calling BswM_FrSM_CurrentState.⌋*()*

[SWS_FrSM_00105] ⌈

Transi-tion	Event [Condition]	Actions
T00	FrSM_Init()	FE_CONFIG
T01 (a)	[reqComMode = FullCom ∧ FrSMIsWakeupEcu ∧ WUReason = NO_WU_BY_BUS ∧ ¬ FrSMIsDualChannelNode]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := SingleChannelWakeup wakeupTransmitted := false FE_WAKEUP start t1 start t3
T01 (b)	[reqComMode = FullCom ∧ FrSMIsWakeupEcu ∧ WUReason = NO_WU_BY_BUS ∧ FrSMIsDualChannelNode]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := DualChannelWakeup FE_SET_WU_CHANNEL_INITIAL wakeupTransmitted := false FE_WAKEUP start t3 start t4
T01 (c)	[reqComMode = FullCom ∧ FrSMIsWakeupEcu ∧ WUReason = PARTIAL_WU_BY_BUS]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := DualChannelWakeupForward FE_SET_WU_CHANNEL_FORWARD FE_WAKEUPwakeupTransmitted := false FE_WAKEUP start t3
T02 (a)	[reqComMode = FullCom ∧ (¬ FrSMIsWakeupEcu ∨ WUReason = ALL_WU_BY_BUS) ∧ ¬ FrSMDelayStartupWithoutWakeup]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := NoWakeup FE_START FE_ALLOW_COLDSTART start t2 start t3
T02 (b)	[reqComMode = FullCom ∧ (¬ FrSMIsWakeupEcu ∨ WUReason = ALL_WU_BY_BUS) ∧ FrSMDelayStartupWithoutWakeup]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := NoWakeup FE_START start t1 start t2 start t3
T03 (a)	[wakeupFinished ∧ reqComMode = FullCom ∧ FrSMNumWakeupPatterns = 1 ∧ wakeupType = SingleChannelWakeup]	FE_START cancel t1 start t1 start t2

T03 (b)	[wakeupFinished
	∧ reqComMode = FullCom
	∧ FrSMNumWakeupPatterns > 1
	∧ wakeupTransmitted
	∧ wakeupType = SingleChannelWakeup
	]
		FE_START
		start t2
		IF t1_IsActive: cancel t1
		ELSE: FE_ALLOW_COLDSTART

▽

△

Transi-tion	Event [Condition]	Actions
T03 (c)	[wakeupFinished ∧ reqComMode = FullCom ∧ FrSMNumWakeupPatterns > 1 ∧ ¬ wakeupTransmitted ∧ wakeupType = SingleChannelWakeup]	FE_START start t2
T03 (d)	[wakeupFinished ∧ reqComMode = FullCom ∧ wakeupType = DualChannelWakeup ∧ wakeupTransmitted ∧ ¬ busTrafficDetected]	FE_START start t2
T03 (e)	[wakeupFinished ∧ reqComMode = FullCom ∧ wakeupType = DualChannelWakeupForward]	FE_START FE_ALLOW_COLDSTART start t2 cancel t4
T04 (a)	t1 [reqComMode = FullCom ∧ vPOC!State ≠ Normal Active]	FE_ALLOW_COLDSTART
T04 (b)	t4 [reqComMode = FullCom ∧ wakeupType = DualChannelWakeup ∧ vPOC!State ≠ Normal Active]	FE_ALLOW_COLDSTART
T05	t2 [startupCounter ≤ FrSMStartupRepetitionsWithWakeup ∧ reqComMode = FullCom ∧ wakeupType ≠ NoWakeup ∧ vPOC!State ≠ Normal Active]	FE_CONFIG FE_WAKEUP startupCounter := startupCounter + 1 start t4 (dual channel node only)
T06	t2 [(FrSMStartupRepetitionsWithWakeup < startupCounter ∧ wakeupType = NoWakeup) ∧ startupCounter ≤ FrSMStartup Repetitions ∧ reqComMode = FullCom ∧ vPOC!State ≠ Normal Active]	FE_CONFIG FE_START FE_ALLOW_COLDSTART startupCounter := startupCounter + 1 start t2
T08	[vPOC!State = Normal Active ∧ ¬ vPOCFreeze ∧ vPOCSlotMode = AllSlots ∧ reqComMode = FullCom]	cancel t1 cancel t2 FE_START_FRIF FE_DEM_STATUS_PASSED FE_DEM_SYNC_LOSS_PASSED FE_FULL_COM_IND cancel t3

▽

△

Transi-tion	Event [Condition]	Actions
T108	[vPOC!State = Normal Active ∧ ¬ vPOCFreeze ∧ vPOCSlotMode ≠ AllSlots ∧ reqComMode = FullCom]	cancel t1 cancel t2 FE_START_FRIF FE_DEM_STATUS_PASSED FE_DEM_SYNC_LOSS_PASSED cancel t3
T09a	FrSM_RequestComMode() [reqComMode = NoCom]	FE_STOP_FRIF FE_HALT FE_NO_COM_IND
T09b	FrSM_RequestComMode() [reqComMode = NoCom]	FE_STOP_FRIF FE_HALT
T10a	[(vPOC!State = Halt ∨ vPOCFreeze) ∧ reqComMode = FullCom ∧ (FrSMCheckWakeupReason ∨ ¬ FrSMIsWakeupEcu)]	FE_DEM_SYNC_LOSS FE_STOP_FRIF FE_NO_COM_IND FE_CONFIG FE_START startupCounter := 1 start t2 start t3
T10b	[(vPOC!State = Halt ∨ vPOCFreeze) ∧ reqComMode = FullCom ∧ (FrSMCheckWakeupReason ∨ ¬ FrSMIsWakeupEcu)]	FE_DEM_SYNC_LOSS FE_STOP_FRIF FE_CONFIG FE_START startupCounter := 1 start t2 start t3
T101	[vPOC!State = Normal Active ∧ ¬ vPOCFreeze ∧ vPOCSlotMode = AllSlots]	FE_FULL_COM_IND
T11a	t_TrcvStdbyDelay[]	FE_TRCV_STANDBY FE_CONFIG
T11b	[(vPOC!State = Halt ∨ vPOCFreeze) ∧ reqComMode = FullCom]	cancel t_TrcvStdbyDelay FE_TRCV_STANDBY FE_CONFIG
T12	[reqComMode = NoCom]	cancel t1 cancel t2 cancel t3 FE_DEM_SYNC_LOSS_PASSED FE_TRCV_STANDBY FE_CONFIG
T13	[reqComMode = NoCom]	FE_DEM_SYNC_LOSS_PASSED FE_TRCV_STANDBY FE_CONFIG cancel t3 cancel t1

▽

△

Transi-tion	Event [Condition]	Actions
T14	FrSM_RequestComMode() [reqComMode = NoCom]	FE_DEM_SYNC_LOSS_PASSED FE_HALT cancel t3

T15	[vPOC!State = Normal Active
	  ∧ ¬ vPOCFreeze
	  ∧ vPOCSlotMode = AllSlots
	]
		FE_DEM_SYNC_LOSS_PASSED
		FE_START_FRIF
		FE_FULL_COM_IND
		cancel t3

T115	[vPOC!State = Normal Active
	  ∧ ¬ vPOCFreeze
	  ∧ vPOCSlotMode ≠ AllSlots
	]
		FE_DEM_SYNC_LOSS_PASSED
		FE_START_FRIF
		cancel t3

T16a	[vPOC!State = Normal Passive ∧ ¬ vPOCFreeze]	FE_DEM_SYNC_LOSS FE_STOP_FRIF FE_NO_COM_IND start t3
T16b	[vPOC!State = Normal Passive ∧ ¬ vPOCFreeze]	FE_DEM_SYNC_LOSS FE_STOP_FRIF start t3

T17	[(vPOC!State = Halt ∨ vPOCFreeze)
	  ∧ reqComMode = FullCom
	  ∧ (FrSMCheckWakeupReason
	    ∨ ¬ FrSMIsWakeupEcu)
	]
		FE_CONFIG
		wakeupType := NoWakeup
		FE_START
		startupCounter := 1
		start t2

T20a	[(vPOC!State = Halt ∨ vPOCFreeze)
	  ∧ reqComMode = FullCom
	  ∧ ¬ FrSMCheckWakeupReason
	  ∧ FrSMIsWakeupEcu
	]
		wakeupType := SingleChannelWakeup
		FE_DEM_SYNC_LOSS
		FE_STOP_FRIF
		FE_NO_COM_IND
		FE_CONFIG
		FE_WAKEUP
		startupCounter := 1
		start t1
		start t3

T20b	[(vPOC!State = Halt ∨ vPOCFreeze)
	  ∧ reqComMode = FullCom
	  ∧ ¬ FrSMCheckWakeupReason
	  ∧ FrSMIsWakeupEcu
	]
		wakeupType := SingleChannelWakeup
		FE_DEM_SYNC_LOSS
		FE_STOP_FRIF
		FE_CONFIG
		FE_WAKEUP
		startupCounter := 1
		start t1
		start t3

T20c	[(vPOC!State = Halt ∨ vPOCFreeze)
	  ∧ reqComMode = FullCom
	  ∧ ¬ FrSMCheckWakeupReason
	  ∧ FrSMIsWakeupEcu
	]
		wakeupType := SingleChannelWakeup
		FE_CONFIG
		FE_WAKEUP
		startupCounter := 1
		start t1
		start t3

T21	[(vPOC!State = Halt ∨ vPOCFreeze) ∧ ¬ t_TrcvStdbyDelay_IsActive]	start t_TrcvStdbyDelay

▽

△

Transition	Event [Condition]	Actions
T30	t3[]	FE_DEM_STATUS_FAILED FE_STARTUP_ERROR_IND
T31	[t3_IsNotActive]	FE_STARTUP_ERROR_IND
T32	[t3_IsNotActive]	FE_STARTUP_ERROR_IND
T33	[t3_IsNotActive]	FE_STARTUP_ERROR_IND

T34	[wakeupFinished
	∧ reqComMode = FullCom
	∧ FrSMNumWakeupPatterns > 1
	∧ (¬ wakeupTransmitted
	∨ busTrafficDetected)
	∧ wakeupType = DualChannelWakeup
	]
		startupCounter := 1
		wakeupType := DualChannelWakeupForward
		FE_SET_WU_CHANNEL_FORWARD
		wakeupTransmitted := false
		busTrafficDetected := false
		FE_WAKEUP
		start t1
		start t3

T40	[lowNumberOfColdstarters]	
T41	[lowNumberOfColdstarters]	

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

⌋()

Note: If synchronization is lost after FullCom has been reached, the FrSM module will 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.

Note: If resynchronization cannot be achieved before t2 expires ([SWS_FrSM_00105] T08 and T108), the same wakeup and startup procedure as for the initial synchronization will 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 FrSMStartupRepetitionsWithWakeup, the startup procedure will be repeated including the wakeup.
- If startupCounter exceeds the threshold FrSMStartupRepetitionsWithWakeup but does not exceed the threshold FrSMStartupRepetitions, the startup procedure will be repeated without wakeup.

Note: When the timer t3 expires, the FrSM will report the production error FRSM_E_CLUSTER_STARTUP.

Note: After timer t3 has expired, the FrSM will call FrNm_StartupError until either synchronisation has been achieved or NoCom is requested ([SWS_FrSM_00105] all transitions where t3 is cancelled).

Note: When the counter startupCounter exceeds the threshold FrSMStartupRepetitions, an ECU that has been configured as a coldstart node will stop performing coldstart attempts. However, if another ECU performs a coldstart, the ECU will join the coldstart.

Note: If no threshold FrSMStartupRepetitions has been configured, an ECU that has been configured as a coldstart node will not stop performing coldstart attempts until either synchronisation 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.

[SWS_FrSM_00149] ⌈When a call of a function of the FrIf API returns a failure (e.g. E_NOT_OK), the FrSM shall ignore this return value and continue with the transition.⌋ *()*

Rationale: When the FrIf returns E_NOT_OK in a production environment, a production error has been reported to DEM. This will usually trigger the reinitialization of the FlexRay stack.

7.4 Configuration description

The FrSM configuration tool reads the ECU configuration description of the FrIf as the mapping of controllers to clusters is contained in the FrIf configuration description.

7.5 Error Classification

Section "Error Handling" of the document [2] "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

7.5.1 Development Errors

[SWS_FrSM_91001] Definiton of development errors in module FrSM ⌈

Type of error	*Related error code*	*Error value*
Invalid pointer in parameter list. In case of this error, the API service shall return immediately without any further action, beside reporting this development error.	FRSM_E_PARAM_POINTER	0x01
Invalid network handle parameter	FRSM_E_INV_HANDLE	0x02
FrSM module was not initialized	FRSM_E_UNINIT	0x03
Invalid communication mode requested	FRSM_E_INV_MODE	0x04
Initialization failed	FRSM_INIT_FAILED	0x05

⌋*()*

7.5.2 Runtime Errors

There are no runtime errors.

7.5.3 Transient Faults

There are no transient faults.

7.5.4 Production Errors

7.5.4.1 FRSM_E_CLUSTER_STARTUP

[SWS_FrSM_00300] Production Error definition for cluster startup failure ⌈

Error Name:	FRSM_E_CLUSTER_STARTUP	
Short Description:	FlexRay cluster startup failure.	
Long Description:	FlexRay controller has not reached the state normal active within the configured time after FlexRay startup.	
Recommended DTC:	Assigned by DEM	
Detection Criteria:	Fail	FlexRay controller has not reached the state normal active within the time t3
	Pass	FlexRay controller has reached the state normal active
Secondary Parameters:	None	
Time Required:	FrSMDurationT3	
Monitor Frequency	Continuous	
MIL illumniation:	Assigned by DEM	

⌋*(SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00458)*

7.5.4.2 FRSM_E_CLUSTER_SYNC_LOSS

[SWS_FrSM_00301] Production Error definition for synchronization loss ⌈

Error Name:	FRSM_E_CLUSTER_SYNC_LOSS	
Short Description:	FlexRay synchronization loss.	
Long Description:	FlexRay controller has lost synchronization after successful startup.	
Recommended DTC:	Assigned by DEM	
Detection Criteria:	Fail	FlexRay controller has lost synchronization after it has reached state normal active.
	Pass	FlexRay controller has reached the state normal active or the request for FlexRay communication has been released.
Secondary Parameters:	None	
Time Required:	Depends on FlexRay configuration.	
Monitor Frequency	Continuous	
MIL illumniation:	Assigned by DEM	

⌋*(SRS_BSW_00385, SRS_BSW_00386, SRS_BSW_00458)*

7.5.5 Extended Production Errors

There are no extended production errors.

8 API specification

8.1 Imported types

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

[SWS_FrSM_00095] Definition of imported datatypes of module FrSM ⌈

Module	***Header File***	***Imported Type***
ComM	Rte_ComM_Type.h	ComM_ModeType
ComStack_Types	ComStack_Types.h	NetworkHandleType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Fr	Fr_GeneralTypes.h	Fr_ChannelType
	Fr_GeneralTypes.h	Fr_ErrorModeType
	Fr_GeneralTypes.h	Fr_POCStateType
	Fr_GeneralTypes.h	Fr_POCStatusType
	Fr_GeneralTypes.h	Fr_SlotModeType
	Fr_GeneralTypes.h	Fr_StartupStateType
	Fr_GeneralTypes.h	Fr_WakeupStatusType
FrIf	FrIf.h	FrIf_StateTransitionType
FrTrcv	Fr_GeneralTypes.h	FrTrcv_TrcvModeType
	Fr_GeneralTypes.h	FrTrcv_TrcvWUReasonType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

⌋*()*

8.2 Type definitions

8.2.1 FrSM_ConfigType

[SWS_FrSM_00198] Definition of datatype FrSM_ConfigType ⌈

Name	FrSM_ConfigType
Kind	Structure
Description	This type contains the implementation-specific post build time configuration structure that is for Fr SM_Init.
Available via	FrSM.h

⌋*()*

8.2.2 FrSM_BswM_StateType

[SWS_FrSM_00199] Definition of datatype FrSM_BswM_StateType ⌈

Name	FrSM_BswM_StateType		
Kind	Enumeration		
Range	FRSM_BSWM_READY	0x00	–
	FRSM_BSWM_READY_ECU_PASSIVE	0x01	–
	FRSM_BSWM_STARTUP	0x02	–
	FRSM_BSWM_STARTUP_ECU_PASSIVE	0x03	–
	FRSM_BSWM_WAKEUP	0x04	–
	FRSM_BSWM_WAKEUP_ECU_PASSIVE	0x05	–
	FRSM_BSWM_HALT_REQ	0x06	–
	FRSM_BSWM_HALT_REQ_ECU_PASSIVE	0x07	–
	FRSM_BSWM_KEYSLOT_ONLY	0x08	–
	FRSM_BSWM_KEYSLOT_ONLY_ECU_PASSIVE	0x09	–
	FRSM_BSWM_ONLINE	0x0A	–
	FRSM_BSWM_ONLINE_ECU_PASSIVE	0x0B	–
	FRSM_BSWM_ONLINE_PASSIVE	0x0C	–
	FRSM_BSWM_ONLINE_PASSIVE_ECU_PASSIVE	0x0D	–
	FRSM_LOW_NUMBER_OF_COLDSTARTERS	0x0E	–
	FRSM_LOW_NUMBER_OF_COLDSTARTERS_ECU_PASSIVE	0x0F	–
Description	This type defines the states that are reported to the BswM using BswM_FrSM_CurrentState.		
Available via	FrSM.h		

⌋*()*

8.3 Function definitions

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

8.3.1 FrSM_Init

[SWS_FrSM_00013] Definition of API function FrSM_Init ⌈

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.	
Available via	FrSM.h	

⌋*(SRS_BSW_00405, SRS_BSW_00438)*

[SWS_FrSM_00126] ⌈The `FrSM_Init` function shall initialize the state machines for all FlexRay clusters and set them into the state `FRSM_READY`, i.e. perform transition T00.⌋*(SRS_BSW_00438, SRS_BSW_00101)*

[SWS_FrSM_00127] ⌈The `FrSM_Init` function shall internally store the configuration data address to enable subsequent API calls to access the configuration data.⌋*(SRS_-BSW_00438, SRS_BSW_00483)*

[SWS_FrSM_00128] ⌈If development error detection is enabled (FrSMDevErrorDetect is ON), the `FrSM_Init` function shall remember internally the successful initialization for other API functions to check for proper module initialization.⌋*(SRS_BSW_00438)*

8.3.2 FrSM_RequestComMode

[SWS_FrSM_00020] Definition of API function FrSM_RequestComMode ⌈

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.	
Available via	FrSM.h	

⌋*()*

[SWS_FrSM_00021] ⌈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 reqComMode.

[SWS_FrSM_00022] ⌈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, FRSM_KEYSLOT_ONLY, FRSM_LOW_NUMBER_OF_COLDSTARTERS or FRSM_ONLINE_PASSIVE), the FrSM_RequestComMode function shall immediately process the corresponding transition of the state machine (see [SWS_FrSM_00093]).⌋*()*

Rationale of [SWS_FrSM_00022]: 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.

[SWS_FrSM_00141] ⌈If ComM_Mode has the value COMM_SILENT_COMMUNICATION, the FrSM shall not store the requested communication mode and return E_NOT_OK. In case development error detection is enabled, the FrSM shall additionally raise the development error code FRSM_E_INV_MODE.⌋*(SRS_BSW_00350)*

[SWS_FrSM_00018] ⌈If development error detection is enabled and the parameter NetworkHandle has an invalid value, the FrSM_RequestComMode function shall raise the development error code FRSM_E_INV_HANDLE and the FrSM_RequestComMode function shall return E_NOT_OK.⌋*(SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350)*

[SWS_FrSM_00019] ⌈If development error detection is enabled and the parameter ComM_Mode has an invalid value, the FrSM_RequestComMode function shall raise the development error code FRSM_E_INV_MODE and the FrSM_RequestComMode function shall return E_NOT_OK.⌋*(SRS_BSW_00350)*

[SWS_FrSM_00061] ⌈If development error detection is enabled and the FrSM module has not been initialized using FrSM_Init, the FrSM_RequestComMode function shall raise the development error code FRSM_E_UNINIT and the function FrSM_RequestComMode shall return E_NOT_OK.⌋*(SRS_BSW_00406, SRS_BSW_00350)*

8.3.3 FrSM_GetCurrentComMode

[SWS_FrSM_00024] Definition of API function FrSM_GetCurrentComMode ⌈

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 initialized using FrSM_Init.
Description	This API function can be used to determine the current communication mode of a FlexRay cluster.	
Available via	FrSM.h	

⌋*()*

[SWS_FrSM_00025] ⌈The `FrSM_GetCurrentComMode` function shall write the current communication mode of the corresponding FlexRay cluster into the given memory location.⌋*()*

[SWS_FrSM_00026] ⌈The `FrSM_GetCurrentComMode` function shall determine the communication mode as follows:

- If the `FrSM` state machine for the FlexRay cluster determined by NetworkHandle is in state `FRSM_ONLINE` or `FRSM_LOW_NUMBER_OF_COLDSTARTERS`, the communication mode is COMM_FULL_COMMUNICATION.
- In any other case, the communication mode is COMM_NO_COMMUNICATION.

⌋*(SRS_BSW_00483)*

[SWS_FrSM_00027] ⌈If development error detection is enabled and the parameter NetworkHandle has an invalid value, the `FrSM_GetCurrentComMode` function shall raise the development error code FRSM_E_INV_HANDLE and the `FrSM_GetCurrentComMode` function shall return E_NOT_OK.⌋*(SRS_BSW_00350)*

[SWS_FrSM_00028] ⌈If development error detection is enabled and the parameter ComM_ModePtr equals NULL_PTR, the `FrSM_GetCurrentComMode` function shall raise the development error code FRSM_E_PARAM_POINTER and the `FrSM_GetCurrentComMode` function shall return E_NOT_OK.⌋*(SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350)*

[SWS_FrSM_00060] ⌈If development error detection is enabled and the `FrSM` module has not been initialized using `FrSM_Init`, the `FrSM_GetCurrentComMode` function shall raise the development error code FRSM_E_UNINIT and the `FrSM_GetCur-`

rentComMode function shall return E_NOT_OK.⌋*(SRS_BSW_00406, SRS_BSW_-00350)*

8.3.4 FrSM_GetVersionInfo

[SWS_FrSM_00029] Definition of API function FrSM_GetVersionInfo ⌈

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.	
Available via	FrSM.h	

⌋*(SRS_BSW_00407)*

8.3.5 FrSM_AllSlots

[SWS_FrSM_00172] Definition of API function FrSM_AllSlots ⌈

Service Name	FrSM_AllSlots	
Syntax	`Std_ReturnType FrSM_AllSlots ( NetworkHandleType NetworkHandle )`	
Service ID [hex]	0x05	
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.
Parameters (inout)	None	
Parameters (out)	None	

▽

△

Return value	Std_ReturnType	`E_OK`: Request accepted `E_NOT_OK`: Request not accepted
Description	This API function can be used to leave the KeySlotOnlyMode.	
Available via	FrSM.h	

⌋*()*

[SWS_FrSM_00197] ⌈The `FrSM_AllSlots` function shall be pre compile time configurable ON/OFF by the configuration parameter FrSMAllSlotsSupport⌋*()*

[SWS_FrSM_00171] ⌈The `FrSM_AllSlots` function shall call FrIf_AllSlots for each controller of the FlexRay cluster. It shall return E_OK if each of these calls returned E_OK, otherwise `FrSM_AllSlots` shall return E_NOT_OK.⌋*()*

[SWS_FrSM_00168] ⌈If development error detection is enabled and the parameter NetworkHandle has an invalid value, the `FrSM_AllSlots` function shall raise the development error code FRSM_E_INV_HANDLE and the `FrSM_AllSlots` function shall return E_NOT_OK.⌋*(SRS_BSW_00369, SRS_BSW_00323, SRS_BSW_00350)*

[SWS_FrSM_00169] ⌈If development error detection is enabled and the `FrSM` module has not been initialized using `FrSM_Init`, the `FrSM_AllSlots` function shall raise the development error code FRSM_E_UNINIT and the `FrSM_AllSlots` function shall return E_NOT_OK.⌋*(SRS_BSW_00406, SRS_BSW_00350)*

8.3.6 FrSM_SetEcuPassive

[SWS_FrSM_00174] Definition of API function FrSM_SetEcuPassive ⌈

Service Name	FrSM_SetEcuPassive	

Syntax	`Std_ReturnType FrSM_SetEcuPassive (`
	`  boolean FrSM_Passive`
	`)`	

Service ID [hex]	0x06	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	FrSM_Passive	This parameter determines whether all FlexRay clusters are set to passive, i.e. receive only.
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	`E_OK`: Request accepted `E_NOT_OK`: Request not accepted
Description	This API function can be used to set all FlexRay clusters of the ECU to a receive only mode.	
Available via	FrSM.h	

⌋*()*

[SWS_FrSM_00177] ⌈The `FrSM_SetEcuPassive` function shall set the state of all `FrSM` state machines to `FRSM_ECU_PASSIVE` if the parameter FrSM_Passive evaluates to true, otherwise it shall set the state of all FrSM state machines to FRSM_ECU_-ACTIVE.⌋()

[SWS_FrSM_00178] ⌈If the state machine of a FlexRay cluster is not in state FRSM_-READY (i.e. the transceivers of the FlexRay cluster are not in standby mode), the function shall execute FE_TRCV_NORMAL for this cluster.⌋()

[SWS_FrSM_00179] ⌈If development error detection is enabled and the FrSM module has not been initialized using FrSM_Init, the FrSM_SetEcuPassive function shall raise the development error code FRSM_E_UNINIT and the FrSM_SetEcuPassive function shall return E_NOT_OK.⌋*(SRS_BSW_00406, SRS_BSW_00350)*

8.4 Callback notifications

The FrSM does not provide any call-back API services to other BSW modules.

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_<FrSMCluster.ShortName>

[SWS_FrSM_00118] Definition of scheduled function FrSM_MainFunction_<FrSMCluster.ShortName> ⌈

Service Name	FrSM_MainFunction_<FrSMCluster.ShortName>

Syntax	`void FrSM_MainFunction_<FrSMCluster.ShortName> (`
	`  void`
	`)`

Service ID [hex]	0x80
Description	–
Available via	SchM_FrSM.h

⌋*(SRS_BSW_00373)*

[SWS_FrSM_00047] ⌈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).

[SWS_FrSM_00192] ⌈If the optional configuration parameter FrSMMinNumberOfColdstarter is configured, the FrSM_MainFunction shall determine the number startup frames by calling FrIf_GetNumOfStartupFrames.⌋()

[SWS_FrSM_00048] ⌈After determining the POC status and optionally the number of startup frames, the FrSM_MainFunction shall process the state machine of the corresponding cluster.⌋*()*

Note: 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 Flex Ray 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_-MainFunction 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.

[SWS_FrSM_00181] ⌈If the FrSM module has not been initialized using FrSM_Init, the FrSM_MainFunction function shall shall return immediately without performing any functionality and without raising any errors.⌋*(SRS_BSW_00450)*

8.6 Expected interfaces

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

8.6.1 Mandatory interfaces

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

[SWS_FrSM_00096] Definition of mandatory interfaces in module FrSM ⌈

API Function	***Header File***	***Description***
BswM_FrSM_CurrentState	BswM_FrSM.h	Function called by FrSM to indicate its current state.
ComM_BusSM_ModeIndication	ComM.h	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 and BswM.
FrIf_AllowColdstart	FrIf.h	Wraps the FlexRay Driver API function Fr_Allow Coldstart().
FrIf_ClearTransceiverWakeup	FrIf.h	Wraps the FlexRay Transceiver Driver API function FrTrcv_ClearTransceiverWakeup(). The enum value "FR_CHANNEL_AB" shall not be used.

▽

△

API Function	*Header File*	*Description*
FrIf_ControllerInit	FrIf.h	Initialized a FlexRay CC.
FrIf_GetPOCStatus	FrIf.h	Wraps the FlexRay Driver API function Fr_Get POCStatus().
FrIf_GetTransceiverWUReason	FrIf.h	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverWUReason(). The enum value "FR_CHANNEL_AB" shall not be used.
FrIf_HaltCommunication	FrIf.h	Wraps the FlexRay Driver API function Fr_Halt Communication().
FrIf_SendWUP	FrIf.h	Wraps the FlexRay Driver API function Fr_Send WUP().
FrIf_SetState	FrIf.h	Requests FrIf state machine transition.
FrIf_SetTransceiverMode	FrIf.h	Wraps the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode(). The enum value "FR_CHANNEL_AB" shall not be used.
FrIf_StartCommunication	FrIf.h	Wraps the FlexRay Driver API function Fr_Start Communication().

⌋*()*

8.6.2 Optional interfaces

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

[SWS_FrSM_00097] Definition of optional interfaces in module FrSM ⌈

API Function	*Header File*	*Description*
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value. This API will be available only if ({Dem/Dem ConfigSet/DemEventParameter/DemEvent ReportingType} == STANDARD_REPORTING)
Det_ReportError	Det.h	Service to report development errors.
FrIf_AllSlots	FrIf.h	Wraps the FlexRay Driver API function Fr_AllSlots
FrIf_GetNumOfStartupFrames	FrIf.h	Wraps the FlexRay Driver API function Fr_GetNum OfStartupFrames and gets a list of the current number of startup frames seen on the cluster. See variable vStartupPairs of [12] for details.
FrIf_GetWakeupRxStatus	FrIf.h	Wraps the FlexRay Driver API function Fr_Get WakeupRxStatus and gets the wakeup received information from the FlexRay controller.
FrIf_SetWakeupChannel	FrIf.h	Wraps the FlexRay Driver API function Fr_Set WakeupChannel(). The enum value "FR_ CHANNEL_AB" shall not be used.
FrNm_StartupError	FrNm.h	This function is called by the FrSM when synchronization of the FlexRay cluster could not be achieved.

⌋*()*

8.6.3 Configurable interfaces

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

8.6.3.1 <Cdd>_SyncLossErrorIndication

[SWS_FrSM_00190] Definition of configurable interface <Cdd>_SyncLossError Indication ⌈

Service Name	<Cdd>_SyncLossErrorIndication	
Syntax	`void <Cdd>_SyncLossErrorIndication ( NetworkHandleType NetworkHandle, boolean SyncLossErrorStatus )`	
Sync/Async	Synchronous	
Reentrancy	Reentrant for different FlexRay clusters	
Parameters (in)	NetworkHandle	Handle of FlexRay cluster
	SyncLossErrorStatus	true: ECU lost synchronization to the FlexRay cluster. false: ECU can synchronize to the FlexRay cluster or request for full communication has been released after the ECU lost its synchronization to the FlexRay cluster.
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	This function is called with parameter SyncLossErrorStatus = true when the ECU loses its synchronization to the FlexRay cluster. The function is called with parameter SyncLossError Status = false either when the ECU can synchronize to the FlexRay cluster or when the request for full communication has been released after the ECU lost its synchronization to the FlexRay cluster.	
Available via	FrSM_Externals.h	

⌋*()*

The name of this function can be configured using the configuration parameter FrSM SyncLossErrorIndicationName (see subsection 10.2.3). The `FrSM` will call this function when the ECU looses its synchronization to the FlexRay cluster, after it could synchronize to the FlexRay cluster or when the FullCom request is released after the ECU lost its synchronization to the FlexRay cluster.

8.7 Service Interfaces

`FrSM` does not provide any service interfaces.

9 Sequence diagrams

9.1 Initialization

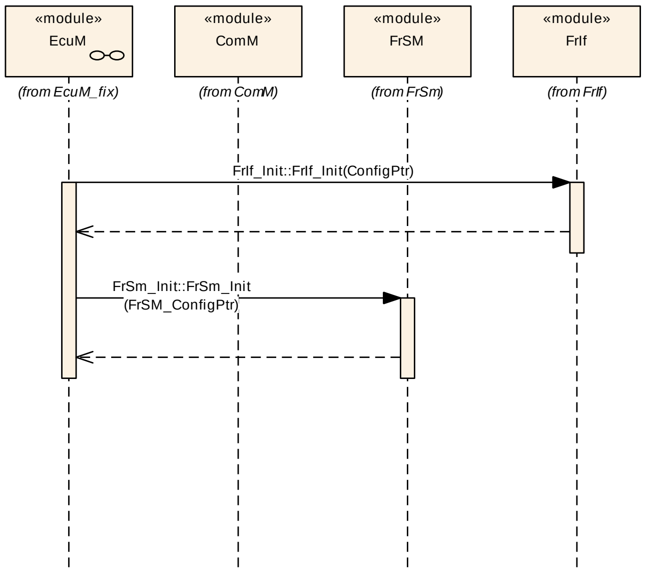


Figure 9.1: Initialization

9.2 Single Channel Wakeup

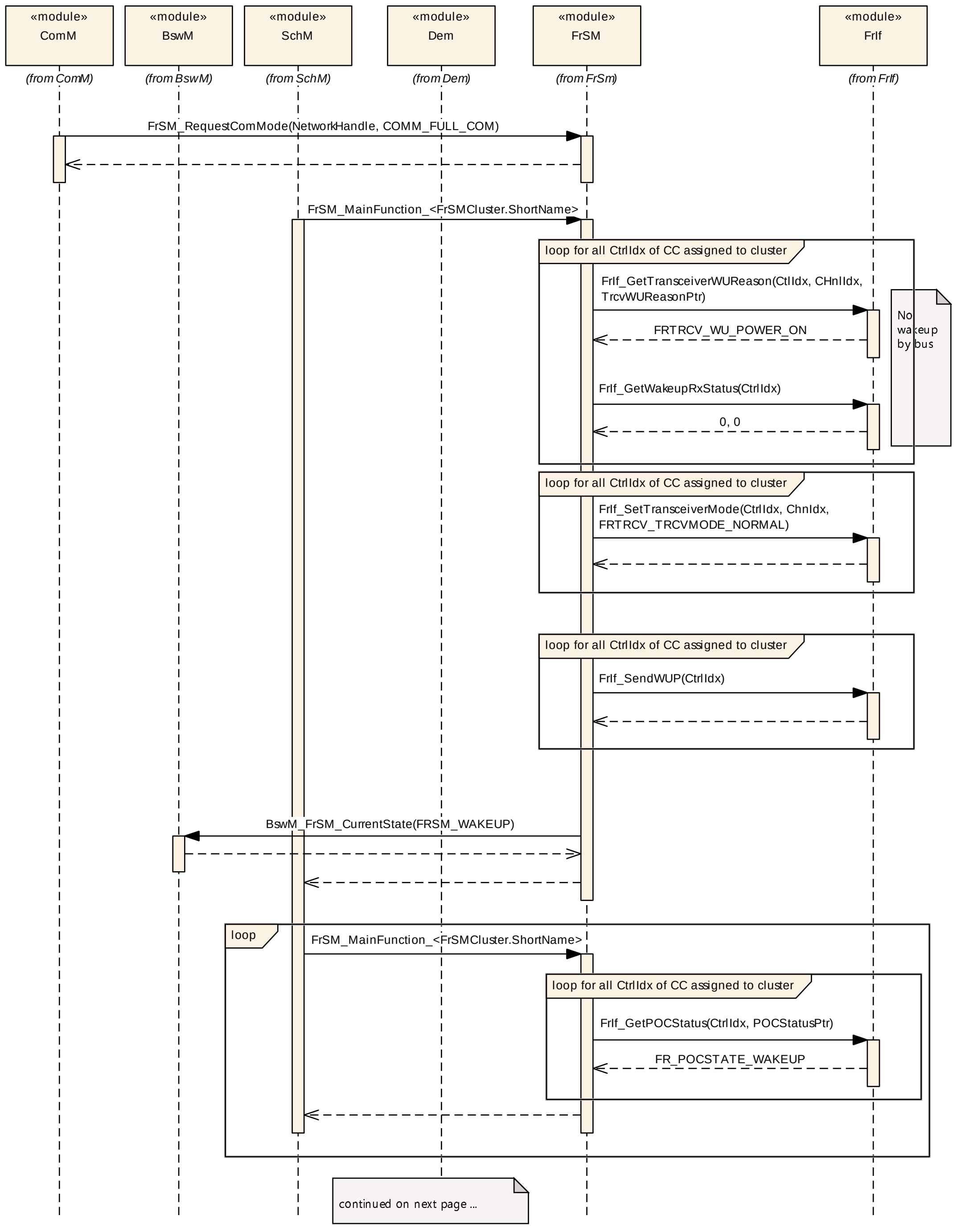

«module» ComM (from ComM)
«module» BswM (from BswM)
«module» SchM (from SchM)
«module» Dem (from Dem)
«module» FrSM (from FrSm)
«module» FrIf (from FrIf)
FrSM_RequestComMode(NetworkHandle, COMM_FULL_COM)
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetTransceiverWUReason(CtlIdx, CHnlIdx, TrcvWUReasonPtr)
FRTRCV_WU_POWER_ON
No wakeup by bus
FrIf_GetWakeupRxStatus(CtrlIdx)
0, 0
loop for all CtrlIdx of CC assigned to cluster
FrIf_SetTransceiverMode(CtrlIdx, ChnIdx, FRTRCV_TRCVMODE_NORMAL)
loop for all CtrlIdx of CC assigned to cluster
FrIf_SendWUP(CtrlIdx)
BswM_FrSM_CurrentState(FRSM_WAKEUP)
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_WAKEUP
continued on next page ...

Figure 9.2: continued on next page

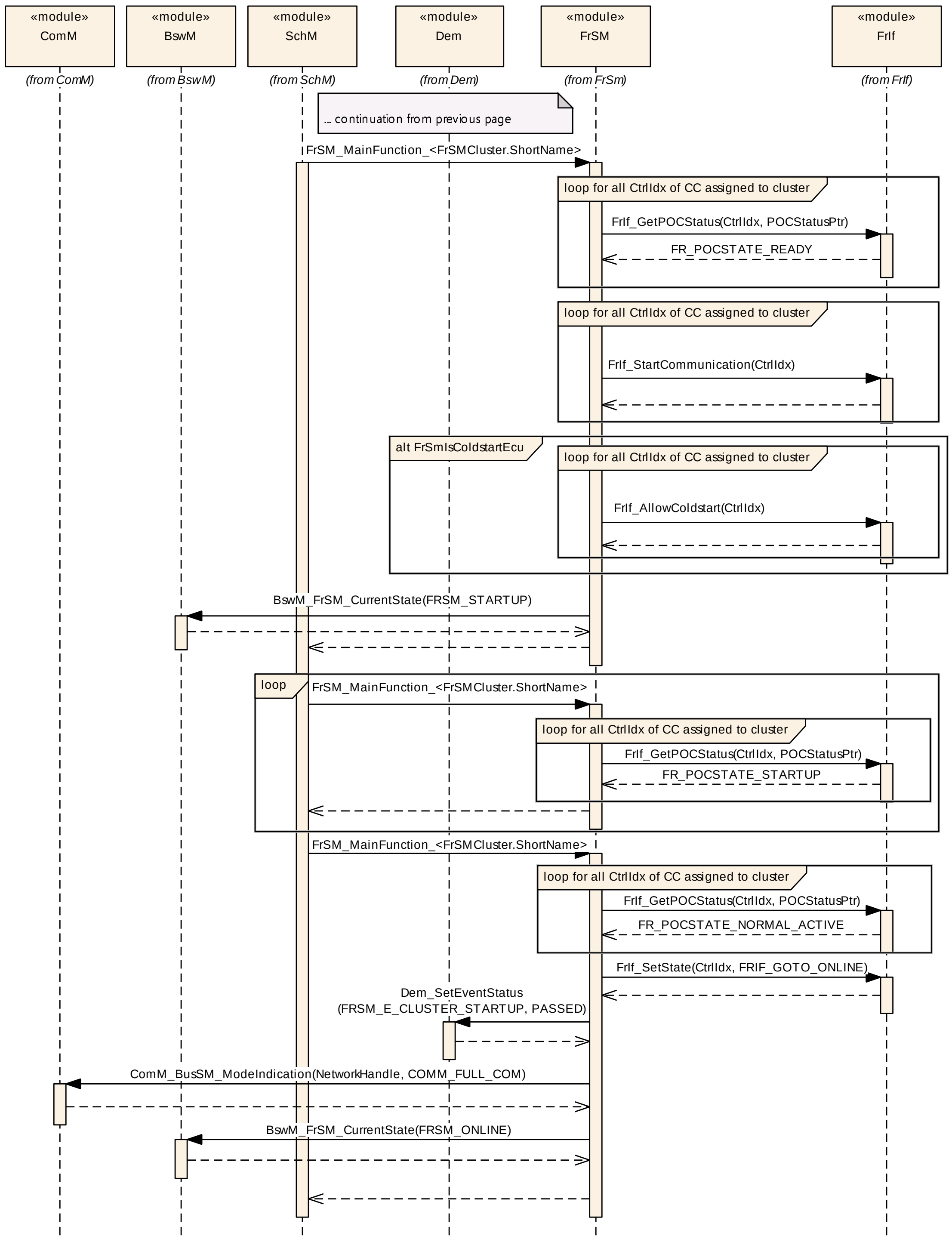

«module» ComM
«module» BswM
«module» SchM
«module» Dem
«module» FrSM
«module» FrIf
(from ComM)
(from BswM)
(from SchM)
(from Dem)
(from FrSm)
(from FrIf)
... continuation from previous page
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_READY
loop for all CtrlIdx of CC assigned to cluster
FrIf_StartCommunication(CtrlIdx)
alt FrSmIsColdstartEcu
loop for all CtrlIdx of CC assigned to cluster
FrIf_AllowColdstart(CtrlIdx)
BswM_FrSM_CurrentState(FRSM_STARTUP)
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_STARTUP
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_NORMAL_ACTIVE
FrIf_SetState(CtrlIdx, FRIF_GOTO_ONLINE)
Dem_SetEventStatus
(FRSM_E_CLUSTER_STARTUP, PASSED)
ComM_BusSM_ModeIndication(NetworkHandle, COMM_FULL_COM)
BswM_FrSM_CurrentState(FRSM_ONLINE)

Figure 9.3: Transition from no communication to full communication for the case of an ECU that has a local wakeup reason.

9.3 Single Channel Passive Startup

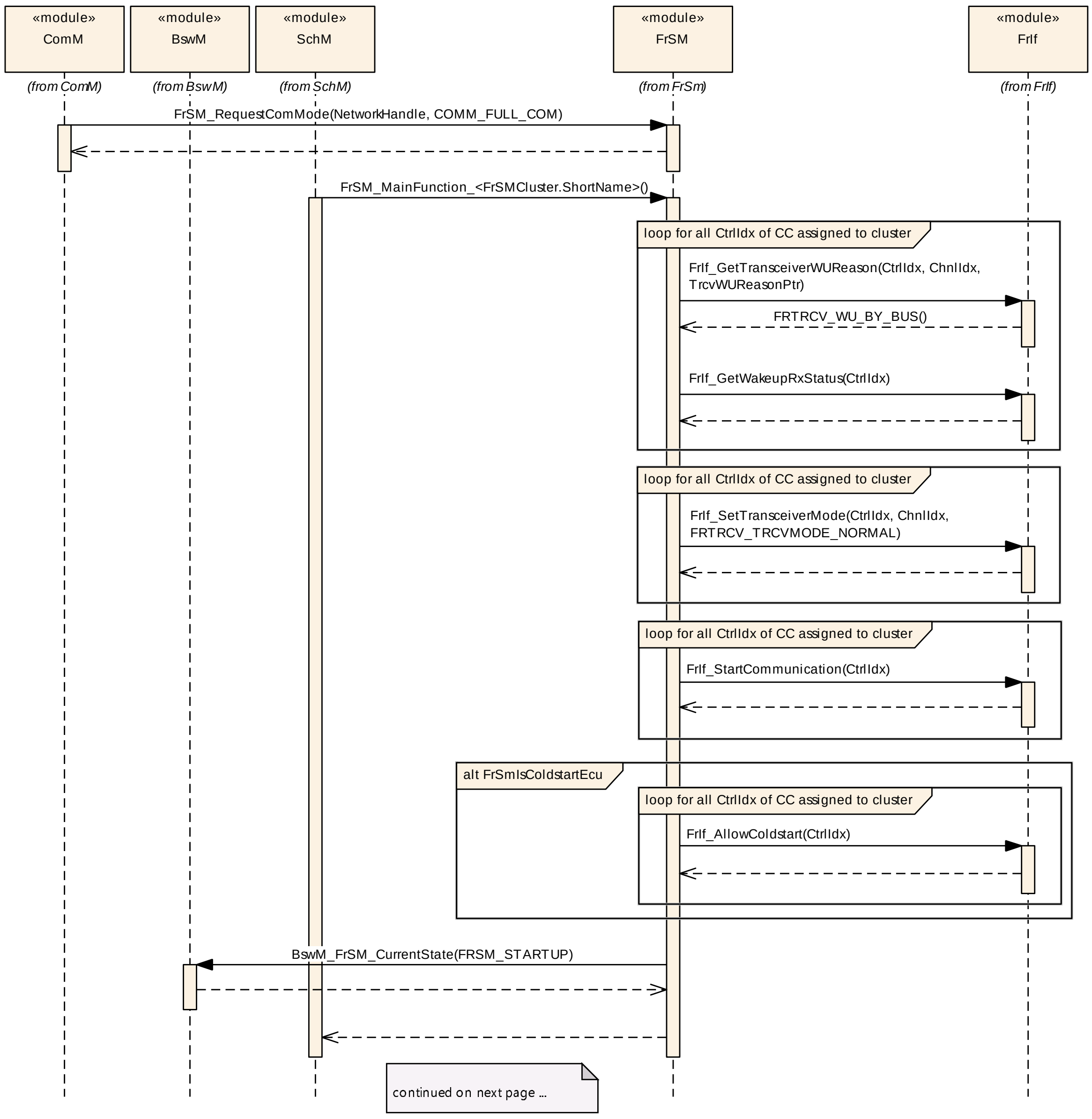

«module» ComM
«module» BswM
«module» SchM
«module» FrSM
«module» FrIf
(from ComM)
(from BswM)
(from SchM)
(from FrSm)
(from FrIf)
FrSM_RequestComMode(NetworkHandle, COMM_FULL_COM)
FrSM_MainFunction_<FrSMCluster.ShortName>()
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetTransceiverWUReason(CtrlIdx, ChnlIdx, TrcvWUReasonPtr)
FRTRCV_WU_BY_BUS()
FrIf_GetWakeupRxStatus(CtrlIdx)
loop for all CtrlIdx of CC assigned to cluster
FrIf_SetTransceiverMode(CtrlIdx, ChnlIdx, FRTRCV_TRCVMODE_NORMAL)
loop for all CtrlIdx of CC assigned to cluster
FrIf_StartCommunication(CtrlIdx)
alt FrSmIsColdstartEcu
loop for all CtrlIdx of CC assigned to cluster
FrIf_AllowColdstart(CtrlIdx)
BswM_FrSM_CurrentState(FRSM_STARTUP)
continued on next page ...

Figure 9.4: continued on next page

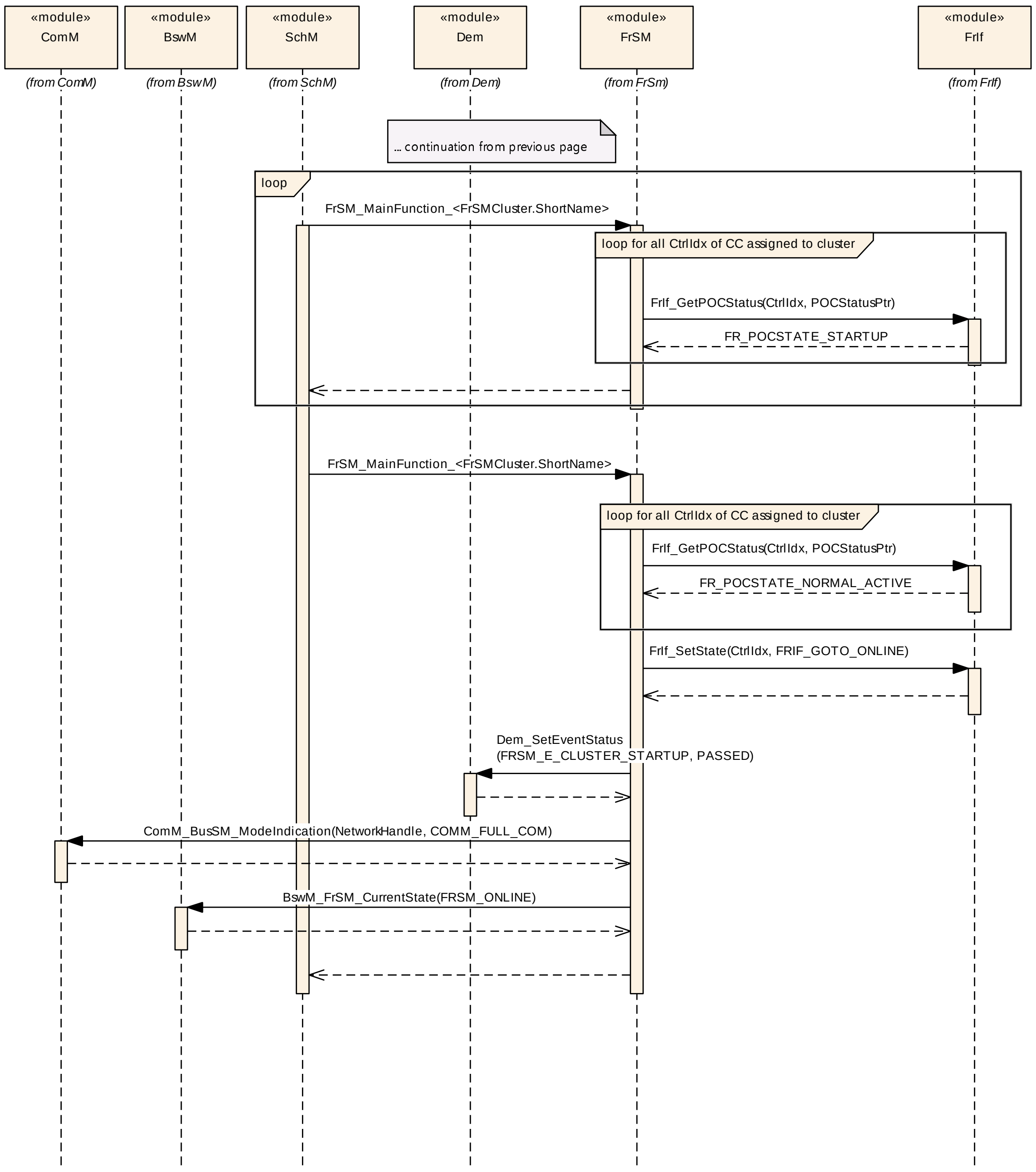

«module» ComM
(from ComM)
«module» BswM
(from BswM)
«module» SchM
(from SchM)
«module» Dem
(from Dem)
«module» FrSM
(from FrSm)
«module» FrIf
(from FrIf)
... continuation from previous page
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_STARTUP
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_NORMAL_ACTIVE
FrIf_SetState(CtrlIdx, FRIF_GOTO_ONLINE)
Dem_SetEventStatus (FRSM_E_CLUSTER_STARTUP, PASSED)
ComM_BusSM_ModeIndication(NetworkHandle, COMM_FULL_COM)
BswM_FrSM_CurrentState(FRSM_ONLINE)

Figure 9.5: Transition from no communication to full communication for the case of an ECU that has been woken up by bus.

9.4 Dual Channel Wakeup

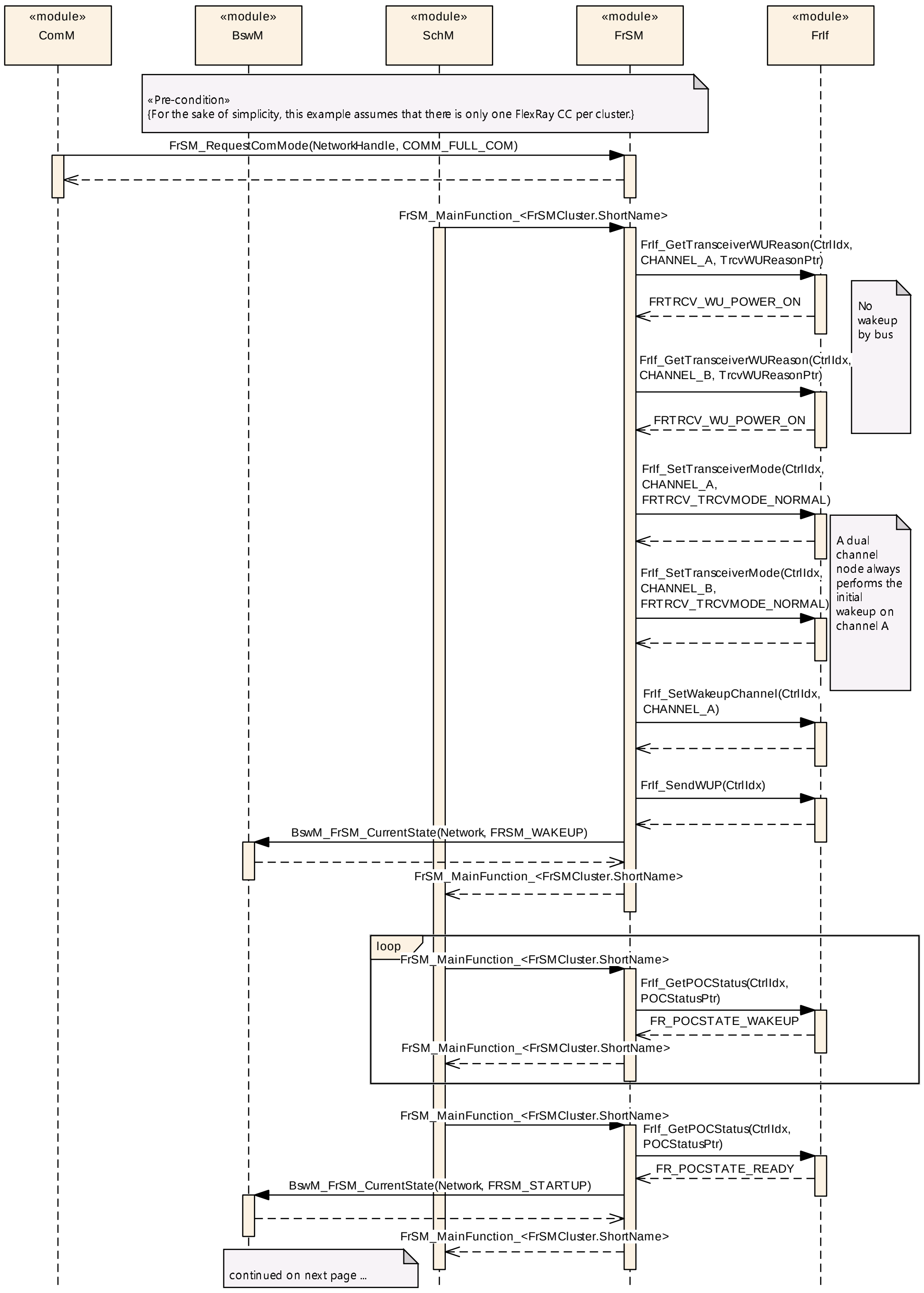

«module» ComM
«module» BswM
«module» SchM
«module» FrSM
«module» FrIf
«Pre-condition»
{For the sake of simplicity, this example assumes that there is only one FlexRay CC per cluster.}
FrSM_RequestComMode(NetworkHandle, COMM_FULL_COM)
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetTransceiverWUReason(CtrlIdx, CHANNEL_A, TrcvWUReasonPtr)
FRTRCV_WU_POWER_ON
No wakeup by bus
FrIf_GetTransceiverWUReason(CtrlIdx, CHANNEL_B, TrcvWUReasonPtr)
FRTRCV_WU_POWER_ON
FrIf_SetTransceiverMode(CtrlIdx, CHANNEL_A, FRTRCV_TRCVMODE_NORMAL)
A dual channel node always performs the initial wakeup on channel A
FrIf_SetTransceiverMode(CtrlIdx, CHANNEL_B, FRTRCV_TRCVMODE_NORMAL)
FrIf_SetWakeupChannel(CtrlIdx, CHANNEL_A)
FrIf_SendWUP(CtrlIdx)
BswM_FrSM_CurrentState(Network, FRSM_WAKEUP)
FrSM_MainFunction_<FrSMCluster.ShortName>
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_WAKEUP
FrSM_MainFunction_<FrSMCluster.ShortName>
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_READY
BswM_FrSM_CurrentState(Network, FRSM_STARTUP)
FrSM_MainFunction_<FrSMCluster.ShortName>
continued on next page ...

Figure 9.6: continued on next page

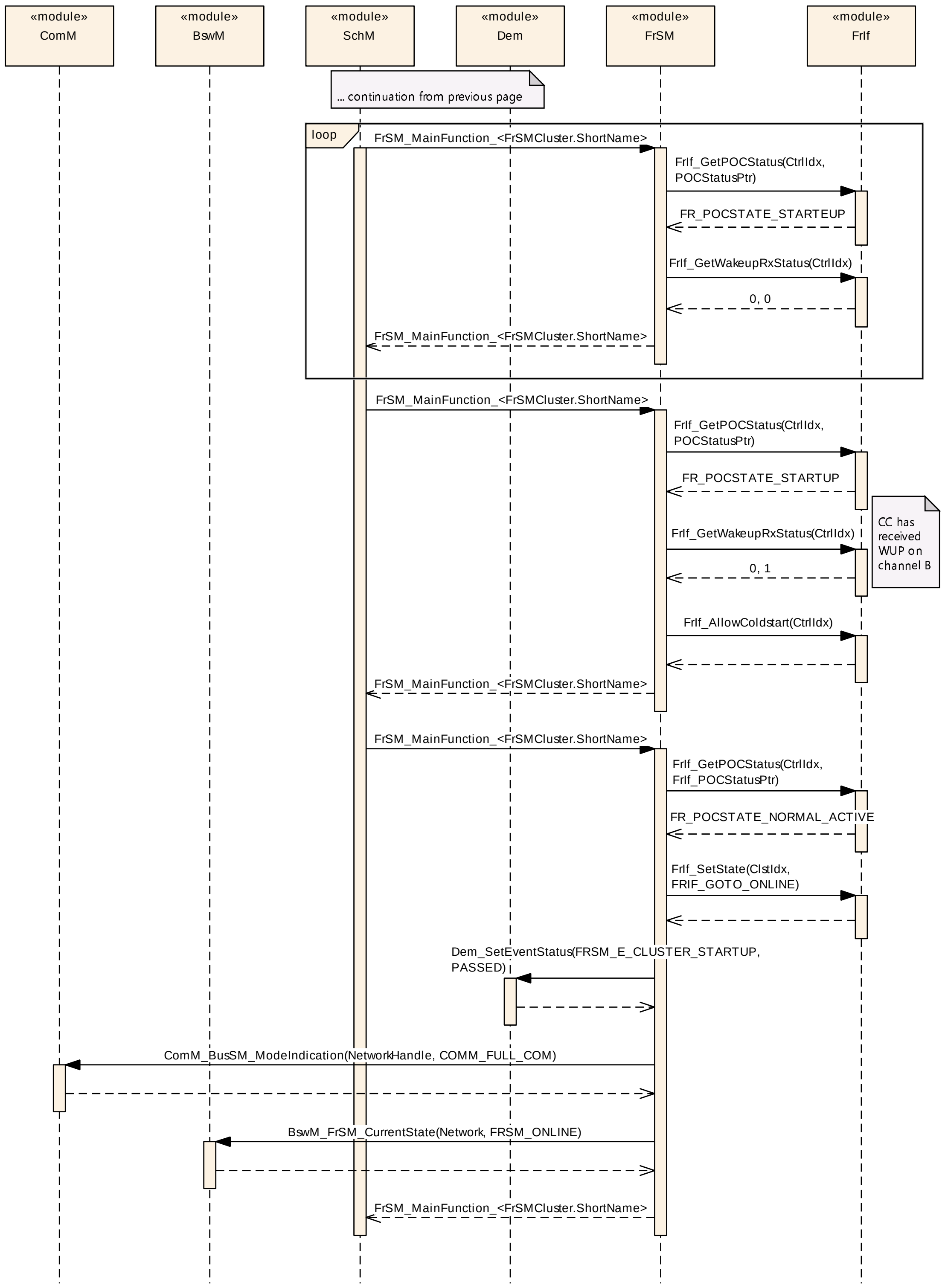

«module» ComM
«module» BswM
«module» SchM
«module» Dem
«module» FrSM
«module» FrIf
... continuation from previous page
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_STARTEUP
FrIf_GetWakeupRxStatus(CtrlIdx)
0, 0
FrSM_MainFunction_<FrSMCluster.ShortName>
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_STARTUP
FrIf_GetWakeupRxStatus(CtrlIdx)
0, 1
CC has received WUP on channel B
FrIf_AllowColdstart(CtrlIdx)
FrSM_MainFunction_<FrSMCluster.ShortName>
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, FrIf_POCStatusPtr)
FR_POCSTATE_NORMAL_ACTIVE
FrIf_SetState(ClstIdx, FRIF_GOTO_ONLINE)
Dem_SetEventStatus(FRSM_E_CLUSTER_STARTUP, PASSED)
ComM_BusSM_ModeIndication(NetworkHandle, COMM_FULL_COM)
BswM_FrSM_CurrentState(Network, FRSM_ONLINE)
FrSM_MainFunction_<FrSMCluster.ShortName>

Figure 9.7: Transition from no communication to full communication for the case of a dual channel ECU with a local wakeup reason.

9.5 Dual Channel Wakeup Forward

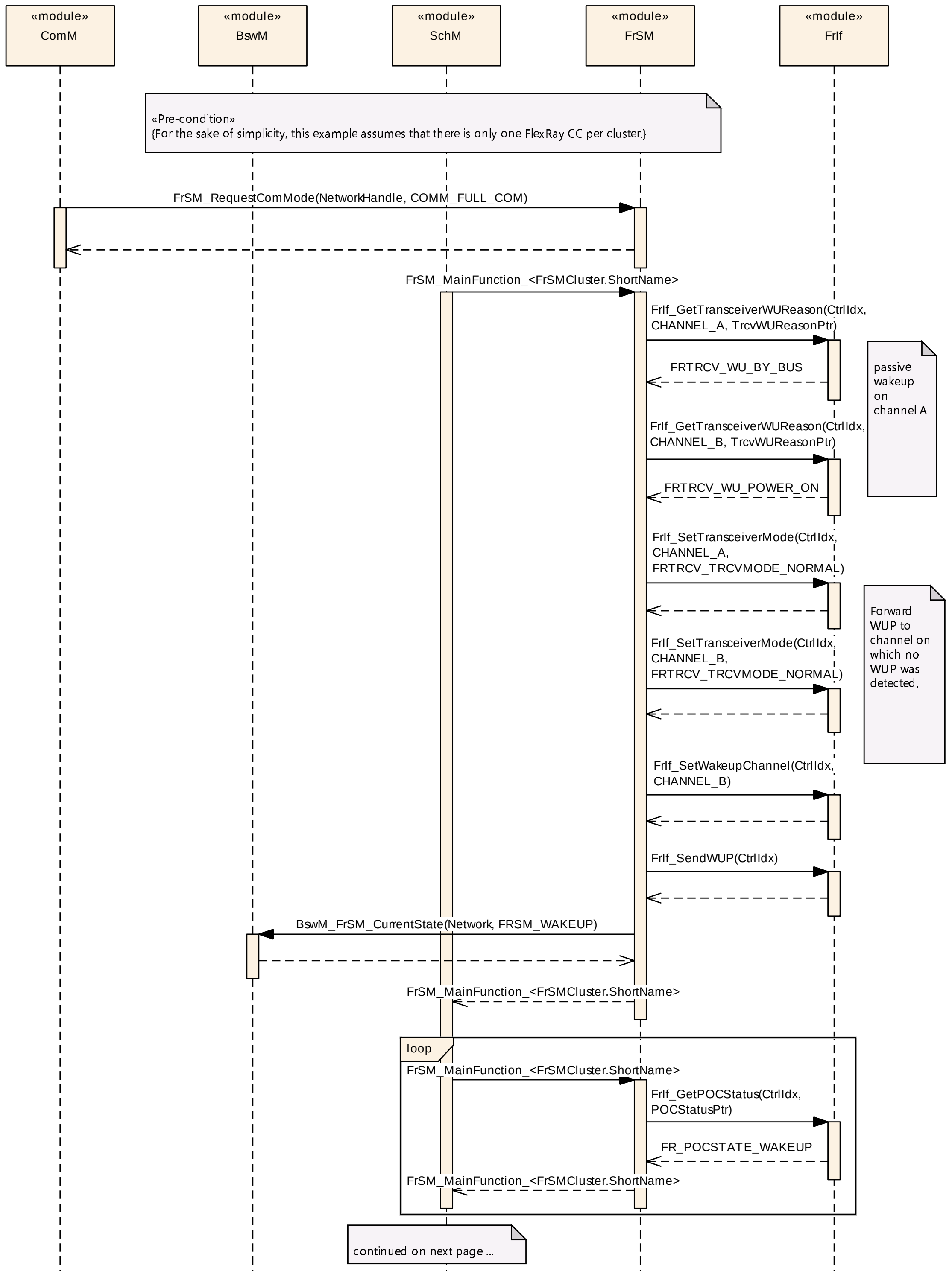

«module» ComM
«module» BswM
«module» SchM
«module» FrSM
«module» FrIf
«Pre-condition»
{For the sake of simplicity, this example assumes that there is only one FlexRay CC per cluster.}
FrSM_RequestComMode(NetworkHandle, COMM_FULL_COM)
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetTransceiverWUReason(CtrlIdx, CHANNEL_A, TrcvWUReasonPtr)
FRTRCV_WU_BY_BUS
passive wakeup on channel A
FrIf_GetTransceiverWUReason(CtrlIdx, CHANNEL_B, TrcvWUReasonPtr)
FRTRCV_WU_POWER_ON
FrIf_SetTransceiverMode(CtrlIdx, CHANNEL_A, FRTRCV_TRCVMODE_NORMAL)
Forward WUP to channel on which no WUP was detected.
FrIf_SetTransceiverMode(CtrlIdx, CHANNEL_B, FRTRCV_TRCVMODE_NORMAL)
FrIf_SetWakeupChannel(CtrlIdx, CHANNEL_B)
FrIf_SendWUP(CtrlIdx)
BswM_FrSM_CurrentState(Network, FRSM_WAKEUP)
FrSM_MainFunction_<FrSMCluster.ShortName>
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_WAKEUP
FrSM_MainFunction_<FrSMCluster.ShortName>
continued on next page ...

Figure 9.8: continued on next page

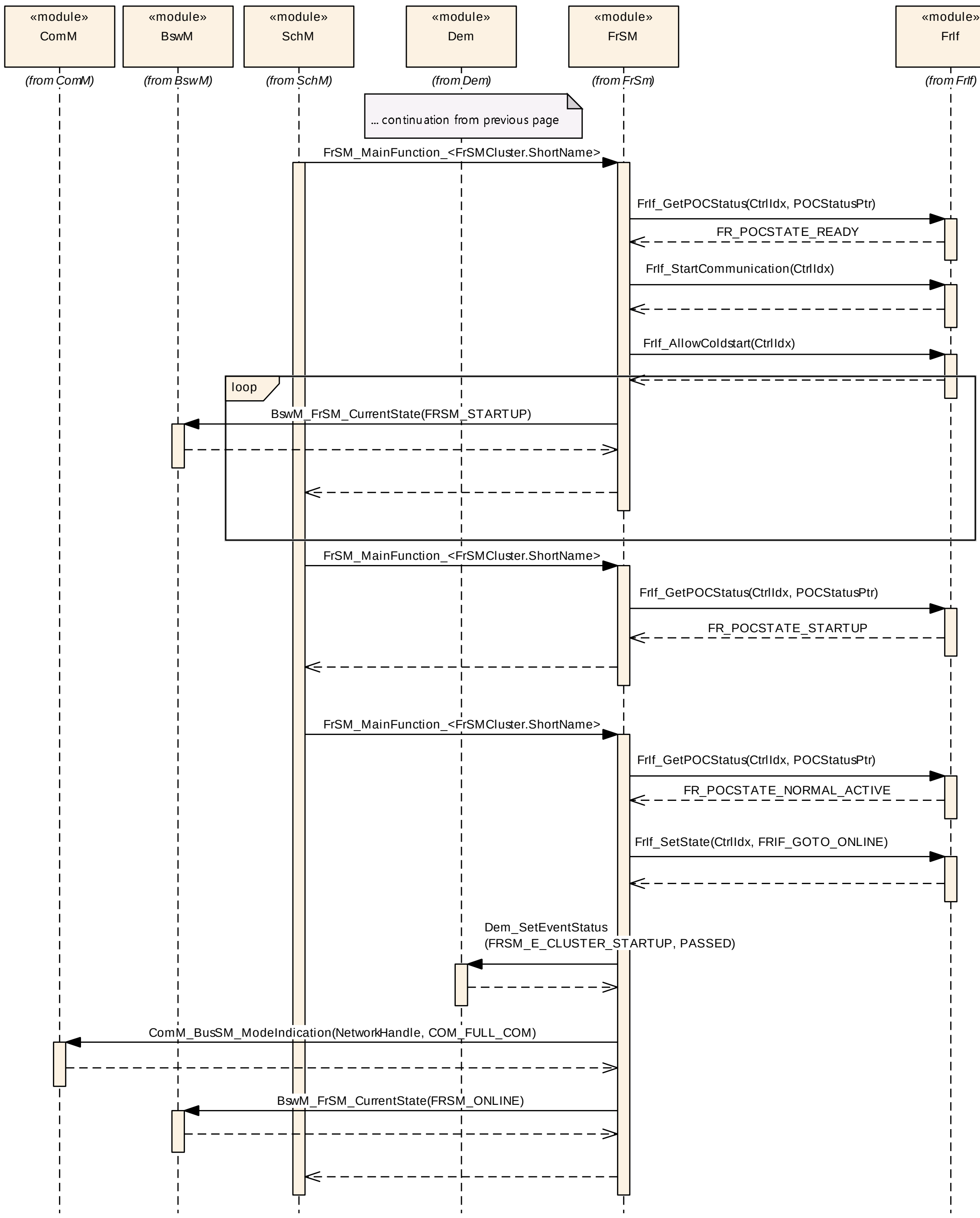

«module» ComM
«module» BswM
«module» SchM
«module» Dem
«module» FrSM
«module» FrIf
(from ComM)
(from BswM)
(from SchM)
(from Dem)
(from FrSm)
(from FrIf)
... continuation from previous page
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_READY
FrIf_StartCommunication(CtrlIdx)
FrIf_AllowColdstart(CtrlIdx)
loop
BswM_FrSM_CurrentState(FRSM_STARTUP)
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_STARTUP
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_NORMAL_ACTIVE
FrIf_SetState(CtrlIdx, FRIF_GOTO_ONLINE)
Dem_SetEventStatus (FRSM_E_CLUSTER_STARTUP, PASSED)
ComM_BusSM_ModeIndication(NetworkHandle, COM_FULL_COM)
BswM_FrSM_CurrentState(FRSM_ONLINE)

Figure 9.9: Transition from no communication to full communication for the case of a dual channel that has been woken up by bus.

9.6 Key Slot Only Mode

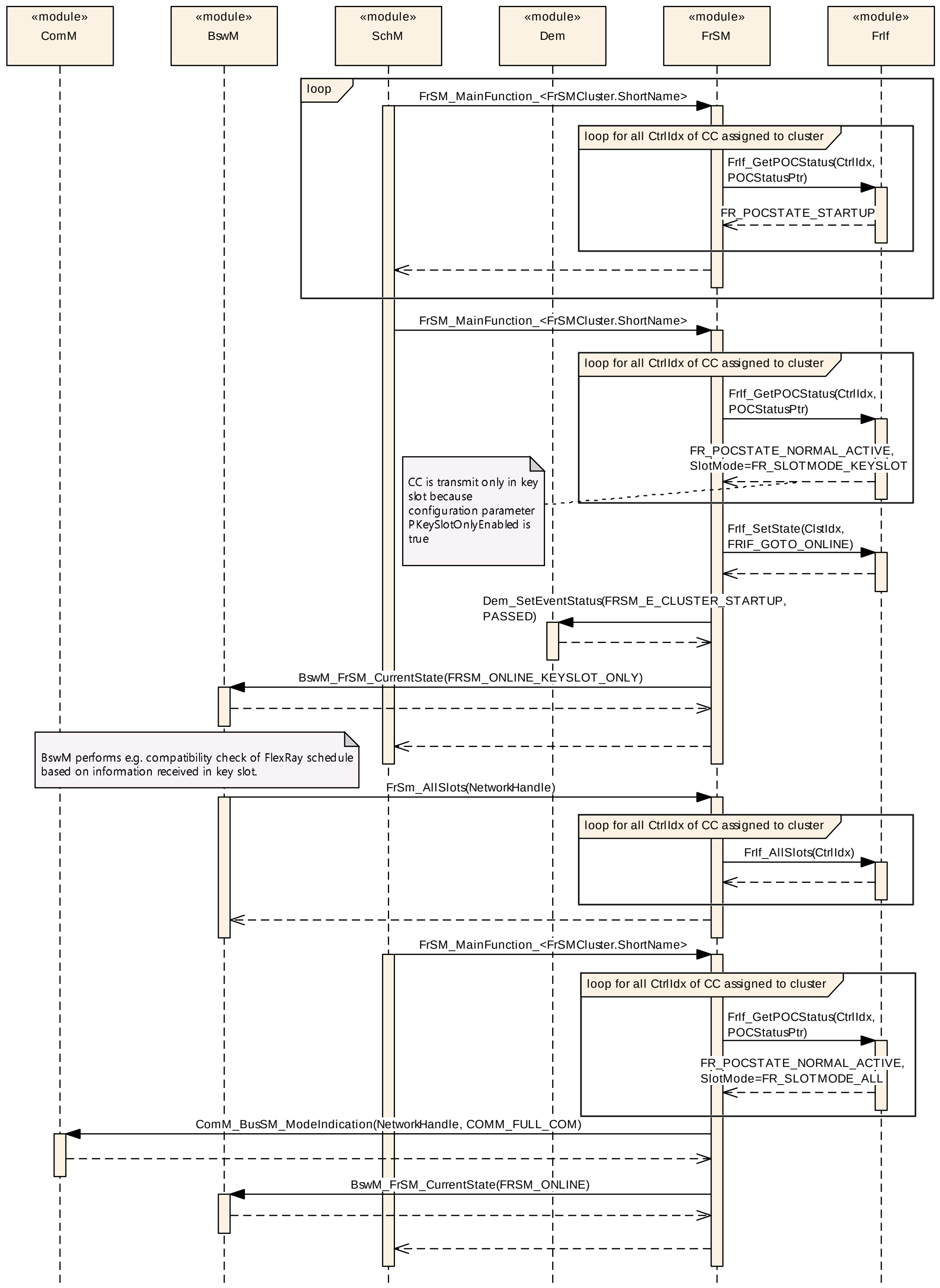

«module» ComM
«module» BswM
«module» SchM
«module» Dem
«module» FrSM
«module» FrIf
loop
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_STARTUP
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_NORMAL_ACTIVE, SlotMode=FR_SLOTMODE_KEYSLOT
CC is transmit only in key slot because configuration parameter PKeySlotOnlyEnabled is true
FrIf_SetState(ClstIdx, FRIF_GOTO_ONLINE)
Dem_SetEventStatus(FRSM_E_CLUSTER_STARTUP, PASSED)
BswM_FrSM_CurrentState(FRSM_ONLINE_KEYSLOT_ONLY)
BswM performs e.g. compatibility check of FlexRay schedule based on information received in key slot.
FrSm_AllSlots(NetworkHandle)
loop for all CtrlIdx of CC assigned to cluster
FrIf_AllSlots(CtrlIdx)
FrSM_MainFunction_<FrSMCluster.ShortName>
loop for all CtrlIdx of CC assigned to cluster
FrIf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_NORMAL_ACTIVE, SlotMode=FR_SLOTMODE_ALL
ComM_BusSM_ModeIndication(NetworkHandle, COMM_FULL_COM)
BswM_FrSM_CurrentState(FRSM_ONLINE)

Figure 9.10: Startup in case of Key Slot Only Mode is Enabled

9.7 Transition from full communication to no communication

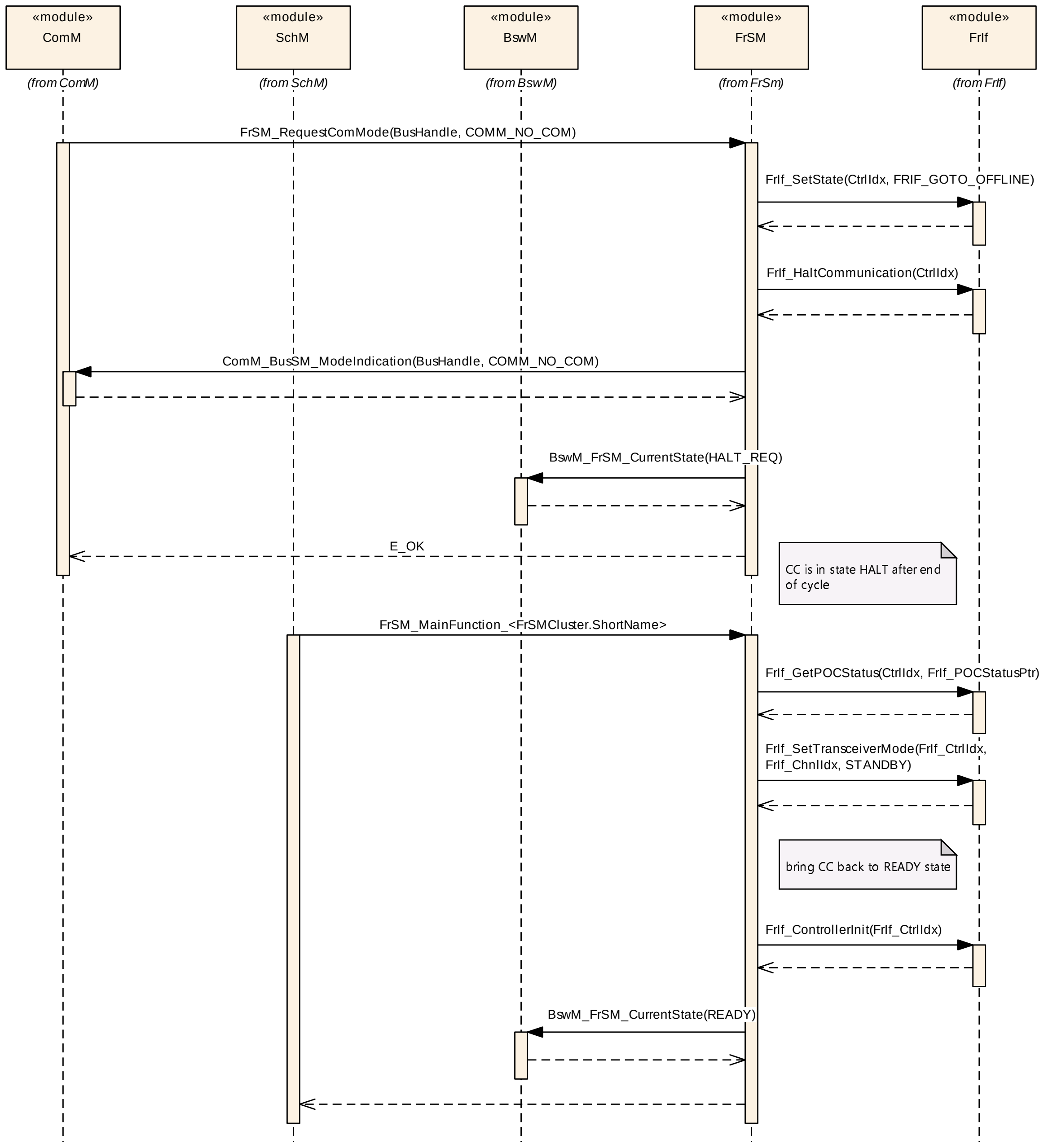

«module» ComM
(from ComM)
«module» SchM
(from SchM)
«module» BswM
(from BswM)
«module» FrSM
(from FrSm)
«module» FrIf
(from FrIf)
FrSM_RequestComMode(BusHandle, COMM_NO_COM)
FrIf_SetState(CtrlIdx, FRIF_GOTO_OFFLINE)
FrIf_HaltCommunication(CtrlIdx)
ComM_BusSM_ModeIndication(BusHandle, COMM_NO_COM)
BswM_FrSM_CurrentState(HALT_REQ)
E_OK
CC is in state HALT after end of cycle
FrSM_MainFunction_<FrSMCluster.ShortName>
FrIf_GetPOCStatus(CtrlIdx, FrIf_POCStatusPtr)
FrIf_SetTransceiverMode(FrIf_CtrlIdx, FrIf_ChnlIdx, STANDBY)
bring CC back to READY state
FrIf_ControllerInit(FrIf_CtrlIdx)
BswM_FrSM_CurrentState(READY)

Figure 9.11: Transition from FullCom to NoCom

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

For details refer to the chapter 10.1 “Introduction to configuration specification” in [2, SWS BSW General].

10.2 Containers and configuration parameters

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

[SWS_FrSM_00064] ⌈The `FrSM` module shall support tool based configuration.⌋ *(SRS_BSW_00159)*

[SWS_FrSM_00065] ⌈The configuration tool shall check the consistency of the configuration parameters at system configuration time.⌋ *(SRS_BSW_00167)*

10.2.1 FrSM

SWS Item	**[ECUC_FrSM_00174]**
Module Name	FrSM
Description	Configuration of the FlexRay State Manager
Post-Build Variant Support	true
Supported Config Variants	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
FrSMConfig	1	This container comprises the cluster specific configuration of the FlexRay State Manager.
FrSMGeneral	1	This container contains the general configuration parameters of the FlexRay State Manager.

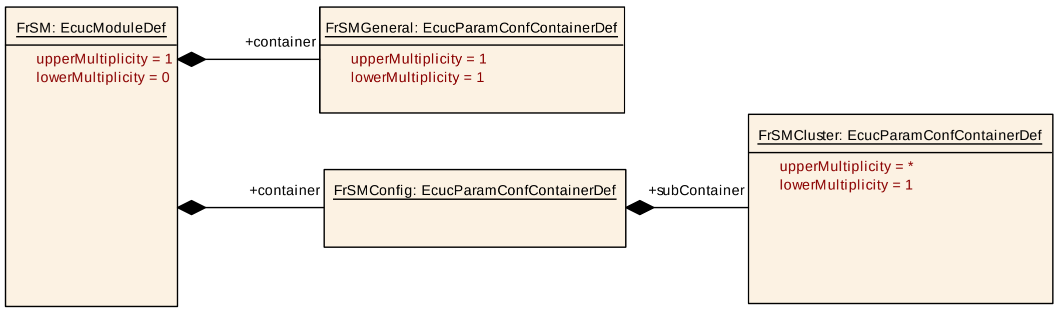


Figure 10.1: FlexRay State Manager Configuration

10.2.2 FrSMConfig

SWS Item	**[ECUC_FrSM_00146]**
Container Name	FrSMConfig
Parent Container	FrSM
Description	This container comprises the cluster specific configuration of the FlexRay State Manager.
Configuration Parameters	

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
FrSMCluster	1..*	This container specifies a FlexRay cluster and all related data. A FlexRay cluster may consist of more than one controller per ECU.

10.2.3 FrSMGeneral

SWS Item	**[ECUC_FrSM_00107]**
Container Name	FrSMGeneral
Parent Container	FrSM
Description	This container contains the general configuration parameters of the FlexRay State Manager.
Configuration Parameters	

SWS Item	**[ECUC_FrSM_00172]**
Parameter Name	FrSMAllSlotsSupport
Parent Container	FrSMGeneral
Description	Configuration parameter to enable/disable FrSM support to enable/disable the switching from key-slot/single-slot mode to all-slot mode.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	–
Post-Build Variant Multiplicity	false

▽

△

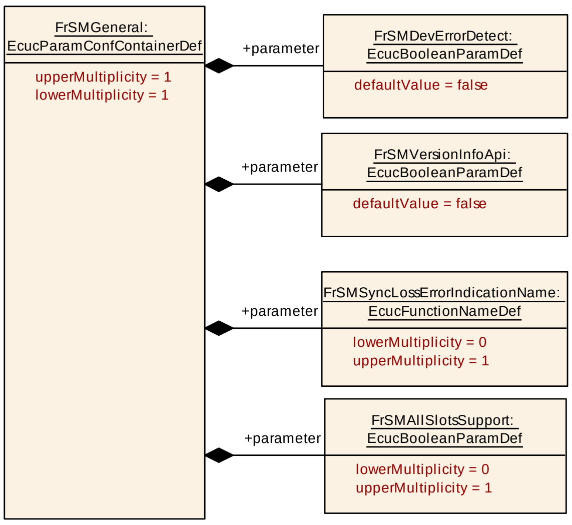
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00066]**		
Parameter Name	FrSMDevErrorDetect		
Parent Container	FrSMGeneral		
Description	Switches the development error detection and notification on or off. ● true: detection and notification is enabled. ● false: detection and notification is disabled.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00167]**		
Parameter Name	FrSMSyncLossErrorIndicationName		
Parent Container	FrSMGeneral		
Description	Name of <Cdd>_SyncLossErrorIndication function that shall be called on loss of synchronization. If this parameter is omitted no indication shall take place.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	–		
Regular Expression	–		
Post-Build Variant Multiplicity	false		
Post-Build Variant Value	false		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00108]**		
Parameter Name	FrSMVersionInfoApi		
Parent Container	FrSMGeneral		
Description	Enables and disables the version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	All Variants
	Link time	–	
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers


FrSMGeneral: EcucParamConfContainerDef
upperMultiplicity = 1
lowerMultiplicity = 1
+parameter
FrSMDevErrorDetect: EcucBooleanParamDef
defaultValue = false
+parameter
FrSMVersionInfoApi: EcucBooleanParamDef
defaultValue = false
+parameter
FrSMSyncLossErrorIndicationName: EcucFunctionNameDef
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrSMAllSlotsSupport: EcucBooleanParamDef
lowerMultiplicity = 0
upperMultiplicity = 1
Figure 10.2: FrSMGeneral Container

10.2.4 FrSMCluster

SWS Item	**[ECUC_FrSM_00067]**
Container Name	FrSMCluster
Parent Container	FrSMConfig
Description	This container specifies a FlexRay cluster and all related data. A FlexRay cluster may consist of more than one controller per ECU.
Configuration Parameters	

SWS Item	**[ECUC_FrSM_00001]**
Parameter Name	FrSMCheckWakeupReason
Parent Container	FrSMCluster

▽

△

Description	If FrSMCheckWakeupReason is true, the FrSM will check the wakeup reason in order to skip the wakeup in case of wakeup by bus. If FrSMCheckWakeupReason is false, the FrSM will always try to perform a wakeup.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00166]**		
Parameter Name	FrSMDelayStartupWithoutWakeup		
Parent Container	FrSMCluster		
Description	If true, timer t1 shall be started instead of immediately calling FrIf_AllowColdstart in case of a startup without wakeup.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00102]**		
Parameter Name	FrSMDurationT1		
Parent Container	FrSMCluster		
Description	The duration of timer t1 in seconds. A value of 0 shall imply that the timer is not used.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSMMainFunctionCycleTime (As timers are checked during the call of FrSM_MainFunction, the effective timer duration will always be a multiple of FrSMMainFunctionCycleTime).		

SWS Item	**[ECUC_FrSM_00089]**
Parameter Name	FrSMDurationT2
Parent Container	FrSMCluster

▽

△

Description	The duration of timer t2 in seconds. A value of 0 shall imply that the timer is not used. The value of this parameter shall be larger than the value of FrSMDurationT1 parameter.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSMMainFunctionCycleTime (As timers are checked during the call of FrSM_MainFunction, the effective timer duration will always be a multiple of FrSMMainFunctionCycleTime).		

SWS Item	**[ECUC_FrSM_00162]**		
Parameter Name	FrSMDurationT3		
Parent Container	FrSMCluster		
Description	The duration of timer t3 in seconds. The value of this parameter shall be larger than the value of FrSMDurationT1 parameter. A value of 0 shall imply that the timer is not used. It shall only be possible to configure a value 0 if no FrNm is used.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSMMainFunctionCycleTime (As timers are checked during the call of FrSM_MainFunction, the effective timer duration will always be a multiple of FrSMMainFunctionCycleTime).		

SWS Item	**[ECUC_FrSM_00173]**		
Parameter Name	FrSMDurationT4		
Parent Container	FrSMCluster		
Description	The timer t4 ensures that a dual channel node will eventually clear its coldstart inhibit bit and become a leading coldstarter.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_FrSM_00068]		
Parameter Name	FrSMIsColdstartEcu		
Parent Container	FrSMCluster		
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	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_FrSM_00109]		
Parameter Name	FrSMIsWakeupEcu		
Parent Container	FrSMCluster		
Description	True: FrSM shall perform a wakeup for this cluster. False: FrSM shall never perform a wakeup for this FlexRay cluster.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	[ECUC_FrSM_00115]		
Parameter Name	FrSMMainFunctionCycleTime		
Parent Container	FrSMCluster		
Description	This parameter defines the cycle time in seconds of the periodic calling of FrSM main function.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	]0 .. INF[		
Default value	–		
Post-Build Variant Value	false		
Value Configuration Class	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	[ECUC_FrSM_00168]
Parameter Name	FrSMMinNumberOfColdstarter
Parent Container	FrSMCluster

▽

△

Description	This parameter defines the number of coldstarter that should not be underrun. If this parameter is not configured the mainfunction shall not check the number of startup frames.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00165]**		
Parameter Name	FrSMNumWakeupPatterns		
Parent Container	FrSMCluster		
Description	Maximum number of Wakeup Patterns the node may send before going to FRSM_STARTUP.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: A value greater than zero is required in case the parameter FrSMIsWakeupEcu is true.		

SWS Item	**[ECUC_FrSM_00069]**		
Parameter Name	FrSMStartupRepetitions		
Parent Container	FrSMCluster		
Description	The number of times an ECU may repeat the startup procedure for a FlexRay cluster.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE

▽

△

	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: This value must be greater or equal to FrSMStartupRepetitionsWith Wakeup		

SWS Item	**[ECUC_FrSM_00094]**		
Parameter Name	FrSMStartupRepetitionsWithWakeup		
Parent Container	FrSMCluster		
Description	The number of times an ECU may repeat the startup procedure including a wakeup for a FlexRay cluster.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00170]**		
Parameter Name	FrSMTrcvStdbyDelay		
Parent Container	FrSMCluster		
Description	The duration of timer t_TrcvStdbyDelay in seconds. The granularity of this parameter shall be restricted to full FlexRay cycles (FrIfGdCycle). A value of 0 shall imply that the timer is not used.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	[0 .. INF]		
Default value	–		
Post-Build Variant Multiplicity	true		
Post-Build Variant Value	true		
Multiplicity Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSmMainFunctionCycleTime		

SWS Item	**[ECUC_FrSM_00070]**		
Parameter Name	FrSMComMNetworkHandleRef		
Parent Container	FrSMCluster		
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	Symbolic name reference to ComMChannel		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00116]**		
Parameter Name	FrSMFrIfClusterRef		
Parent Container	FrSMCluster		
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	Symbolic name reference to FrIfCluster		
Post-Build Variant Value	true		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

Included Containers		
Container Name	**Multiplicity**	**Scope / Dependency**
FrSMClusterDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.

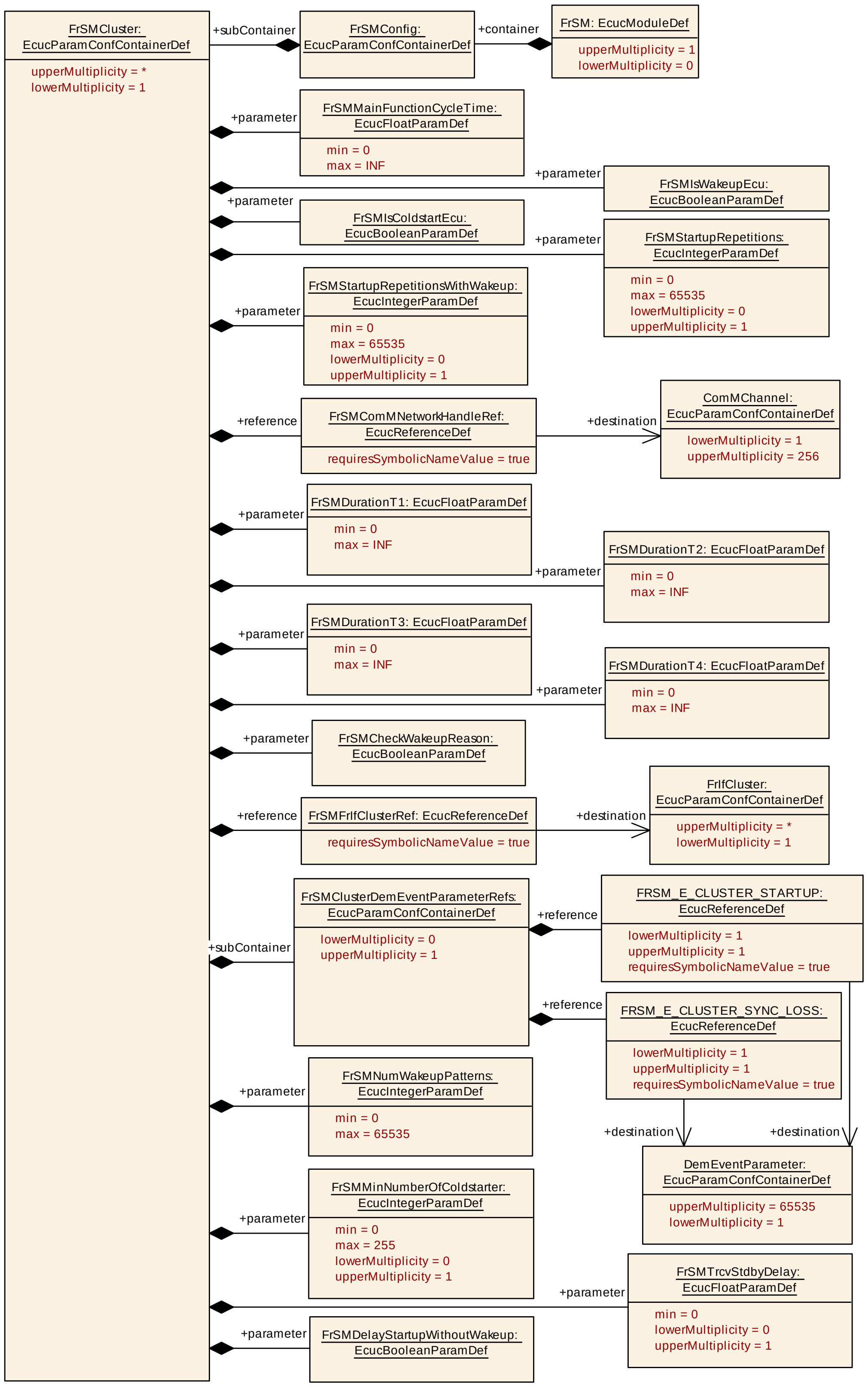

FrSMCluster: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+subContainer
FrSMConfig: EcucParamConfContainerDef
+container
FrSM: EcucModuleDef
upperMultiplicity = 1
lowerMultiplicity = 0
+parameter
FrSMMainFunctionCycleTime: EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMIsWakeupEcu: EcucBooleanParamDef
+parameter
FrSMIsColdstartEcu: EcucBooleanParamDef
+parameter
FrSMStartupRepetitions: EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrSMStartupRepetitionsWithWakeup: EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
FrSMComMNetworkHandleRef: EcucReferenceDef
requiresSymbolicNameValue = true
+destination
ComMChannel: EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = 256
+parameter
FrSMDurationT1: EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMDurationT2: EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMDurationT3: EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMDurationT4: EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMCheckWakeupReason: EcucBooleanParamDef
+reference
FrSMFrIfClusterRef: EcucReferenceDef
requiresSymbolicNameValue = true
+destination
FrIfCluster: EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+subContainer
FrSMClusterDemEventParameterRefs: EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
FRSM_E_CLUSTER_STARTUP: EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
requiresSymbolicNameValue = true
+reference
FRSM_E_CLUSTER_SYNC_LOSS: EcucReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
requiresSymbolicNameValue = true
+destination
+destination
DemEventParameter: EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 1
+parameter
FrSMNumWakeupPatterns: EcucIntegerParamDef
min = 0
max = 65535
+parameter
FrSMMinNumberOfColdstarter: EcucIntegerParamDef
min = 0
max = 255
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrSMTrcvStdbyDelay: EcucFloatParamDef
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrSMDelayStartupWithoutWakeup: EcucBooleanParamDef

Figure 10.3: FrSMCluster Container

10.2.5 FrSMClusterDemEventParameterRefs

SWS Item	**[ECUC_FrSM_00163]**
Container Name	FrSMClusterDemEventParameterRefs
Parent Container	FrSMCluster
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_SetEventStatus in case the corresponding error occurs. The Event Id is taken from the referenced DemEventParameter's DemEventId symbolic value. The standardized errors are provided in this container and can be extended by vendor-specific error references.
Configuration Parameters	

SWS Item	**[ECUC_FrSM_00164]**		
Parameter Name	FRSM_E_CLUSTER_STARTUP		
Parent Container	FrSMClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "FRSM_E_CLUSTER_STARTUP" has occurred. If the reference is not configured the error shall be reported as DET error.		
Multiplicity	1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

SWS Item	**[ECUC_FrSM_00169]**		
Parameter Name	FRSM_E_CLUSTER_SYNC_LOSS		
Parent Container	FrSMClusterDemEventParameterRefs		
Description	Reference to the DemEventParameter which shall be issued when the error "FRSM_E_CLUSTER_SYNC_LOSS" has occurred. If the reference is not configured the error shall be reported as DET error.		
Multiplicity	1		
Type	Symbolic name reference to DemEventParameter		
Post-Build Variant Value	false		
Value Configuration Class	**Pre-compile time**	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	–	
Scope / Dependency	scope: local		

No Included Containers

10.3 Published Information

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

A Not applicable requirements

[SWS_FrSM_NA_00186] ⌈This specification item references requirements that are not applicable, because it is no requirement against FrSM SWS or only against ECUC elements.⌋*(SRS_BSW_00170, SRS_BSW_00419, SRS_BSW_00375, SRS_BSW_00416, SRS_BSW_00437, SRS_BSW_00168, SRS_BSW_00423, SRS_BSW_00425, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00336, SRS_BSW_00422, SRS_BSW_00417, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00005, SRS_BSW_00415, SRS_BSW_00164, SRS_BSW_00325, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00314, SRS_BSW_00439, SRS_BSW_00449, SRS_BSW_00377, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, SRS_BSW_00172, SRS_BSW_00312, SRS_BSW_00330, SRS_BSW_00331, SRS_BSW_00343, SRS_BSW_00345, SRS_BSW_00351, SRS_BSW_00357, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00394, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00399, SRS_BSW_00401, SRS_BSW_00403, SRS_BSW_00448, SRS_BSW_00452, SRS_BSW_00453, SRS_BSW_00454, SRS_BSW_00456, SRS_BSW_00457, SRS_BSW_00462, SRS_BSW_00466, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00473, SRS_BSW_00478, SRS_BSW_00479, SRS_BSW_00486, SRS_BSW_00490, SRS_BSW_00491, SRS_BSW_00492, SRS_BSW_00493)*

[SWS_FrSM_NA_00001] ⌈This specification item references requirements that are not applicable, because CanNm does not have any service functionality.⌋*(SRS_BSW_00459, SRS_BSW_00494)*