

<b>Document Title</b>	Specification of FlexRay State Manager
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	254
<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R20-11

Document Change History			
Date	Release	Changed by	Change Description
2020-11-30	R20-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Improved Error Handling Sections</li> </ul>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Updated Transitions T03 and T06</li> <li>Changed Document Status from Draft to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation</li> </ul>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Added wakeup forwarding for dual channel FlexRay networks</li> <li>Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Revised development error handling.</li> <li>Debugging support marked as obsolete</li> <li>Minor corrections / clarifications / editorial changes; for details please refer to the ChangeDocumentation</li> </ul>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Changed development error checking of FrSM_Init pointer parameter.</li> <li>Editorial changes</li> </ul>

Document Change History			
Date	Release	Changed by	Change Description
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Removed Dual Channel Wakeup Echo</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>Added immediate handling of NoCom requests in normal passive mode or key slot only mode</li> <li>Editorial changes</li> <li>Removed chapter(s) on change documentation</li> </ul>
2013-03-15	4.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>FlexRay Transceiver Mode Switch can be delayed</li> <li>Formal updates</li> </ul>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Short term loss of synchronization is reported to DEM or DET.</li> <li>Number of startup frames can be monitored during normal operation.</li> <li>Revised production error handling.</li> </ul>
2010-09-30	3.1.5	AUTOSAR Administration	<ul style="list-style-type: none"> <li>The amount of wakeup patterns can be configured</li> <li>Clearing the Coldstart Inhibit Mode can be delayed also for passive wakeup.</li> <li>Removed enabling and disabling of transceiver wakeups</li> </ul>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Added support of FlexRay Dual Channel Wakeup</li> <li>Added support of FlexRay Single Slot Mode</li> <li>Added support of Passive Mode (Receive only)</li> <li>Improved timeout supervision of FlexRay startup</li> <li>Legal disclaimer revised</li> </ul>
2008-08-13	3.1.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Legal disclaimer revised</li> </ul>
2008-02-01	3.0.2	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Chapter 8 API Spelling harmonized</li> </ul>
2007-12-21	3.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>Initial Release</li> </ul>

## Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

## Table of Contents

1	Introduction and functional overview .....	6
2	Acronyms and abbreviations.....	7
3	Related documentation .....	8
3.1	Input documents .....	8
3.2	Related standards and norms .....	8
3.3	Related specification.....	8
4	Constraints and assumptions.....	10
4.1	Limitations .....	10
4.2	Applicability to car domains .....	10
5	Dependencies to other modules .....	11
5.1	AUTOSAR BSW Scheduler.....	11
5.2	Communication Manager.....	11
5.3	AUTOSAR FlexRay Interface .....	11
5.4	AUTOSAR Default Error Tracer .....	11
5.5	AUTOSAR Diagnostic Event Manager .....	11
5.6	AUTOSAR BSW Mode Manager.....	11
5.7	AUTOSAR FlexRay Network Management.....	11
5.8	File structure .....	11
5.8.1	Code file structure .....	11
5.8.2	Header file structure .....	12
6	Requirements traceability .....	13
7	Functional specification.....	16
7.1	Background & Rationale .....	16
7.2	Main Task of the FlexRay State Manager .....	16
7.3	State Machine of the FlexRay State Manager .....	16
7.3.1	General.....	16
7.3.2	States .....	17
7.3.3	Variables.....	18
7.3.4	State Machine Configuration.....	19
7.3.5	Conditions.....	20
7.3.6	Timers.....	21
7.3.7	Functional Elements.....	21
7.3.8	Wakeup Pattern Transmission .....	24
7.3.9	Transitions .....	24
7.4	Configuration description .....	30
7.5	Error classification .....	31
7.5.1	Development Errors .....	31
7.5.2	Runtime Errors .....	31
7.5.3	Transient Faults.....	31
7.5.4	Production Errors .....	31
7.5.5	Extended Production Errors.....	32
8	API specification.....	33

8.1	Imported types .....	33
8.2	Type definitions .....	33
8.2.1	FrSM_ConfigType .....	33
8.2.2	FrSM_BswM_StateType .....	34
8.3	Function definitions .....	35
8.3.1	FrSM_Init .....	35
8.3.2	FrSM_RequestComMode .....	35
8.3.3	FrSM_GetCurrentComMode .....	37
8.3.4	FrSM_GetVersionInfo .....	38
8.3.5	FrSM_AllSlots .....	39
8.3.6	FrSM_SetEcuPassive .....	40
8.4	Call-back notifications .....	41
8.5	Scheduled functions .....	41
8.5.1	FrSM_MainFunction_<FrSMCluster.ShortName> .....	41
8.6	Expected Interfaces .....	42
8.6.1	Mandatory Interfaces .....	42
8.6.2	Optional Interfaces .....	43
8.6.3	Configurable Interfaces .....	44
9	Sequence diagrams .....	46
9.1	Initialization .....	46
9.2	Single Channel Wakeup .....	47
9.3	Single Channel Passive Startup .....	50
9.4	Dual Channel Wakeup .....	52
9.5	Dual Channel Wakeup Forward .....	55
9.6	Key Slot Only Mode .....	57
9.7	Transition from full communication to no communication .....	58
10	Configuration specification .....	59
10.1	How to read this chapter .....	59
10.2	Containers and configuration parameters .....	59
10.2.1	FrSM .....	60
10.2.2	FrSMConfig .....	60
10.2.3	FrSMGeneral .....	61
10.2.4	FrSMCluster .....	64
10.2.5	FrSMClusterDemEventParameterRefs .....	71
10.3	Published Information .....	72
11	Not applicable requirements .....	73



## 2 Acronyms and abbreviations

<b>Acronym/ Abbreviation</b>	<b>Description:</b>
<b>API</b>	Application Program Interface
<b>AUTOSAR</b>	Automotive Open System Architecture
<b>BSW</b>	Basic Software
<b>CC</b>	Communication Controller
<b>CHI</b>	Controller Host Interface
<b>ComM</b>	AUTOSAR Communication Manager
<b>DCM</b>	Diagnostic Communication Manager
<b>Dem/DEM</b>	Diagnostic Event Manager
<b>Det/DET</b>	Default Error Tracer
<b>e.g.</b>	[lat.] exempli gratia = [eng.] for example
<b>ECU</b>	Electronic Control Unit
<b>EcuM</b>	ECU State Manager
<b>Fr</b>	FlexRay Driver
<b>FrIf</b>	FlexRay Interface (AUTOSAR BSW module)
<b>FrSM</b>	FlexRay State Manager
<b>FrTrcv</b>	FlexRay Transceiver Driver
<b>i.e.</b>	[lat.] id est = [eng.] that is
<b>Id/ID</b>	Identifier
<b>N/A</b>	Not applicable
<b>NM</b>	Network Management
<b>PDU</b>	Protocol Data Unit
<b>POC</b>	Protocol Operation Control
<b>POCState</b>	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).
<b>RTE</b>	Runtime Environment
<b>RX</b>	Reception
<b>SchM</b>	Schedule Manager
<b>SW</b>	Software
<b>TX</b>	Transmission
<b>UML</b>	Unified Modeling Language
<b>vPOC</b>	Data structure provided from the <a href="#">CC</a> to the host at the <a href="#">CHI</a> , which contains the actual <a href="#">POC</a> status of the <a href="#">CC</a> .
<b>vPOC!Freeze</b>	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.
<b>vPOC!SlotMode</b>	vPOC!SlotMode denotes the SlotMode field that is part of the vPOC data structure.
<b>WUP</b>	Wake-Up Pattern
<b>XML</b>	Extensible markup language

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

## 3 Related documentation

### 3.1 Input documents

- [1] List of Basic Software Modules  
AUTOSAR\_TR\_BSWModuleList.pdf
- [2] Layered Software Architecture  
AUTOSAR\_EXP\_LayeredSoftwareArchitecture.pdf
- [3] General Requirements on Basic Software Modules  
AUTOSAR\_SRS\_BSWGeneral.pdf
- [4] Specification of ECU Configuration  
UTOSAR\_TPS\_ECUConfiguration.pdf
- [5] Specification of Communication Stack Types  
AUTOSAR\_SWS\_CommunicationStackTypes.pdf
- [6] Requirements on FlexRay  
AUTOSAR\_SRS\_FlexRay.pdf
- [7] Specification of FlexRay Interface  
AUTOSAR\_SWS\_FlexRayInterface.pdf
- [8] Specification of FlexRay Driver  
AUTOSAR\_SWS\_FlexRayDriver.pdf
- [9] Specification of Communication Manager  
AUTOSAR\_SWS\_ComManager.pdf
- [10] Requirements on Mode Management  
AUTOSAR\_SRS\_ModeManagement.pdf
- [11] Basic Software Module Description Template,  
AUTOSAR\_TPS\_BSWModuleDescriptionTemplate.pdf
- [12] General Specification of Basic Software Modules  
AUTOSAR\_SWS\_BSWGeneral.pdf

### 3.2 Related standards and norms

- [13] FlexRay Communications System Protocol Specification Version 2.1 Rev A

### 3.3 Related specification



AUTOSAR provides a General Specification on Basic Software modules [12] (SWS BSW General), which is also valid for FlexRay State Manager.

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

## 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 [11]) is required. Furthermore, it enables the synchronized operation of several ECUs within a car.

The FlexRay State Manager can be used for all domain applications which use the FlexRay Protocol.

## 5 Dependencies to other modules

### 5.1 AUTOSAR BSW Scheduler

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

### 5.2 Communication Manager

The [ComM](#) requests network communication modes and is notified by the FrSM when a communication mode is reached.

### 5.3 AUTOSAR FlexRay Interface

The FrSM uses the API of the [Frlf](#) 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 FlexRay State Manager 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 FlexRay State Manager 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 FlexRay State Manager 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 FlexRay State Manager has to have access to the FlexRay 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 *SWS\_BSWGeneral*.

### 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 traceability

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_00005	Modules of the $\mu$ C Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_FrSM_00186
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_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_FrSM_00186
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_FrSM_00186
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_FrSM_00186
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_FrSM_00065
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_FrSM_00186
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_FrSM_00186
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_FrSM_00186
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_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_FrSM_00186
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_FrSM_00186
SRS_BSW_00347	A Naming separation of different instances of BSW drivers shall be in place	SWS_FrSM_00186
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_FrSM_00186

SRS_BSW_00360	AUTOSAR Basic Software Modules callback functions are allowed to have parameters	SWS_FrSM_00186
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_00375	Basic Software Modules shall report wake-up reasons	SWS_FrSM_00186
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_FrSM_00186
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_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_FrSM_00186
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_FrSM_00186
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_FrSM_00186
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_FrSM_00186
SRS_BSW_00419	If a pre-compile time configuration parameter is implemented as "const" it should be placed into a separate c-file	SWS_FrSM_00186
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_FrSM_00186
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_FrSM_00186
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_FrSM_00186
SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_FrSM_00186
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_FrSM_00186
SRS_BSW_00429	Access to OS is restricted	SWS_FrSM_00186
SRS_BSW_00432	Modules should have separate main	SWS_FrSM_00186

	processing functions for read/receive and write/transmit data path	
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_FrSM_00186
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_FrSM_00126, SWS_FrSM_00127, SWS_FrSM_00128
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_FrSM_00186
SRS_BSW_00440	The callback function invocation by the BSW module shall follow the signature provided by RTE to invoke servers via Rte_Call API	SWS_FrSM_00186
SRS_BSW_00449	BSW Service APIs used by Autosar Application Software shall return a Std_ReturnType	SWS_FrSM_00186
SRS_BSW_00450	A Main function of a un-initialized module shall return immediately	SWS_FrSM_00181
SRS_ModeMgm_09081	The Communication Manager shall provide an API allowing collecting communication requests	SWS_FrSM_00020
SRS_ModeMgm_09084	The Communication Manager shall provide an API which allows application to query the current communication mode	SWS_FrSM_00024

## 7 Functional specification

### 7.1 Background & Rationale

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

### 7.2 Main Task of the FlexRay State Manager

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

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

The FlexRay State Manager module shall not directly access the FlexRay hardware (FlexRay Communication Controller and FlexRay Transceiver), but by means of the FlexRay Interface module.

The FlexRay Interface 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]** [The FlexRay State Manager 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_<FrSMCluster.ShortName>` assigned to that cluster (see section 8.5.1). However, as defined in section 8.3.2, 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:

<i>FrSM Cluster State</i>	<i>Mapped FlexRay <a href="#">CC</a> state</i>	<i>Description</i>
FRSM_READY	<a href="#">POC</a> :ready	
FRSM_WAKEUP	<a href="#">POC</a> :wake-up	FrSM performs wake-up
FRSM_STARTUP	<a href="#">POC</a> :start-up	FrSM performs startup
FRSM_HALT_REQ	<a href="#">POC</a> :normal active or <a href="#">POC</a> :normal passive	FrSM performs a shutdown
FRSM_ONLINE	<a href="#">POC</a> :normal active	Full Communication
FRSM_ONLINE_PASSIVE	<a href="#">POC</a> :normal passive	Due to clock synchronization errors no data is transmitted or received.
FRSM_KEYSLOT_ONLY	<a href="#">POC</a> :normal active ^ <a href="#">vPOC!SlotMode</a> ≠ AllSlots	Data can only be transmitted in the key slots.
FRSM_LOW_NUMBER_OF_COLDSTARTERS	<a href="#">POC</a> :normal active	Full communication; FlexRay is synchronized based on sync frames only.

] ()

**[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:

<i>Passive State</i>	<i>Description</i>
FRSM_ECU_ACTIVE	When the FrSM is concurrently in state <a href="#">FRSM_READY</a> , 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 <a href="#">FRSM_READY</a> , the transceivers are in set into mode FRTRCV_TRCVMODE_STANDBY, otherwise into mode FRTRCV_TRCVMODE_RECEIVEONLY.

] ()

**[SWS\_FrSM\_00180]** [For reporting these two concurrent states to the BswM, a corresponding value of FrSM\_BswM\_StateType shall be determined as follows:

<i>FrSM Cluster State</i>	<i>Passive State</i>	<i>FrSM_BswM_StateType value</i>
<a href="#">FRSM_READY</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM_READY
<a href="#">FRSM_READY</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM_READY_ECU_PASSIVE
<a href="#">FRSM_WAKEUP</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM_WAKEUP
<a href="#">FRSM_WAKEUP</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM_WAKEUP_ECU_PASSIVE
<a href="#">FRSM_STARTUP</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM_STARTUP
<a href="#">FRSM_STARTUP</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM_STARTUP_ECU_PASSIVE
<a href="#">FRSM_ONLINE</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM_ONLINE
<a href="#">FRSM_ONLINE</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM_ONLINE_ECU_PASSIVE
<a href="#">FRSM_ONLINE_PASSIVE</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM_ONLINE_PASSIVE
<a href="#">FRSM_ONLINE_PASSIVE</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM_ONLINE_PASSIVE_ECU_PASSIVE

<a href="#">FRSM KEYSLOT ONLY</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM KEYSLOT ONLY
<a href="#">FRSM KEYSLOT ONLY</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM KEYSLOT ONLY ECU PASSIVE
<a href="#">FRSM HALT REQUEST</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM HALT REQUEST
<a href="#">FRSM HALT REQUEST</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM HALT REQUEST ECU PASSIVE
<a href="#">FRSM LOW NUMBER OF COLD-STARTERS</a>	<a href="#">FRSM ECU ACTIVE</a>	FRSM_LOW_NUMBER_OF_COLDSTARTERS
<a href="#">FRSM LOW NUMBER OF COLD-STARTERS</a>	<a href="#">FRSM ECU PASSIVE</a>	FRSM_LOW_NUMBER_OF_COLD-STARTERS ECU PASSIVE

] ()

### 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.

<b>FrSM Variable</b>	<b>Type</b>	<b>Description</b>
reqComMode	<a href="#">ComM_ModeType</a>	The communication mode that has been requested by the <a href="#">ComM</a> . 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 <a href="#">ComM_ModeType</a> these modes are ordered as follows: <a href="#">NoCom</a> < <a href="#">SilentCom</a> < <a href="#">FullCom</a>
startupCounter	Integer	The number of startup attempts that have been performed
wakeupType	Enum	The following values are supported: <ul style="list-style-type: none"> <li>SingleChannelWakeup</li> <li>DualChannelWakeup</li> <li>DualChannelWakeupForward</li> <li>NoWakeup</li> </ul>
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 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 chapter 10.2 for each FlexRay cluster:

<b>FrSM Configuration Parameter</b>	<b>Type</b>	<b>Description</b>
FrSMIsWakeupEcu	Boolean	See chapter 10.2
FrSMCheckWakeupReason	Boolean	See chapter 10.2
FrSMIsColdstartEcu	Boolean	See chapter 10.2
FrSMIsDualChannelNode	Boolean	This configuration 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	The number of times an ECU may repeat the startup procedure including a wakeup for a FlexRay cluster. If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value $\infty$
FrSMStartupRepetitions	Integer	Determines how often the ECU can repeat the startup procedure by reinitializing the FlexRay <a href="#">CC</a> , see chapter 10.2. This value must not be smaller than <a href="#">FrSMStartupRepetitionsWithWakeup</a> . If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value $\infty$
FrSMNumWakeupPatterns	Integer	Maximum number of Wakeup Patterns the node may send before going to <a href="#">FRSM_STARTUP</a> .

FrSMDelayStartupWithoutWakeup	Boolean	If true, timer <a href="#">t1</a> shall be started instead of immediately calling FrIf_AllowColdstart in case of a startup without wakeup.
FrSMMinNumberOfColdstarter	Integer	Minimum number of startup frames that have to be present, see chapter 10.2

### 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	<p>If <a href="#">FrSMCheckWakeupReason</a> is false, WUReason evaluates to NO_WU_BY_BUS.</p> <p>Otherwise if <a href="#">FrSMCheckWakeupReason</a> 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</p> <ul style="list-style-type: none"> <li>• NO_WU_BY_BUS in case no wakeup has been detected.</li> <li>• 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</li> <li>• 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.</li> </ul>
t1_IsActive	boolean	Evaluates to true if <a href="#">t1</a> has been started and has not expired yet, otherwise to false
t3_IsNotActive	boolean	Evaluates to false if <a href="#">t3</a> is running and has not expired, otherwise to true.
t_TrcvStdbby-Delay_IsActive	boolean	Evaluates to true if <a href="#">t_TrcvStdbbyDelay</a> has been started and has not expired yet, otherwise to false.
wakeupFinished	boolean	Evaluates to false if the wakeup pattern transmission as defined in section 7.3.8 is still in progress, otherwise to true.
lowNumberOfColdstarters	boolean	= FrIf_GetNumOfStartupFrames() < <a href="#">FrSMMinNumberOfColdstarter</a>

### 7.3.6 Timers

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

Timer	Description
t1	The timer <a href="#">t1</a> models the delay of clearing the coldstart inhibit mode (i.e. calling <code>FrIf_AllowColdstart</code> ). The duration of this timer can be statically configured with the configuration parameter <code>FrSMDurationT1</code> .
t2	The timer <a href="#">t2</a> 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 <code>FrSMDurationT2</code> .
t3	The timer t3 supervises the transition to <a href="#">FullCom</a> . The duration of this timer can be statically configured with the configuration parameter <code>FrSMDurationT3</code> .
t4	The timer t4 ensures that a dual channel node will eventually clear its coldstart inhibit bit and become a leading coldstarter.
t_TrvcStdbbyDelay	The timer t_TrvcStdbbyDelay models the time difference after which the FlexRay State Manager 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 `FrSMTrcvStdbbyDelay` is not configured or set to 0, timer t\_TrvcStdbbyDelay 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 <code>FrIf_SendWUP</code> 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 <code>FrIf_SetWakeupChannel</code> 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 <code>FrIf_SetWakeupChannel</code> 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 <a href="#">WUReason</a> .
FE_CONFIG	Call <code>FrIf_ControllerInit</code> for each controller of the FlexRay cluster.
FE_START	Call <code>FrIf_StartCommunication</code> for each controller of the FlexRay cluster.
FE_ALLOW_COLDSTART	Call <code>FrIf_AllowColdstart</code> for each controller of the FlexRay cluster if the configuration parameter <a href="#">FrSMIsColdstartEcu</a> is true.
FE_HALT	Call <code>FrIf_HaltCommunication</code> for each controller of the FlexRay cluster.
FE_TRCV_STANDBY	Call <code>FrIf_SetTransceiverMode</code> with <code>FrIf_TrcvMode</code> as <code>FRTRCV_TRCVMODE_STANDBY</code> for each transceiver of the FlexRay cluster.
FE_TRCV_NORMAL	In case the FrSM state machine is in state <a href="#">FRSM ECU ACTIVE</a> , call <code>FrIf_SetTransceiverMode</code> with <code>FrIf_TrcvMode</code> as <code>FRTRCV_TRCVMODE_NORMAL</code> and <code>FrIf_ClearTransceiverWakeup</code> for each transceiver of the FlexRay cluster. In case the FrSM state machine is in state <a href="#">FRSM ECU PASSIVE</a> , call <code>FrIf_SetTransceiverMode</code> with <code>FrIf_TrcvMode</code> as <code>FRTRCV_TRCVMODE_RECEIVEONLY</code> and <code>FrIf_ClearTransceiverWakeup</code> for each transceiver of the FlexRay cluster.
FE_START_FRIF	Set the <a href="#">FrIf</a> state to ONLINE by calling <code>FrIf_SetState</code> with <code>FrIf_StateTransition</code> as <code>FRIF_GOTO_ONLINE</code> for the cluster.
FE_STOP_FRIF	Set the <a href="#">FrIf</a> state to OFFLINE by calling <code>FrIf_SetState</code> with <code>FrIf_StateTransition</code> as <code>FRIF_GOTO_OFFLINE</code> for the cluster.
FE_DEM_STATUS_FAILED	Report status of production error <a href="#">FRSM E CLUSTER STARTUP</a> as failed.
FE_DEM_STATUS_PASSED	Report status of production error <a href="#">FRSM E CLUSTER STARTUP</a> as passed.
FE_DEM_SYNC_LOSS	Report the status of the production error <a href="#">FRSM E CLUSTER SYNC LOSS</a> as failed. If the name of an indication function (see section 8.6.3) is configured, call the indication function with the parameter <code>SyncLossErrorStatus = true</code> .

FE_DEM_SYNC_LOSS_PASSED	If the name of an indication function (see section 8.6.3) is configured, call the indication function with the parameter SyncLossErrorStatus = false. Additionally report the status of the production error <a href="#">FRSM_E_CLUSTER_SYNC_LOSS</a> as passed.
FE_FULL_COM_IND	Indicate to the <a href="#">ComM</a> that <a href="#">FullCom</a> has been reached by calling ComM_BusSM_ModeIndication ( <a href="#">FullCom</a> )
FE_NO_COM_IND	Indicate to the <a href="#">ComM</a> that <a href="#">FullCom</a> has been left by calling ComM_BusSM_ModeIndication ( <a href="#">NoCom</a> ).
FE_STARTUP_ERROR_IND	Call FrNm_StartupError.

### 7.3.8 Wakeup Pattern Transmission

**[SWS\_FrSM\_00208]** The FlexRay State Manager 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!WakeupStatus of the FlexRay controllers and act as follows:

vPOC!WakeupStatus	Actions
FR_WAKEUP_RECEIVED_HEADER, FR_WAKEUP_RECEIVED_WUP	<a href="#">busTrafficDetected</a> := true
FR_WAKEUP_TRANSMITTED	<a href="#">wakeupTransmitted</a> := true
FR_WAKEUP_UNDEFINED FR_WAKEUP_COLLISION_HEADER FR_WAKEUP_COLLISION_WUP FR_WAKEUP_COLLISION_UNKNOWN	<a href="#">wakeupTransmitted</a> := false

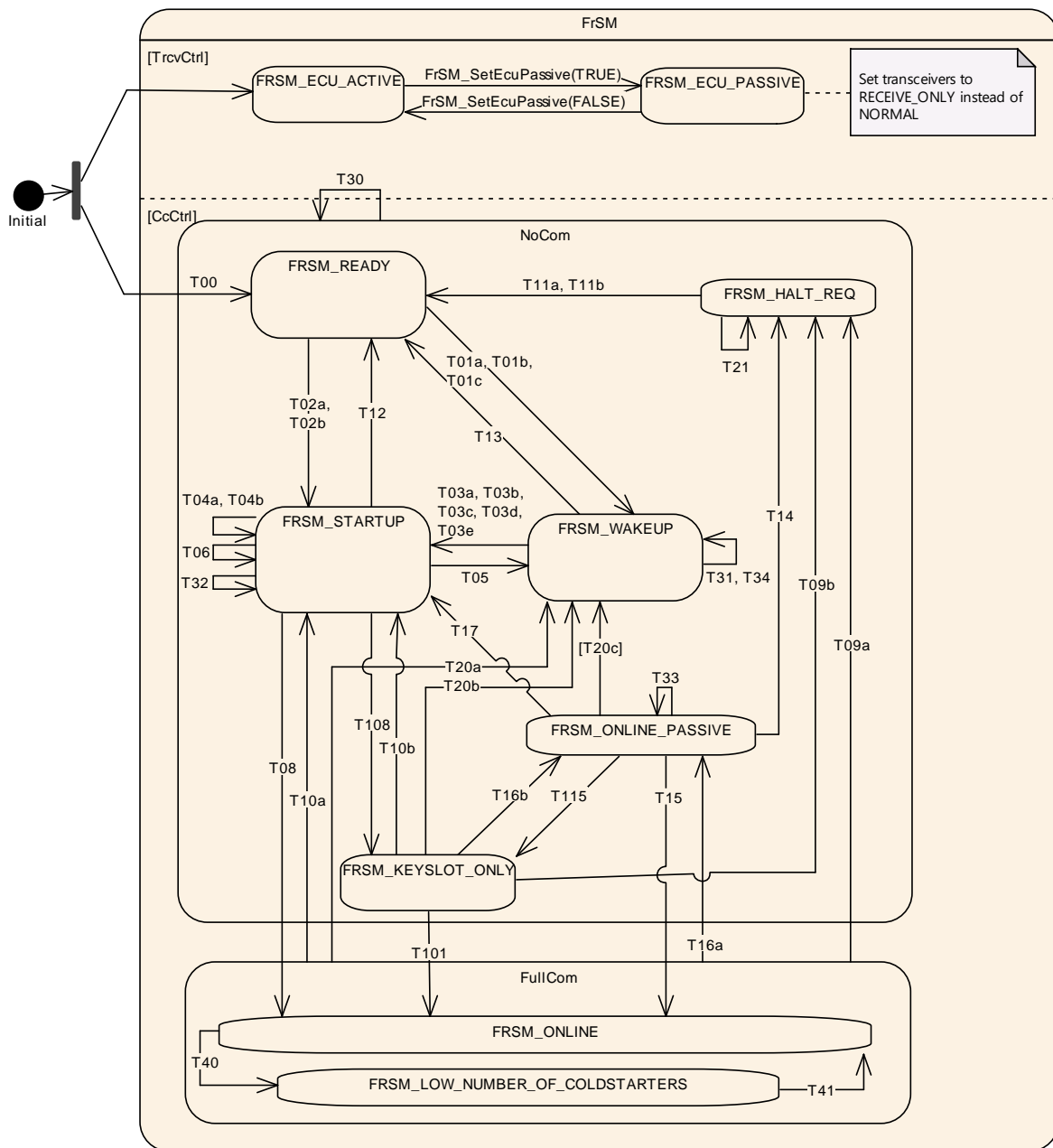
- If [busTrafficDetected](#) = false and wakeupCounter < [FrSMNumWakeupPatterns](#), execute [FE\\_WAKEUP](#)
- Increment the wakeupCounter

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

**[SWS\_FrSM\_00093]** The following FrSM state machine diagram defines source state and the target state of the transitions, which are defined in detail in the table following this diagram.





**Figure 2 FrSM state machine of the FlexRay State Manager**

Note that the states are described in section 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\_00145]** [After every transition to a different state, the FrSM shall inform the BswM by calling BswM\_FrSM\_CurrentState. ] ()

**[SWS\_FrSM\_00105]** [The FrSM shall execute the actions of the transition in the order that is defined in the following table.

Transition	Event [Condition]	Actions
T00	<a href="#">FrSM_Init()</a>	<a href="#">FE_CONFIG</a>
T01 (a)	[ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ <a href="#">FrSMIsWakeupEcu</a> $\wedge$ <a href="#">WUReason</a> = <a href="#">NO_WU_BY_BUS</a> $\wedge$ $\neg$ <a href="#">FrSMIsDualChannelNode</a> ]	<a href="#">FE_TRCV_NORMAL</a> <a href="#">startupCounter</a> := 1 <a href="#">wakeupType</a> := <a href="#">SingleChannelWakeup</a> <a href="#">wakeupTransmitted</a> := false <a href="#">FE_WAKEUP</a> start <a href="#">t1</a> start <a href="#">t3</a>
T01 (b)	[ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ <a href="#">FrSMIsWakeupEcu</a> $\wedge$ <a href="#">WUReason</a> = <a href="#">NO_WU_BY_BUS</a> $\wedge$ <a href="#">FrSMIsDualChannelNode</a> ]	<a href="#">FE_TRCV_NORMAL</a> <a href="#">startupCounter</a> := 1 <a href="#">wakeupType</a> := <a href="#">DualChannelWakeup</a> <a href="#">FE_SET_WU_CHANNEL_INITIAL</a> <a href="#">wakeupTransmitted</a> := false <a href="#">FE_WAKEUP</a> start <a href="#">t3</a> start <a href="#">t4</a>
T01 (c)	[ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ <a href="#">FrSMIsWakeupEcu</a> $\wedge$ <a href="#">WUReason</a> = <a href="#">PARTIAL_WU_BY_BUS</a> ]	<a href="#">FE_TRCV_NORMAL</a> <a href="#">startupCounter</a> := 1 <a href="#">wakeupType</a> := <a href="#">DualChannelWakeupForward</a> <a href="#">FE_SET_WU_CHANNEL_FORWARD</a> <a href="#">FE_WAKEUPwakeupTransmitted</a> := false <a href="#">FE_WAKEUP</a> start <a href="#">t3</a>
T02 (a)	[ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ ( $\neg$ <a href="#">FrSMIsWakeupEcu</a> $\vee$ <a href="#">WUReason</a> = <a href="#">ALL_WU_BY_BUS</a> ) $\wedge$ $\neg$ <a href="#">FrSMDelayStartupWithoutWakeup</a> ]	<a href="#">FE_TRCV_NORMAL</a> <a href="#">startupCounter</a> := 1 <a href="#">wakeupType</a> := <a href="#">NoWakeup</a> <a href="#">FE_START</a> <a href="#">FE_ALLOW_COLDSTART</a> start <a href="#">t2</a> start <a href="#">t3</a>
T02 (b)	[ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ ( $\neg$ <a href="#">FrSMIsWakeupEcu</a> $\vee$ <a href="#">WUReason</a> = <a href="#">ALL_WU_BY_BUS</a> ) $\wedge$ <a href="#">FrSMDelayStartupWithoutWakeup</a> ]	<a href="#">FE_TRCV_NORMAL</a> <a href="#">startupCounter</a> := 1 <a href="#">wakeupType</a> := <a href="#">NoWakeup</a> <a href="#">FE_START</a> start <a href="#">t1</a> start <a href="#">t2</a> start <a href="#">t3</a>
T03 (a)	[ <a href="#">wakeupFinished</a> $\wedge$ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ <a href="#">FrSMNumWakeupPatterns</a> = 1 $\wedge$ <a href="#">wakeupType</a> = <a href="#">SingleChannelWakeup</a> ]	<a href="#">FE_START</a> cancel <a href="#">t1</a> start <a href="#">t1</a> start <a href="#">t2</a>
T03 (b)	[ <a href="#">wakeupFinished</a> $\wedge$ <a href="#">reqComMode</a> = <a href="#">FullCom</a> $\wedge$ <a href="#">FrSMNumWakeupPatterns</a> > 1 $\wedge$ <a href="#">wakeupTransmitted</a> $\wedge$ <a href="#">wakeupType</a> = <a href="#">SingleChannelWakeup</a> ]	<a href="#">FE_START</a> start <a href="#">t2</a> IF <a href="#">t1_IsActive</a> : cancel <a href="#">t1</a> ELSE: <a href="#">FE_ALLOW_COLDSTART</a>

Transition	Event [Condition]	Actions
T03 (c)	[ <u>wakeupFinished</u> $\wedge$ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>FrSMNumWakeupPatterns</u> > 1 $\wedge$ $\neg$ <u>wakeupTransmitted</u> $\wedge$ <u>wakeupType</u> = <u>SingleChannelWakeup</u> ]	<u>FE_START</u> start <u>t2</u>
T03 (d)	[ <u>wakeupFinished</u> $\wedge$ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>wakeupType</u> = <u>DualChannelWakeup</u> $\wedge$ <u>wakeupTransmitted</u> $\wedge$ $\neg$ <u>busTrafficDetected</u> ]	<u>FE_START</u> start <u>t2</u>
T03 (e)	[ <u>wakeupFinished</u> $\wedge$ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>wakeupType</u> = <u>DualChannelWakeup-Forward</u> ]	<u>FE_START</u> <u>FE_ALLOW_COLDSTART</u> start <u>t2</u> cancel <u>t4</u>
T04 (a)	<u>t1</u> [ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>vPOC!State</u> $\neq$ Normal Active ]	<u>FE_ALLOW_COLDSTART</u>
T04 (b)	<u>t4</u> [ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>wakeupType</u> = <u>DualChannelWakeup</u> $\wedge$ <u>vPOC!State</u> $\neq$ Normal Active ]	<u>FE_ALLOW_COLDSTART</u>
T05	<u>t2</u> [ <u>startupCounter</u> $\leq$ <u>FrSMStartupRepetitionsWithWakeup</u> $\wedge$ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>wakeupType</u> $\neq$ <u>NoWakeup</u> $\wedge$ <u>vPOC!State</u> $\neq$ Normal Active] ]	<u>FE_CONFIG</u> <u>FE_WAKEUP</u> <u>startupCounter</u> := <u>startupCounter</u> + 1 start <u>t4</u> (dual channel node only)
T06	<u>t2</u> [ ( <u>FrSMStartupRepetitionsWithWakeup</u> < <u>startupCounter</u> $\vee$ <u>wakeupType</u> = <u>NoWakeup</u> ) $\wedge$ <u>startupCounter</u> $\leq$ <u>FrSMStartupRepetitions</u> $\wedge$ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ <u>vPOC!State</u> $\neq$ Normal Active] ]	<u>FE_CONFIG</u> <u>FE_START</u> <u>FE_ALLOW_COLDSTART</u> <u>startupCounter</u> := <u>startupCounter</u> + 1 start <u>t2</u>
T08	[ <u>vPOC!State</u> = Normal Active $\wedge$ $\neg$ <u>vPOC!Freeze</u> $\wedge$ <u>vPOC!SlotMode</u> = AllSlots $\wedge$ <u>reqComMode</u> = <u>FullCom</u> ]	cancel <u>t1</u> cancel <u>t2</u> <u>FE_START_FRIF</u> <u>FE_DEM_STATUS_PASSED</u> <u>FE_DEM_SYNC_LOSS_PASSED</u> <u>FE_FULL_COM_IND</u> cancel <u>t3</u>
T108	[ <u>vPOC!State</u> = Normal Active $\wedge$ $\neg$ <u>vPOC!Freeze</u> $\wedge$ <u>vPOC!SlotMode</u> $\neq$ AllSlots $\wedge$ <u>reqComMode</u> = <u>FullCom</u> ]	cancel <u>t1</u> cancel <u>t2</u> <u>FE_START_FRIF</u> <u>FE_DEM_STATUS_PASSED</u> <u>FE_DEM_SYNC_LOSS_PASSED</u> cancel <u>t3</u>
T09a	<u>FrSM_RequestComMode()</u> [ <u>reqComMode</u> = <u>NoCom</u> ]	<u>FE_STOP_FRIF</u> <u>FE_HALT</u> <u>FE_NO_COM_IND</u>
T09b	<u>FrSM_RequestComMode()</u> [ <u>reqComMode</u> = <u>NoCom</u> ]	<u>FE_STOP_FRIF</u> <u>FE_HALT</u>
T10a	[ ( <u>vPOC!State</u> = Halt $\vee$ <u>vPOC!Freeze</u> ) $\wedge$ <u>reqComMode</u> = <u>FullCom</u> $\wedge$ ( <u>FrSmCheckWakeupReason</u> $\vee$ $\neg$ <u>FrSMIsWakeupEcu</u> ) ]	<u>FE_DEM_SYNC_LOSS</u> <u>FE_STOP_FRIF</u> <u>FE_NO_COM_IND</u> <u>FE_CONFIG</u> <u>FE_START</u> <u>startupCounter</u> := 1 start <u>t2</u>

Transition	Event [Condition]	Actions
		start <a href="#">t3</a>
T10b	[ <a href="#">(vPOC!State = Halt ∨ vPOC!Freeze)</a> $\wedge$ <a href="#">reqComMode = FullCom</a> $\wedge$ ( <a href="#">FrSmCheckWakeupReason</a> $\vee \neg$ <a href="#">FrSMIsWakeupEcu</a> ) ]	<a href="#">FE_DEM_SYNC_LOSS</a> <a href="#">FE_STOP_FRIF</a> <a href="#">FE_CONFIG</a> <a href="#">FE_START</a> <a href="#">startupCounter := 1</a> start <a href="#">t2</a> start <a href="#">t3</a>
T101	[ <a href="#">vPOC!State = Normal Active</a> $\wedge \neg$ <a href="#">vPOC!Freeze</a> $\wedge$ <a href="#">vPOC!SlotMode = AllSlots</a> ]	<a href="#">FE_FULL_COM_IND</a>
T11a	<a href="#">t_TrcvStdbbyDelay[]</a>	<a href="#">FE_TRCV_STANDBY</a> <a href="#">FE_CONFIG</a>
T11b	[ ( <a href="#">vPOC!State = Halt</a> $\vee$ <a href="#">vPOC!Freeze</a> ) $\wedge$ <a href="#">reqComMode = FullCom</a> ]	cancel <a href="#">t_TrcvStdbbyDelay</a> <a href="#">FE_TRCV_STANDBY</a> <a href="#">FE_CONFIG</a>
T12	[ <a href="#">reqComMode = NoCom</a> ]	cancel <a href="#">t1</a> cancel <a href="#">t2</a> cancel <a href="#">t3</a> <a href="#">FE_DEM_SYNC_LOSS_PASSED</a> <a href="#">FE_TRCV_STANDBY</a> <a href="#">FE_CONFIG</a>
T13	[ <a href="#">reqComMode = NoCom</a> ]	<a href="#">FE_DEM_SYNC_LOSS_PASSED</a> <a href="#">FE_TRCV_STANDBY</a> <a href="#">FE_CONFIG</a> cancel <a href="#">t3</a> cancel <a href="#">t1</a>
T14	<a href="#">FrSM_RequestComMode()</a> [ <a href="#">reqComMode = NoCom</a> ]	<a href="#">FE_DEM_SYNC_LOSS_PASSED</a> <a href="#">FE_HALT</a> cancel <a href="#">t3</a>
T15	[ <a href="#">vPOC!State = Normal Active</a> $\wedge \neg$ <a href="#">vPOC!Freeze</a> $\wedge$ <a href="#">vPOC!SlotMode = AllSlots</a> ]	<a href="#">FE_DEM_SYNC_LOSS_PASSED</a> <a href="#">FE_START_FRIF</a> <a href="#">FE_FULL_COM_IND</a> cancel <a href="#">t3</a>
T115	[ <a href="#">vPOC!State = Normal Active</a> $\wedge \neg$ <a href="#">vPOC!Freeze</a> $\wedge$ <a href="#">vPOC!SlotMode</a> $\neq$ <a href="#">AllSlots</a> ]	<a href="#">FE_DEM_SYNC_LOSS_PASSED</a> <a href="#">FE_START_FRIF</a> cancel <a href="#">t3</a>
T16a	[ <a href="#">vPOC!State = Normal Passive</a> $\wedge \neg$ <a href="#">vPOC!Freeze</a> ]	<a href="#">FE_DEM_SYNC_LOSS</a> <a href="#">FE_STOP_FRIF</a> <a href="#">FE_NO_COM_IND</a> start <a href="#">t3</a>
T16b	[ <a href="#">vPOC!State = Normal Passive</a> $\wedge \neg$ <a href="#">vPOC!Freeze</a> ]	<a href="#">FE_DEM_SYNC_LOSS</a> <a href="#">FE_STOP_FRIF</a> start <a href="#">t3</a>
T17	[ <a href="#">(vPOC!State = Halt ∨ vPOC!Freeze)</a> $\wedge$ <a href="#">reqComMode = FullCom</a> $\wedge$ ( <a href="#">FrSmCheckWakeupReason</a> $\vee \neg$ <a href="#">FrSMIsWakeupEcu</a> ) ]	<a href="#">FE_CONFIG</a> <a href="#">wakeupType := NoWakeup</a> <a href="#">FE_START</a> <a href="#">startupCounter := 1</a> start <a href="#">t2</a>
T20a	[ <a href="#">(vPOC!State = Halt ∨ vPOC!Freeze)</a> $\wedge$ <a href="#">reqComMode = FullCom</a> $\wedge \neg$ <a href="#">FrSmCheckWakeupReason</a> $\wedge$ <a href="#">FrSMIsWakeupEcu</a> ]	<a href="#">wakeupType := SingleChannelWakeup</a> <a href="#">FE_DEM_SYNC_LOSSFE_STOP_FRIF</a> <a href="#">FE_NO_COM_IND</a> <a href="#">FE_CONFIG</a> <a href="#">FE_WAKEUP</a> <a href="#">startupCounter := 1</a> start <a href="#">t1</a>

Transition	Event [Condition]	Actions
		start <a href="#">t3</a>
T20b	[ <a href="#">(vPOC!State = Halt ∨ vPOC!Freeze)</a> $\wedge$ <a href="#">reqComMode = FullCom</a> $\wedge$ $\neg$ <a href="#">FrSmCheckWakeupReason</a> $\wedge$ <a href="#">FrSMIsWakeupEcu</a>	<a href="#">wakeupType := SingleChannelWakeup</a> <a href="#">FE_DEM_SYNC_LOSSFE_STOP_FRIF</a> <a href="#">FE_CONFIG</a> <a href="#">FE_WAKEUP</a> <a href="#">startupCounter := 1</a> start <a href="#">t1</a> start <a href="#">t3</a>
T20c	[ <a href="#">(vPOC!State = Halt ∨ vPOC!Freeze)</a> $\wedge$ <a href="#">reqComMode = FullCom</a> $\wedge$ $\neg$ <a href="#">FrSmCheckWakeupReason</a> $\wedge$ <a href="#">FrSMIsWakeupEcu</a>	<a href="#">wakeupType := SingleChannelWakeup</a> <a href="#">FE_CONFIG</a> <a href="#">FE_WAKEUP</a> <a href="#">startupCounter := 1</a> start <a href="#">t1</a> start <a href="#">t3</a>
T21	[ <a href="#">(vPOC!State = Halt</a> $\vee$ <a href="#">vPOC!Freeze)</a> $\wedge$ $\neg$ <a href="#">t_TrcvStdbbyDelay_IsActive</a>	start <a href="#">t_TrcvStdbbyDelay</a>
T30	<a href="#">t3[]</a>	<a href="#">FE_DEM_STATUS_FAILED</a> <a href="#">FE_STARTUP_ERROR_IND</a>
T31	<a href="#">t3_IsNotActive</a>	<a href="#">FE_STARTUP_ERROR_IND</a>
T32	<a href="#">t3_IsNotActive</a>	<a href="#">FE_STARTUP_ERROR_IND</a>
T33	<a href="#">t3_IsNotActive</a>	<a href="#">FE_STARTUP_ERROR_IND</a>
T34	[ <a href="#">wakeupFinished</a> $\wedge$ <a href="#">reqComMode = FullCom</a> $\wedge$ <a href="#">FrSMNumWakeupPatterns &gt; 1</a> $\wedge$ ( $\neg$ <a href="#">wakeupTransmitted</a> $\vee$ <a href="#">busTrafficDetected</a> ) $\wedge$ <a href="#">wakeupType = DualChannelWakeup</a> ]	<a href="#">startupCounter := 1</a> <a href="#">wakeupType := DualChannelWakeupForward</a> <a href="#">FE_SET_WU_CHANNEL_FORWARD</a> <a href="#">wakeupTransmitted := false</a> <a href="#">busTrafficDetected := false</a> <a href="#">FE_WAKEUP</a> start <a href="#">t1</a> start <a href="#">t3</a>
T40	[ <a href="#">lowNumberOfColdstarters</a> ]	
T41	[ $\neg$ <a href="#">lowNumberOfColdstarters</a> ]	

Legend:  $\wedge$  AND

$\vee$  OR

$\neg$  NOT

$:=$  assignment

start t: start timer *t*

cancel t: stop timer *t*

[...] guard condition for transition

*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 (see [FrSm076](#) and [FrSm077](#)), 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 (see [FrSm160](#) and [FrSm161](#)).

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 FlexRay Interface API returns a failure (e.g. E\_NOT\_OK), the FrSM shall ignore this return value and continue with the transition. ] ()

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

## 7.5 Error classification

Section 7.x "Error Handling" of the document "General Specification of Basic Software Modules" [12] 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]

<i>Type of error</i>	<i>Related error code</i>	<i>Error value</i>
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

<b>Error Name:</b>	FRSM_E_CLUSTER_STARTUP
--------------------	------------------------

<b>Short Description:</b>	FlexRay cluster startup failure.	
<b>Long Description:</b>	FlexRay controller has not reached the state <i>normal active</i> within the configured time after FlexRay startup.	
<b>Recommended DTC:</b>	Assigned by DEM	
<b>Detection Criteria:</b>	Fail	FlexRay controller has not reached the state normal active within the time <a href="#">t3</a>
	Pass	FlexRay controller has reached the state normal active
<b>Secondary Parameters:</b>	None	
<b>Time Required:</b>	FrSMDurationT3	
<b>Monitor Frequency</b>	Continuous	
<b>MIL illumination:</b>	Assigned by DEM	

#### 7.5.4.2 FRSM\_E\_CLUSTER\_SYNC\_LOSS

<b>Error Name:</b>	FRSM_E_CLUSTER_SYNC_LOSS	
<b>Short Description:</b>	FlexRay synchronization loss.	
<b>Long Description:</b>	FlexRay controller has lost synchronization after successful startup.	
<b>Recommended DTC:</b>	Assigned by DEM	
<b>Detection Criteria:</b>	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.
<b>Secondary Parameters:</b>	None	
<b>Time Required:</b>	Depends on FlexRay configuration.	
<b>Monitor Frequency</b>	Continuous	
<b>MIL illumination:</b>	Assigned by DEM	

#### 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 modules are listed:

[SWS\_FrSM\_00095]

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
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_POCSStateType
	Fr_GeneralTypes.h	Fr_POCSStatusType
	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_TrvcModeType
	Fr_GeneralTypes.h	FrTrcv_TrvcWUReasonType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

()

### 8.2 Type definitions

#### 8.2.1 FrSM\_ConfigType

[SWS\_FrSM\_00198]

<b>Name</b>	FrSM_ConfigType
<b>Kind</b>	Structure
<b>Description</b>	This type contains the implementation-specific post build time configuration structure

	that is for FrSM_Init.
<b>Available via</b>	FrSm.h

()

## 8.2.2 FrSM\_BswM\_StateType

[SWS\_FrSM\_00199]

<b>Name</b>	FrSM_BswM_StateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	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	--
<b>Description</b>	This type defines the states that are reported to the BswM using BswM_FrSM_CurrentState.		
<b>Available via</b>	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]

<b>Service Name</b>	FrSm_Init	
<b>Syntax</b>	<pre>void FrSm_Init (     const FrSM_ConfigType* FrSM_ConfigPtr )</pre>	
<b>Service ID [hex]</b>	0x01	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	FrSM_ConfigPtr	Pointer to a selected configuration structure
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	Initializes the FlexRay State Manager.	
<b>Available via</b>	FrSm.h	

](SRS\_BSW\_00405, SRS\_BSW\_00381, 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)

[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]

<b>Service Name</b>	FrSM_RequestComMode	
<b>Syntax</b>	<pre>Std_ReturnType FrSM_RequestComMode (     NetworkHandleType NetworkHandle,     ComM_ModeType ComM_Mode )</pre>	
<b>Service ID [hex]</b>	0x02	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant for different FlexRay clusters	
<b>Parameters (in)</b>	Network Handle	This parameter identifies the FlexRay cluster for which a communication mode is requested.
	ComM_Mode	This parameter holds the requested communication mode.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_Return-Type	E_OK: Request accepted E_NOT_OK: Request not accepted
<b>Description</b>	This API function is used by the ComM to startup or shutdown the communication on a FlexRay cluster.	
<b>Available via</b>	FrSm.h	

](SRS\_ModeMgm\_09081)

**[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\\_COLD-STARTERS](#) or [FRSM\\_ONLINE\\_PASSIVE](#)), the [FrSM\\_RequestComMode](#) function shall immediately process the corresponding transition of the state machine (see section 7.2). ] ()

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](#). ] ()

**[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)

**[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. ] ()

**[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)

### 8.3.3 FrSM\_GetCurrentComMode

**[SWS\_FrSM\_00024]**

<b>Service Name</b>	FrSM_GetCurrentComMode	
<b>Syntax</b>	<pre>Std_ReturnType FrSM_GetCurrentComMode (     NetworkHandleType NetworkHandle,     ComM_ModeType* ComM_ModePtr )</pre>	
<b>Service ID [hex]</b>	0x03	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different FlexRay clusters	
<b>Parameters (in)</b>	Network Handle	Handle of communication network
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	ComM_Mode Ptr	Pointer to the memory location where the current communication mode shall be stored
<b>Return value</b>	Std_Return-Type	E_OK: Request accepted E_NOT_OK: Request was not accepted as the FrSM has not been initialized using FrSM_Init.

<b>Description</b>	This API function can be used to determine the current communication mode of a FlexRay cluster.
<b>Available via</b>	FrSm.h

](SRS\_ModeMgm\_09084) **[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.

] ()

**[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. ] ()

**[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)

**[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\\_GetCurrentComMode](#) function shall return E\_NOT\_OK. ] (SRS\_BSW\_00406)

### 8.3.4 FrSM\_GetVersionInfo

**[SWS\_FrSM\_00029]**

<b>Service Name</b>	FrSM_GetVersionInfo
<b>Syntax</b>	<pre>void FrSM_GetVersionInfo (     Std_VersionInfoType* versioninfo )</pre>
<b>Service ID [hex]</b>	0x04
<b>Sync/Async</b>	Synchronous

<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	versioninfo	Pointer to where to store the version information of this module.
<b>Return value</b>	None	
<b>Description</b>	<p>This service returns the version information of this module. The version information includes:</p> <ul style="list-style-type: none"> <li>• Module Id</li> <li>• Vendor Id</li> <li>• Vendor specific version numbers (BSW00407).</li> </ul> <p>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.</p>	
<b>Available via</b>	FrSm.h	

](SRS\_BSW\_00407)

### 8.3.5 FrSM\_AllSlots

[SWS\_FrSM\_00172]

<b>Service Name</b>	FrSm_AllSlots	
<b>Syntax</b>	<pre>Std_ReturnType FrSm_AllSlots (     NetworkHandleType NetworkHandle )</pre>	
<b>Service ID [hex]</b>	0x05	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant for different FlexRay clusters	
<b>Parameters (in)</b>	Network Handle	This parameter identifies the FlexRay cluster for which a communication mode is requested.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	

<b>Return value</b>	Std_Return-Type	E_OK: Request accepted E_NOT_OK: Request not accepted
<b>Description</b>	This API function can be used to leave the KeySlotOnlyMode.	
<b>Available via</b>	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)

[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)

### 8.3.6 FrSM\_SetEcuPassive

[SWS\_FrSM\_00174]

<b>Service Name</b>	FrSm_SetEcuPassive	
<b>Syntax</b>	Std_ReturnType FrSm_SetEcuPassive ( boolean FrSM_Passive )	
<b>Service ID [hex]</b>	0x06	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	FrSM_Passive	This parameter determines whether all FlexRay clusters are set to passive, i.e. receive only.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_Return-Type	E_OK: Request accepted E_NOT_OK: Request not accepted
<b>Description</b>	This API function can be used to set all FlexRay clusters of the ECU to a receive	



	only mode.
<b>Available via</b>	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)

## 8.4 Call-back notifications

The FlexRay State Manager 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]

<b>Service Name</b>	FrSM_MainFunction_<FrSMCluster.ShortName>
<b>Syntax</b>	void FrSM_MainFunction_<FrSMCluster.ShortName> ( void )
<b>Service ID [hex]</b>	0x80
<b>Description</b>	--
<b>Available via</b>	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 FlexRay cycle have no benefit.

Note: After [FullCom](#) has been reached, the invocation of the [FrSM\\_MainFunction](#) can optionally be synchronized to the FlexRay global time to ensure that the [FrSM\\_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 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

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

**[SWS\_FrSM\_00096]**

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.
Frlf_AllowColdstart	Frlf.h	Wraps the FlexRay Driver API function Fr_AllowColdstart().
Frlf_Clear- TransceiverWakeup	Frlf.h	Wraps the FlexRay Transceiver Driver API function FrTrcv_ClearTransceiverWakeup(). The enum value "FR_CHANNEL_AB" shall not be used.
Frlf_ControllerInit	Frlf.h	Initialized a FlexRay CC.
Frlf_GetPOCStatus	Frlf.h	Wraps the FlexRay Driver API function Fr_GetPOCStatus().
Frlf_GetTransceiver- WUReason	Frlf.h	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverWUReason(). The enum value "FR_CHANNEL_AB" shall not be used.
Frlf_Halt- Communication	Frlf.h	Wraps the FlexRay Driver API function Fr_HaltCommunication().
Frlf_SendWUP	Frlf.h	Wraps the FlexRay Driver API function Fr_SendWUP().
Frlf_SetState	Frlf.h	Requests Frlf state machine transition.
Frlf_SetTransceiver- Mode	Frlf.h	Wraps the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode(). The enum value "FR_CHANNEL_AB" shall not be used.
Frlf_Start- Communication	Frlf.h	Wraps the FlexRay Driver API function Fr_StartCommunication().

]()

## 8.6.2 Optional Interfaces

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

### [SWS\_FrSM\_00097]

<b>API Function</b>	<b>Header File</b>	<b>Description</b>
Dem_Set- EventStatus	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/DemConfigSet/ DemEventParameter/DemEventReportingType) == STANDARD_ REPORTING)
Det_Report- Error	Det.h	Service to report development errors.
Frlf_AllSlots	Frlf.h	Wraps the FlexRay Driver API function Fr_AllSlots
Frlf_GetNum- OfStartup- Frames	Frlf.h	Wraps the FlexRay Driver API function Fr_GetNumOfStartupFrames and gets a list of the current number of startup frames seen on the cluster. See variable vStartupPairs of [12] for details.

FrIf_Get-WakeupRx-Status	FrIf.h	Wraps the FlexRay Driver API function Fr_GetWakeupRxStatus and gets the wakeup received information from the FlexRay controller.
FrIf_Set-Wakeup-Channel	FrIf.h	Wraps the FlexRay Driver API function Fr_SetWakeupChannel(). 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.

l()

### 8.6.3 Configurable Interfaces

#### 8.6.3.1 <Cdd>\_SyncLossErrorIndication

[SWS\_FrSM\_00190]

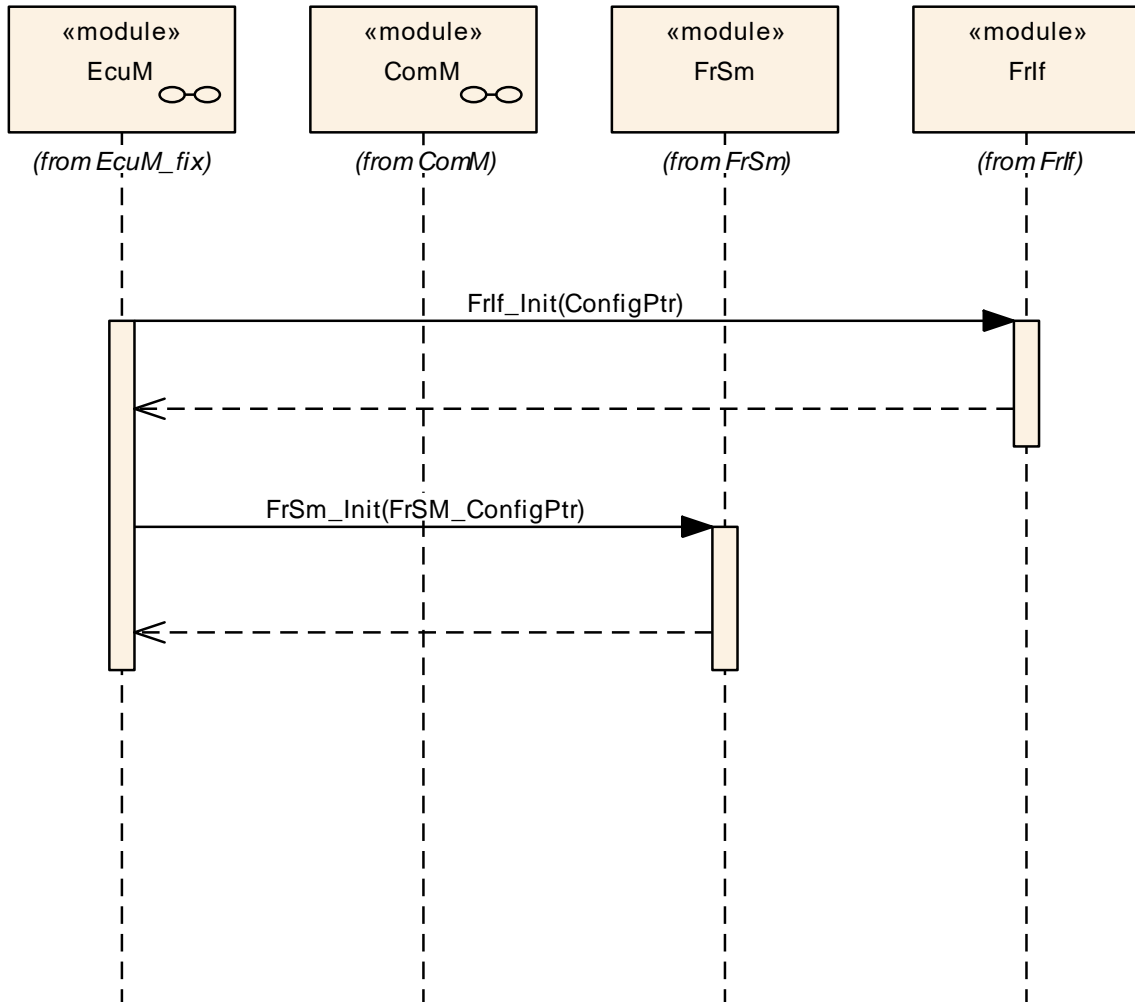
<b>Service Name</b>	<Cdd>_SyncLossErrorIndication	
<b>Syntax</b>	<pre>void &lt;Cdd&gt;_SyncLossErrorIndication (     NetworkHandleType NetworkHandle,     boolean SyncLossErrorStatus )</pre>	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different FlexRay clusters	
<b>Parameters (in)</b>	Network Handle	Handle of FlexRay cluster
	SyncLoss ErrorStatus	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 Flex Ray cluster.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	<p>This function is called with parameter SyncLossErrorStatus = true when the ECU loses its synchronization to the FlexRay cluster. The function is called with parameter SyncLossErrorStatus = 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.</p>	
<b>Available via</b>	FrSm_Externals.h	

l()

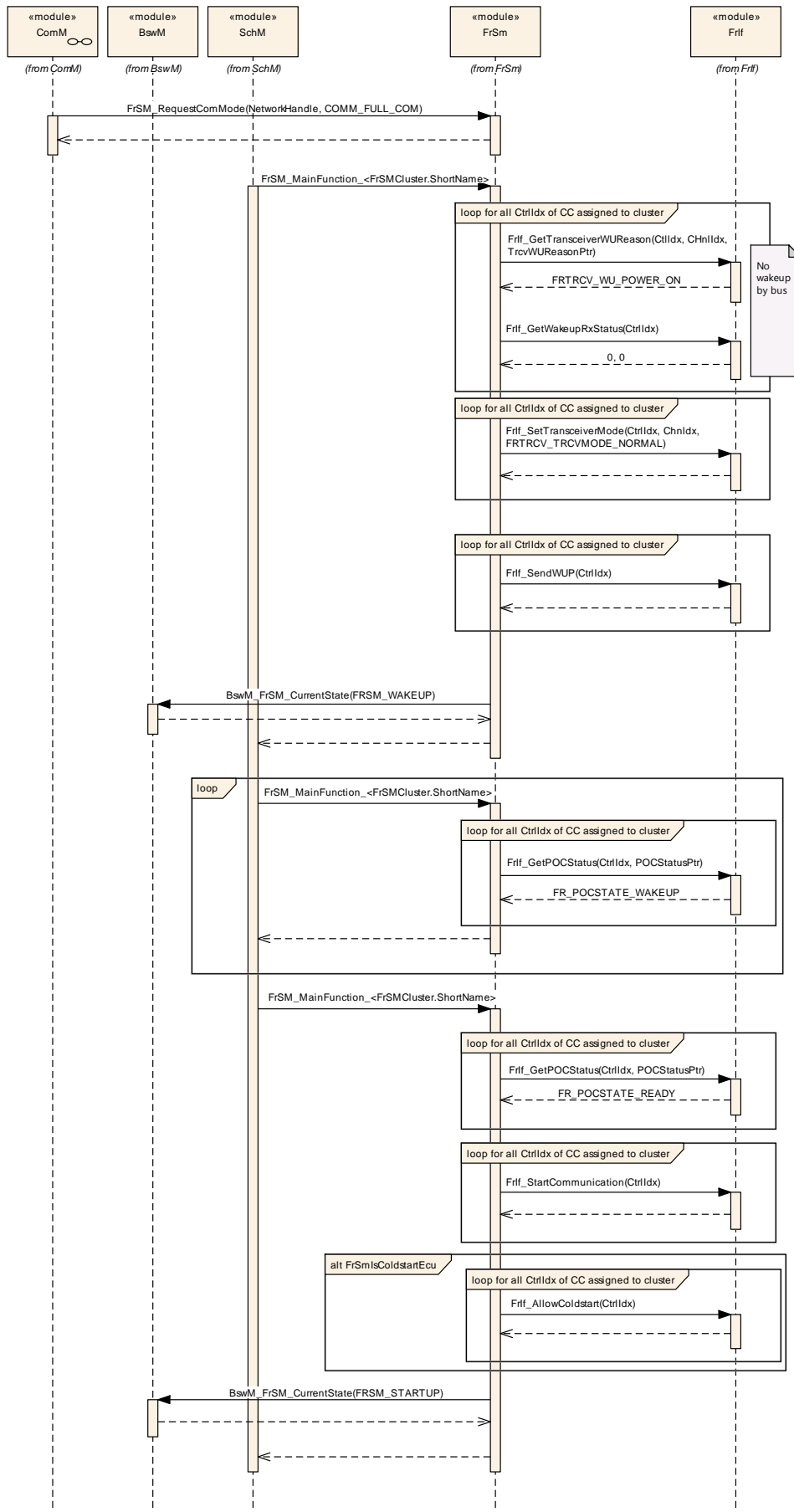
The name of this function can be configured using the configuration parameter `FrMmSyncLossErrorIndicationName` (see chapter 10). The FlexRay State Manager will call this function when the ECU loses its synchronization to the FlexRay cluster, after it could synchronize to the FlexRay cluster or when the `FullCom` request is released after the ECU lost its synchronization to the FlexRay cluster.

## 9 Sequence diagrams

### 9.1 Initialization



## 9.2 Single Channel Wakeup





(continued)

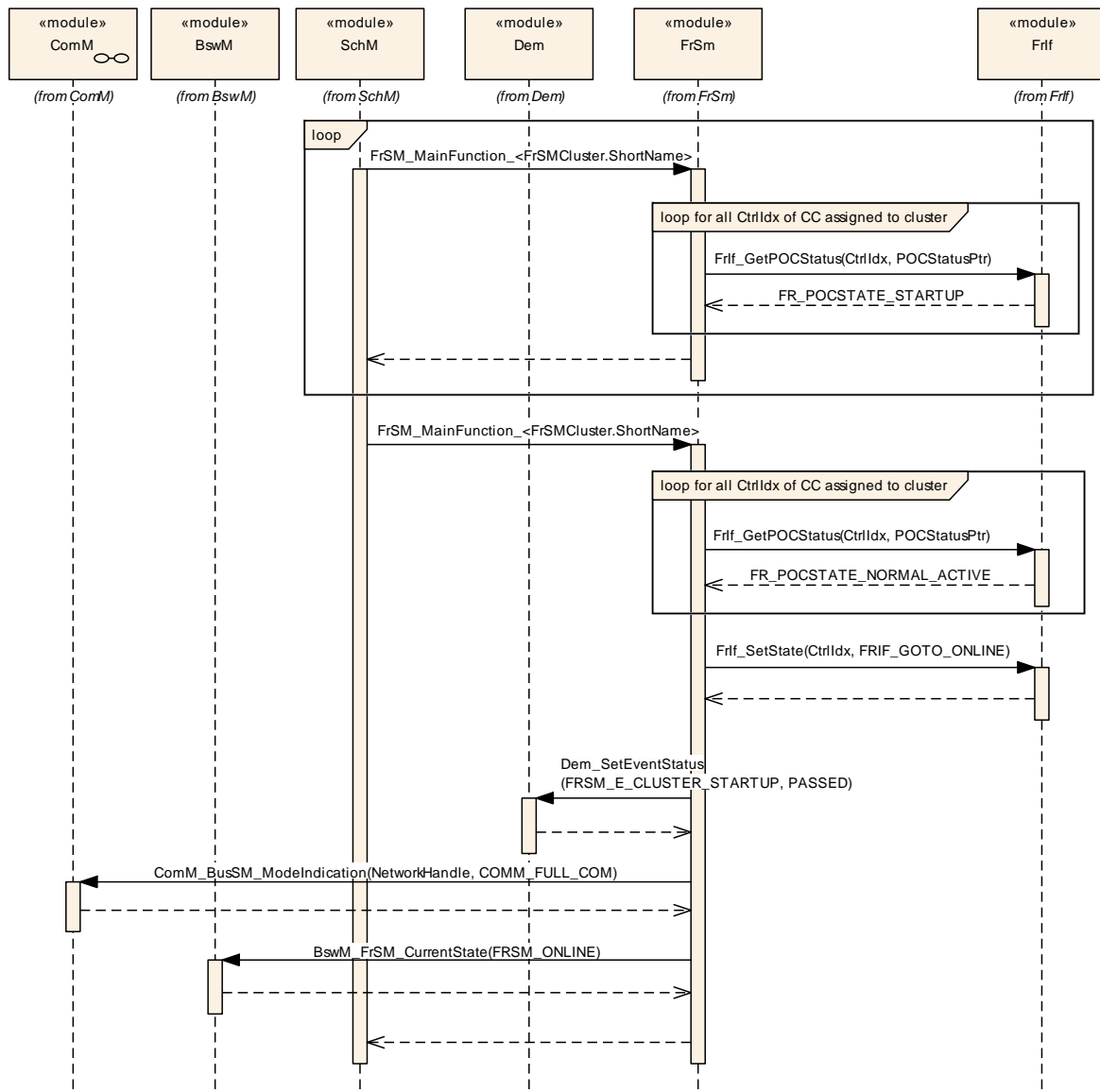
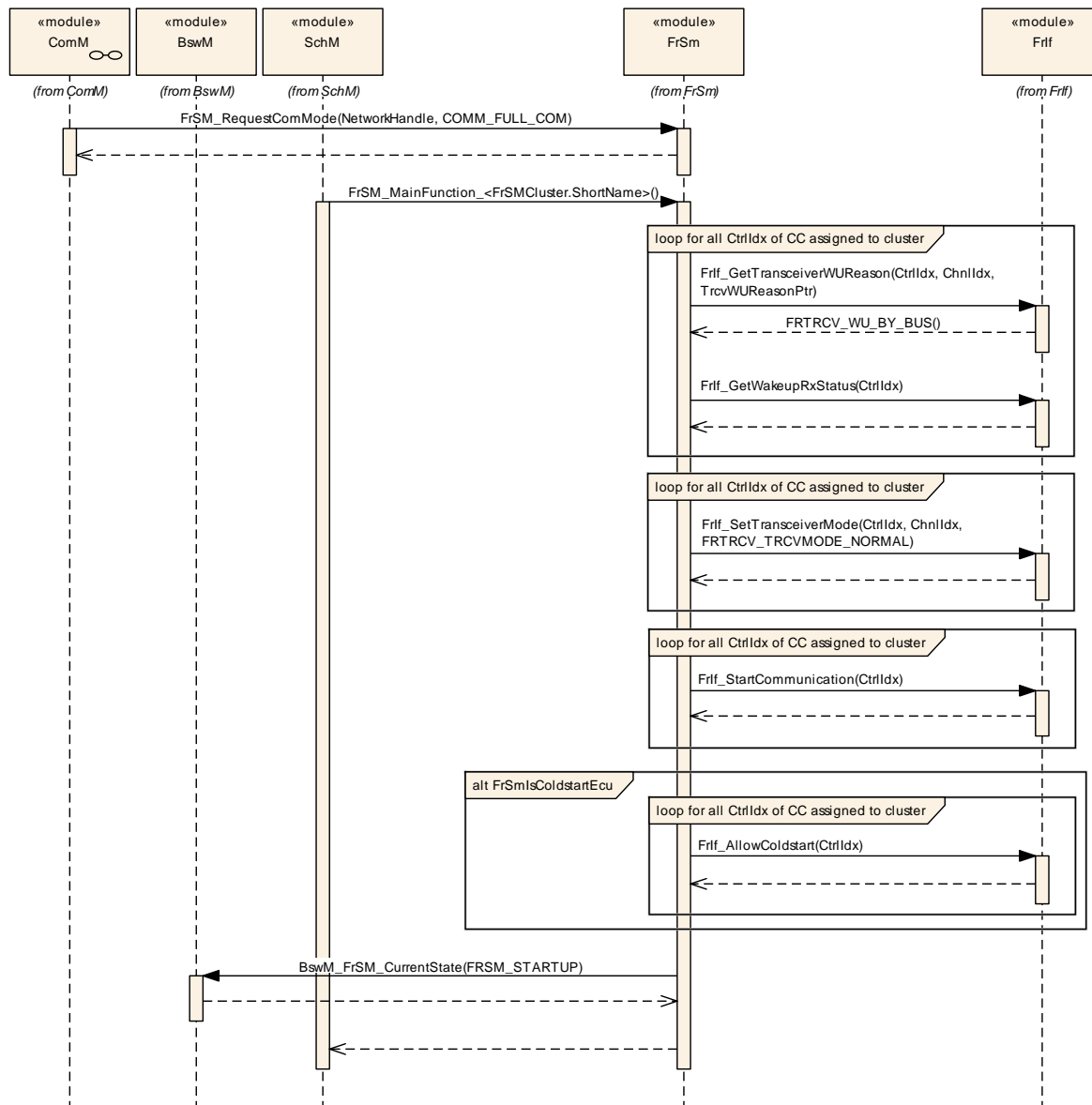
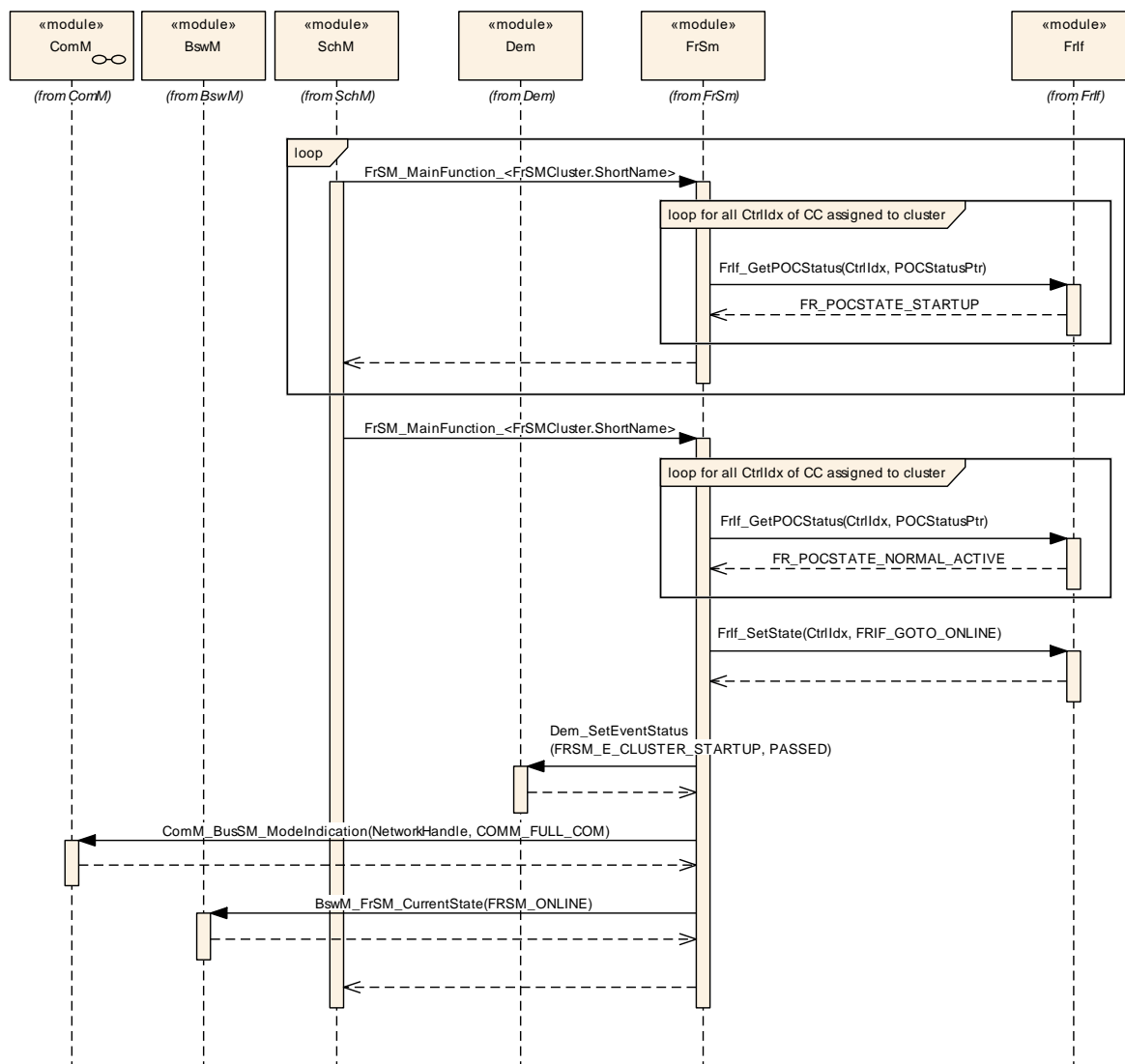


Figure 3 Transition from no communication to full communication for the case of an ECU that has a local wakeup reason.

### 9.3 Single Channel Passive Startup

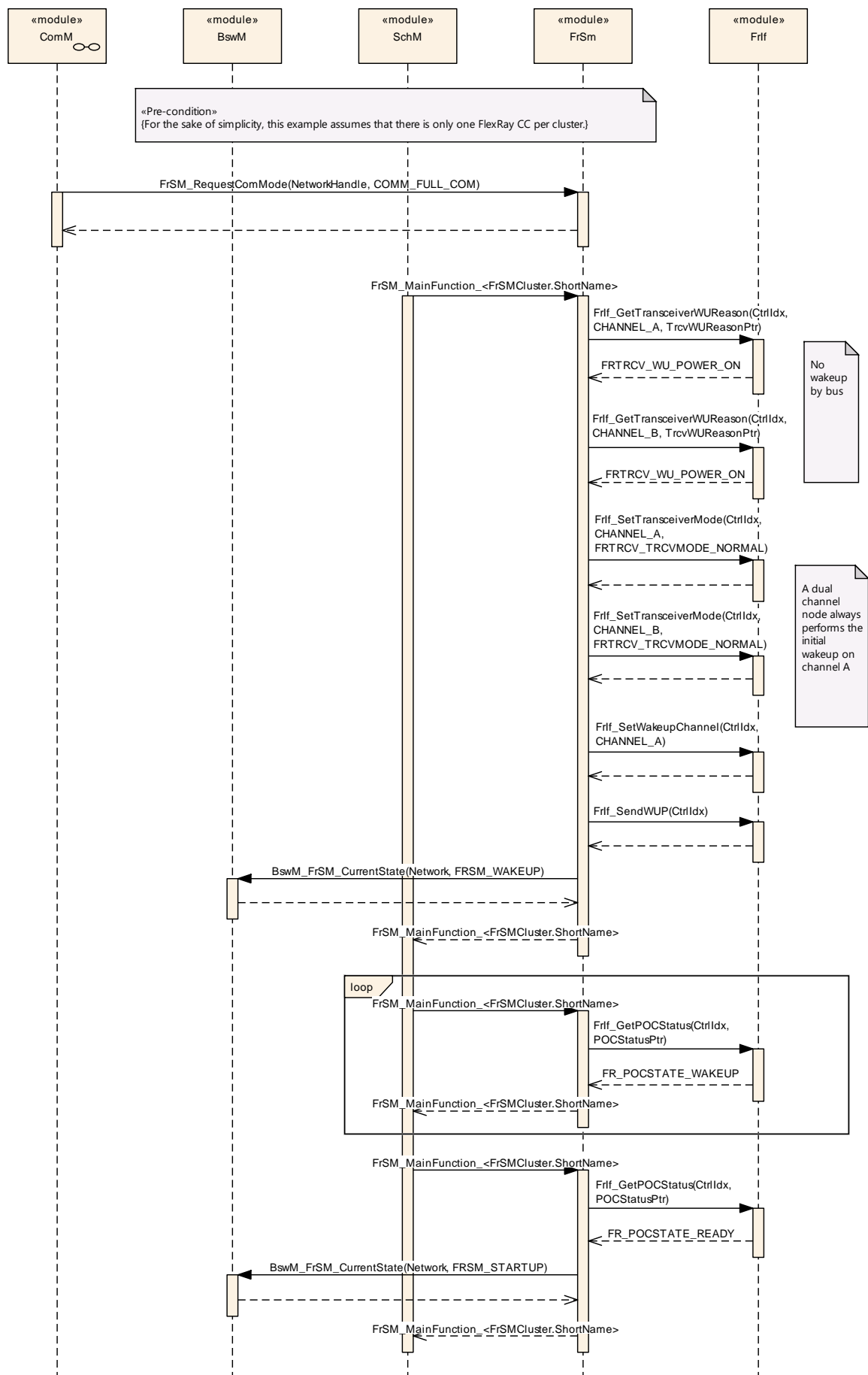


(continued)

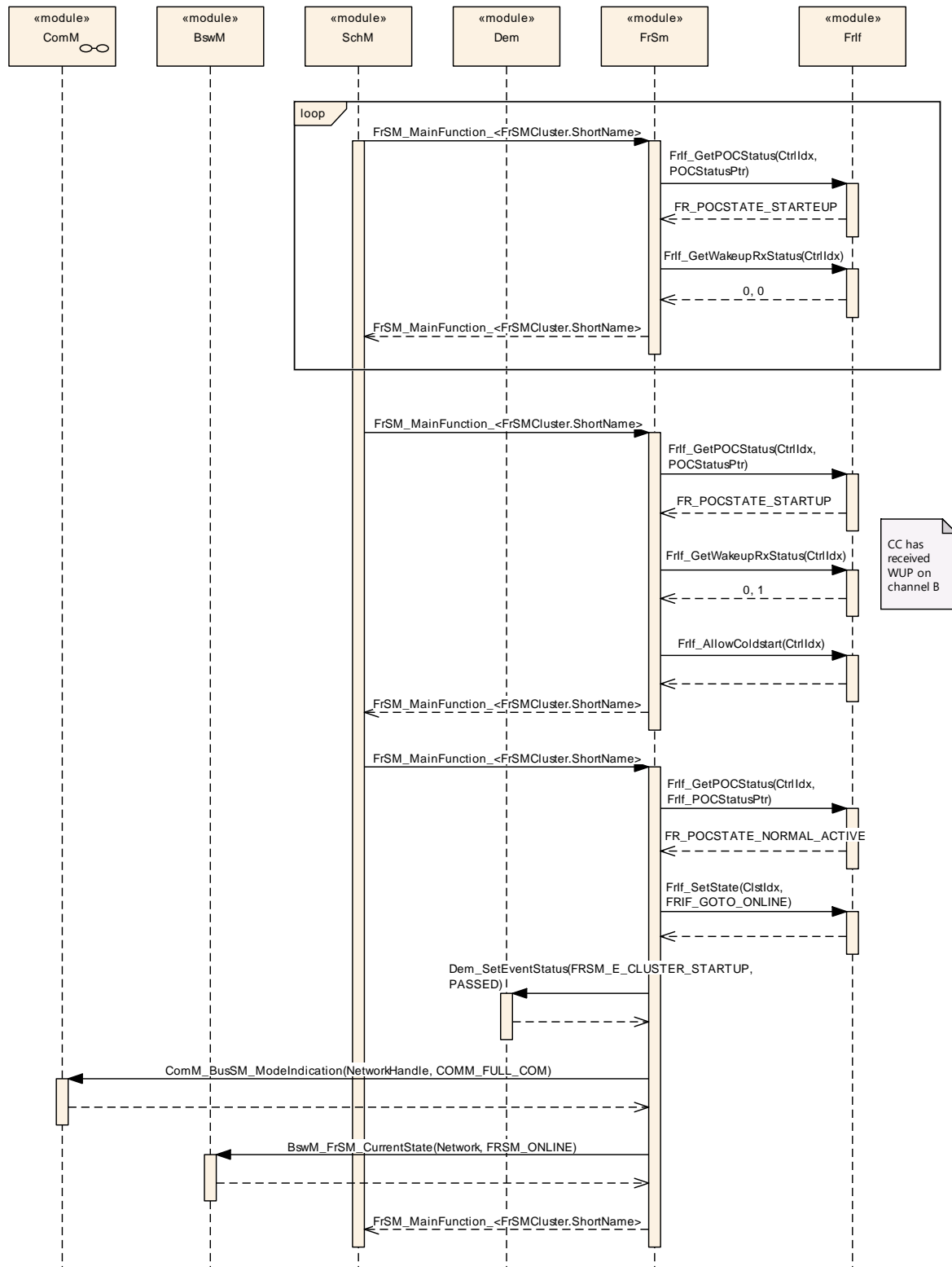


**Figure 4 Transition from no communication to full communication for the case of an ECU that has been woken up by bus.**

## 9.4 Dual Channel Wakeup

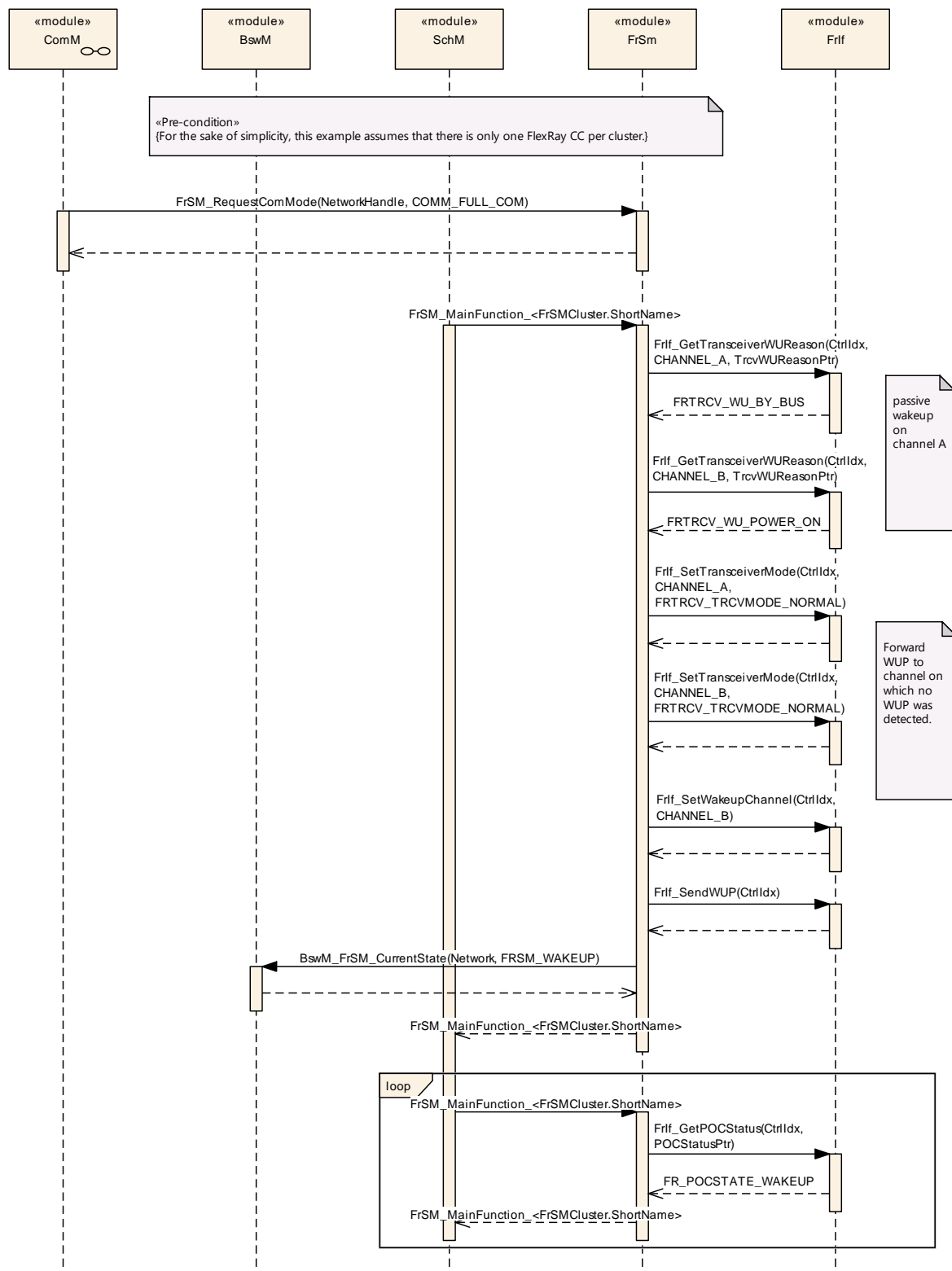


(continued)

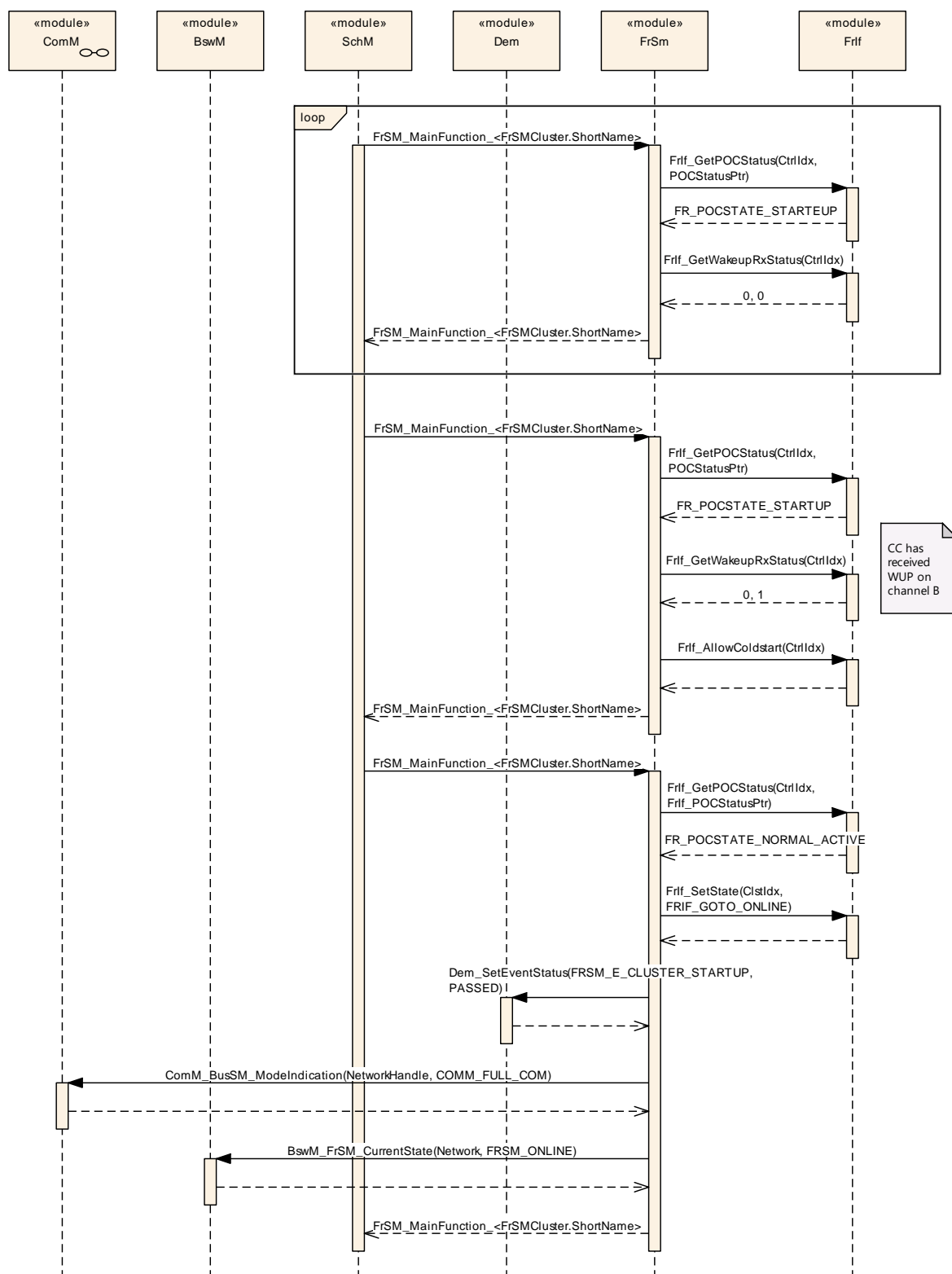


**Figure 5 Transition from no communication to full communication for the case of a dual channel ECU with a local wakeup reason.**

## 9.5 Dual Channel Wakeup Forward



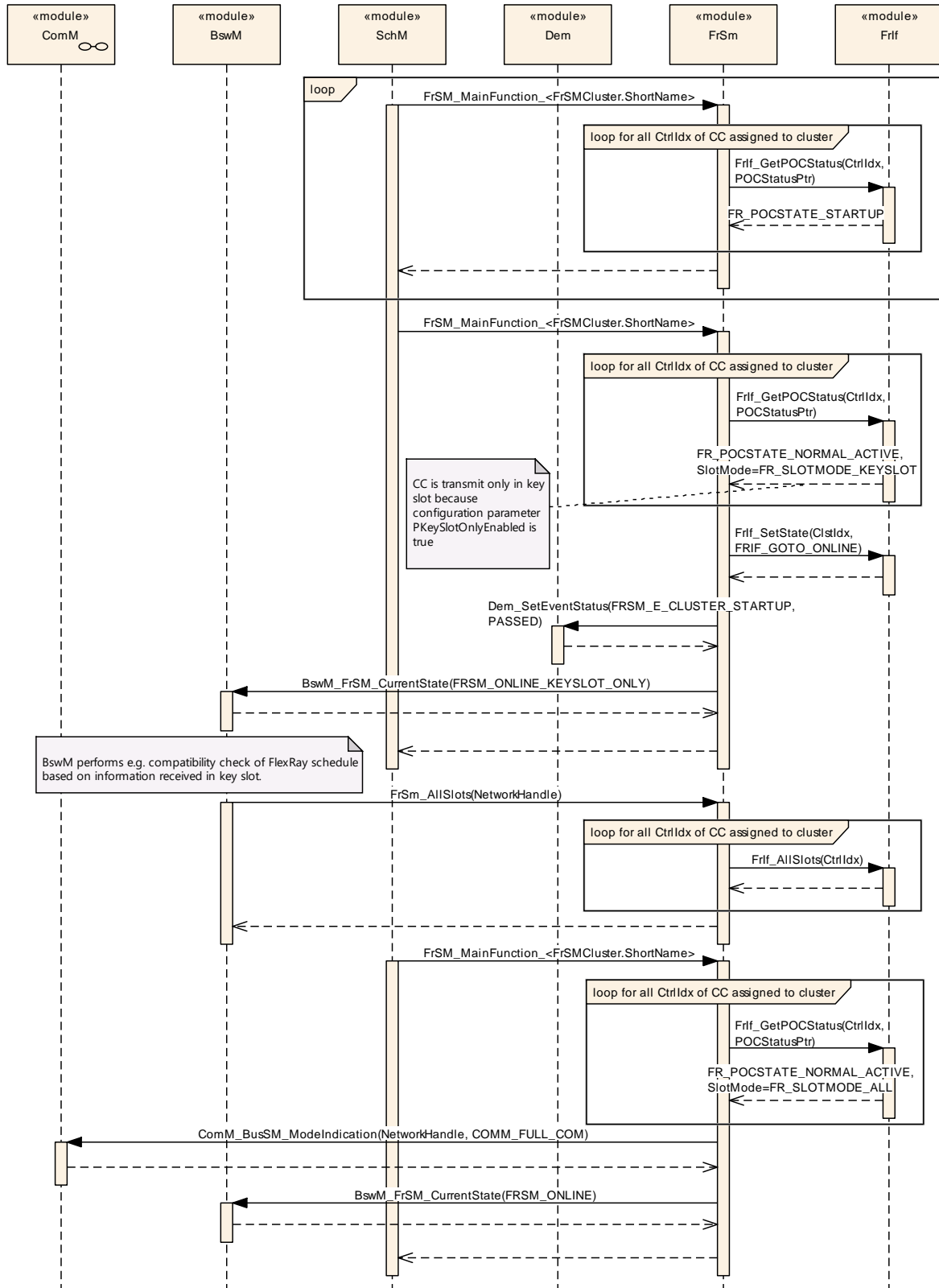
(continued on next page)



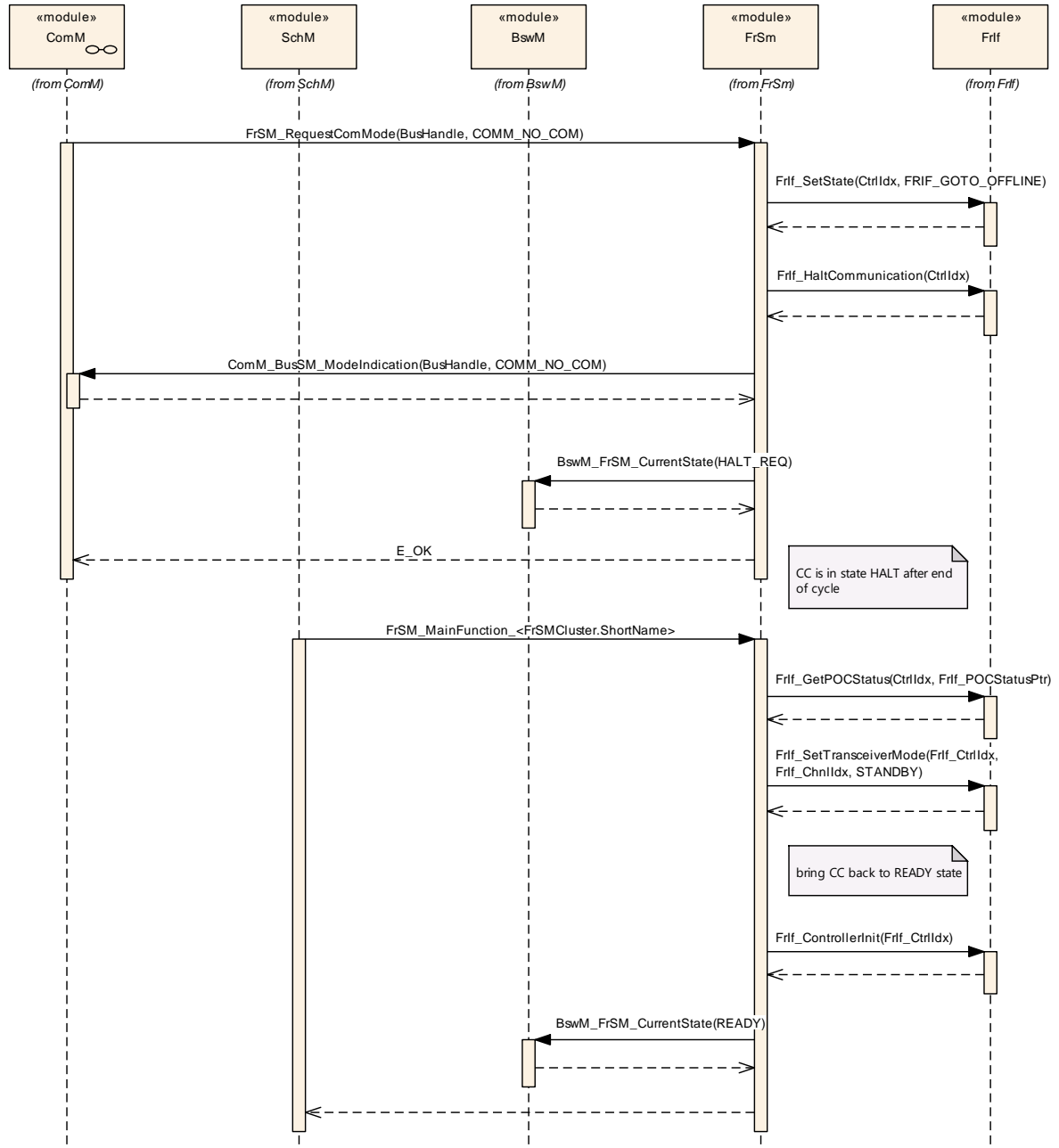
**Figure 6 Transition from no communication to full communication for the case of a dual channel that has been woken up by bus.**



## 9.6 Key Slot Only Mode



## 9.7 Transition from full communication to no communication



## 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 *SWS\_BSWGeneral*.

### 10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters are described Chapters 0 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

<b>SWS Item</b>	<b>ECUC_FrSM_00174 :</b>
<b>Module Name</b>	FrSM
<b>Module Description</b>	Configuration of the FlexRay State Manager
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	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.

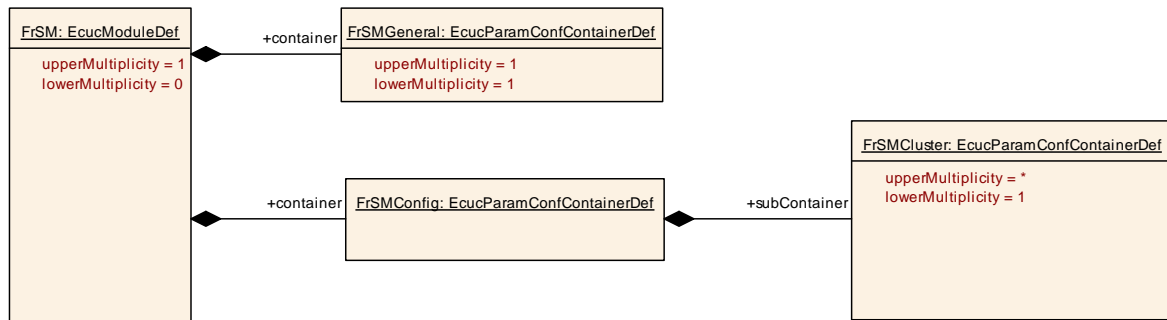


Figure 7 FlexRay State Manager Configuration

### 10.2.2 FrSMConfig

<b>SWS Item</b>	<b>ECUC_FrSM_00146 :</b>
<b>Container Name</b>	FrSMConfig
<b>Parent Container</b>	FrSM
<b>Description</b>	This container comprises the cluster specific configuration of the FlexRay State Manager.
<b>Configuration Parameters</b>	

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

<b>SWS Item</b>	<b>ECUC_FrSM_00107 :</b>
<b>Container Name</b>	FrSMGeneral
<b>Parent Container</b>	FrSM
<b>Description</b>	This container contains the general configuration parameters of the FlexRay State Manager.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_FrSM_00172 :</b>		
<b>Name</b>	FrSMAllSlotsSupport		
<b>Parent Container</b>	FrSMGeneral		
<b>Description</b>	Configuration parameter to enable/disable FrSM support to enable/disable the switching from key-slot/single-slot mode to all-slot mode.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FrSM_00066 :</b>		
<b>Name</b>	FrSMDevErrorDetect		
<b>Parent Container</b>	FrSMGeneral		
<b>Description</b>	Switches the development error detection and notification on or off. <ul style="list-style-type: none"> <li>true: detection and notification is enabled.</li> <li>false: detection and notification is disabled.</li> </ul>		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FrSM_00167 :</b>		
<b>Name</b>	FrSMSyncLossErrorIndicationName		
<b>Parent Container</b>	FrSMGeneral		
<b>Description</b>	Name of <Cdd>_SyncLossErrorIndication function that shall be called on loss of synchronization. If this parameter is omitted no indication shall take place.		
<b>Multiplicity</b>	0..1		

<b>Type</b>	EcucFunctionNameDef		
<b>Default value</b>	--		
<b>maxLength</b>	--		
<b>minLength</b>	--		
<b>regularExpression</b>	--		
<b>Post-Build Variant Multiplicity</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC FrSM 00108 :</b>		
<b>Name</b>	FrSMVersionInfoApi		
<b>Parent Container</b>	FrSMGeneral		
<b>Description</b>	Enables and disables the version info API		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

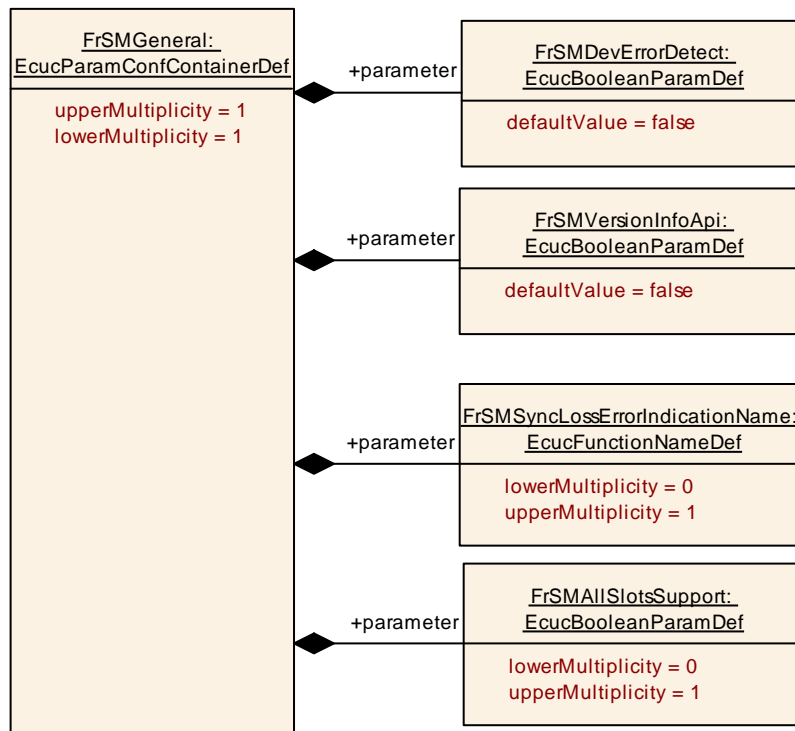


Figure 8 FrSMGeneral Container

## 10.2.4 FrSMCluster

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

<b>SWS Item</b>	<b>ECUC_FrSM_00001 :</b>		
<b>Name</b>	FrSMCheckWakeupReason		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

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

<b>SWS Item</b>	<b>ECUC_FrSM_00102 :</b>		
<b>Name</b>	FrSMDurationT1		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	The duration of timer t1 in seconds. A value of 0 shall imply that the timer is not used.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	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).
--	--

<b>SWS Item</b>	<b>ECUC FrSM_00089 :</b>		
<b>Name</b>	FrSMDurationT2		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	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).		

<b>SWS Item</b>	<b>ECUC FrSM_00162 :</b>		
<b>Name</b>	FrSMDurationT3		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	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).		

<b>SWS Item</b>	<b>ECUC FrSM_00173 :</b>		
<b>Name</b>	FrSMDurationT4		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	The timer t4 ensures that a dual channel node will eventually clear its coldstart inhibit bit and become a leading coldstarter.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. INF]		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FrSM_00068 :</b>		
<b>Name</b>	FrSMIsColdstartEcu		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	True: The ECU is a coldstart node for this FlexRay cluster. False: The ECU is no coldstart node for this FlexRay cluster.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

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

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

<b>SWS Item</b>	<b>ECUC_FrSM_00168 :</b>		
<b>Name</b>	FrSMMinNumberOfColdstarter		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	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.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<b>Default value</b>	--		
<b>Post-Build Variant</b>	true		

<b>Multiplicity</b>			
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local		

SWS Item	ECUC_FrSM_00165 :		
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.		

<b>SWS Item</b>	<b>ECUC_FrSM_00069 :</b>		
<b>Name</b>	FrSMStartupRepetitions		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	The number of times an ECU may repeat the startup procedure for a FlexRay cluster.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 65535		
<b>Default value</b>	--		
<b>Post-Build Variant Multiplicity</b>	true		
<b>Post-Build Variant Value</b>	true		
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME
	<b>Post-build time</b>	X	VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: This value must be greater or equal to FrSMStartupRepetitionsWithWakeup		

<b>SWS Item</b>	<b>ECUC_FrSM_00094 :</b>		
<b>Name</b>	FrSMStartupRepetitionsWithWakeup		
<b>Parent Container</b>	FrSMCluster		
<b>Description</b>	The number of times an ECU may repeat the startup procedure including a wakeup for a FlexRay cluster.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucIntegerParamDef		

<b>Range</b>	0 .. 65535	
<b>Default value</b>	--	
<b>Post-Build Variant Multiplicity</b>	true	
<b>Post-Build Variant Value</b>	true	
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X VARIANT-PRE-COMPILE
	<b>Link time</b>	X VARIANT-LINK-TIME
	<b>Post-build time</b>	X VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X VARIANT-PRE-COMPILE
	<b>Link time</b>	X VARIANT-LINK-TIME
	<b>Post-build time</b>	X VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local	

<b>SWS Item</b>	<b>ECUC_FrSM_00170 :</b>	
<b>Name</b>	FrSMTrcvStdbyDelay	
<b>Parent Container</b>	FrSMCluster	
<b>Description</b>	The duration of timer t_TrvcStdbyDelay 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.	
<b>Multiplicity</b>	0..1	
<b>Type</b>	EcucFloatParamDef	
<b>Range</b>	[0 .. INF]	
<b>Default value</b>	--	
<b>Post-Build Variant Multiplicity</b>	true	
<b>Post-Build Variant Value</b>	true	
<b>Multiplicity Configuration Class</b>	<b>Pre-compile time</b>	X VARIANT-PRE-COMPILE
	<b>Link time</b>	X VARIANT-LINK-TIME
	<b>Post-build time</b>	X VARIANT-POST-BUILD
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X VARIANT-PRE-COMPILE
	<b>Link time</b>	X VARIANT-LINK-TIME
	<b>Post-build time</b>	X VARIANT-POST-BUILD
<b>Scope / Dependency</b>	scope: local dependency: FrSmMainFunctionCycleTime	

<b>SWS Item</b>	<b>ECUC_FrSM_00070 :</b>	
<b>Name</b>	FrSMComMNetworkHandleRef	
<b>Parent Container</b>	FrSMCluster	
<b>Description</b>	Reference to the unique handle to identify one certain FlexRay network correspond to one of the network handles of the ComM configuration.	
<b>Multiplicity</b>	1	
<b>Type</b>	Symbolic name reference to [ ComMChannel ]	
<b>Post-Build Variant Value</b>	false	
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X VARIANT-PRE-COMPILE
	<b>Link time</b>	X VARIANT-LINK-TIME, VARIANT-POST-BUILD
	<b>Post-build time</b>	--
<b>Scope / Dependency</b>	scope: local	

<b>SWS Item</b>	<b>ECUC_FrSM_00116 :</b>	
<b>Name</b>	FrSMFrIfClusterRef	
<b>Parent Container</b>	FrSMCluster	
<b>Description</b>	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.	
<b>Multiplicity</b>	1	

<b>Type</b>	Symbolic name reference to [ FrlfCluster ]		
<b>Post-Build Variant Value</b>	true		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
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.

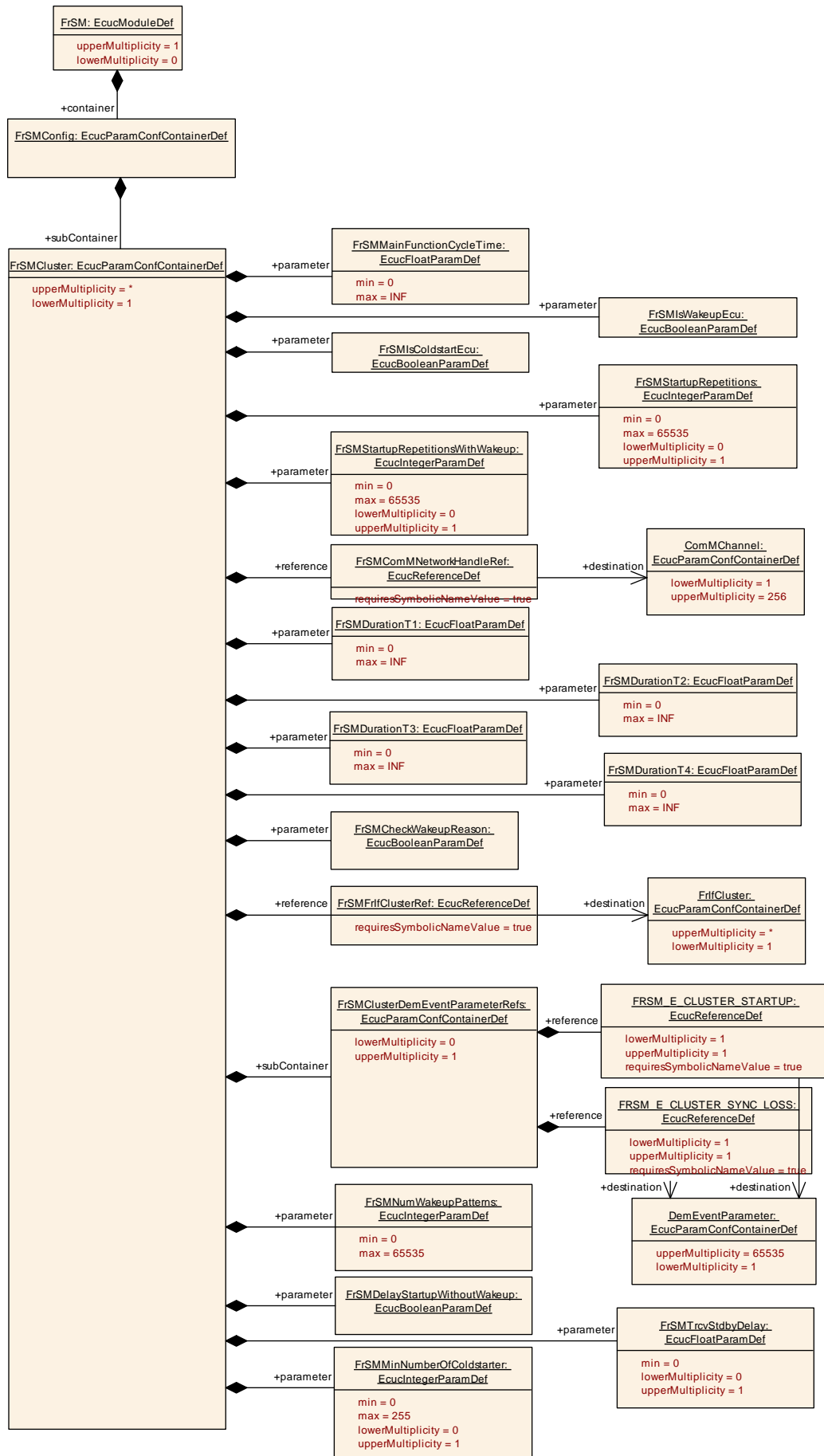


Figure 9 FrSMCluster Container

### 10.2.5 FrSMClusterDemEventParameterRefs

<b>SWS Item</b>	<b>ECUC_FrSM_00163 :</b>
<b>Container Name</b>	FrSMClusterDemEventParameterRefs
<b>Parent Container</b>	FrSMCluster
<b>Description</b>	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.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_FrSM_00164 :</b>		
<b>Name</b>	FRSM_E_CLUSTER_STARTUP		
<b>Parent Container</b>	FrSMClusterDemEventParameterRefs		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_FrSM_00169 :</b>		
<b>Name</b>	FRSM_E_CLUSTER_SYNC_LOSS		
<b>Parent Container</b>	FrSMClusterDemEventParameterRefs		
<b>Description</b>	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.		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<b>Post-Build Variant Value</b>	false		
<b>Value Configuration Class</b>	<b>Pre-compile time</b>	X	VARIANT-PRE-COMPILE
	<b>Link time</b>	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

**No Included Containers**

### 10.3 Published Information

For details refer to the chapter 10.3 “Published Information” in *SWS\_BSWGeneral*.



## 11 Not applicable requirements

**[SWS\_FrSM\_00186]** [ These requirements are not applicable to this specification. ]

(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)