Document Title	Specification of FlexRay State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	254
Document Classification	Standard
Document Version	2.5.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
31.03.2014	2.5.0	AUTOSAR Release Management	• Removed Dual Channel Wakeup Echo
31.10.2013	2.4.0	AUTOSAR Release Management	• Added immediate handling of NoCom requests in normal passive mode or key slot only mode • Editorial changes • Removed chapter(s) on change documentation
27.02.2013	2.3.0	AUTOSAR Administration	• FlexRay Transceiver Mode Switch can be delayed • Formal updates
09.12.2011	2.2.0	AUTOSAR Administration	• Short term loss of synchronization is reported to DEM or DET. • Number of startup frames can be monitored during normal operation. • Revised production error handling.
03.11.2010	2.1.0	AUTOSAR Administration	• The amount of wakeup patterns can be configured • Clearing the Coldstart Inhibit Mode can be delayed also for passive wakeup. • Removed enabling and disabling of transceiver wakeups

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
30.11.2009	2.0.0	AUTOSAR Administration	• Added support of FlexRay Dual Channel Wakeup • Added support of FlexRay Single Slot Mode • Added support of Passive Mode (Receive only) • Improved timeout supervision of FlexRay startup • Legal disclaimer revised
23.06.2008	1.0.2	AUTOSAR Administration	Legal disclaimer revised
01.02.2008	1.0.1	AUTOSAR Administration	Chapter 8 API Spelling harmonized
20.11.2007	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

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

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

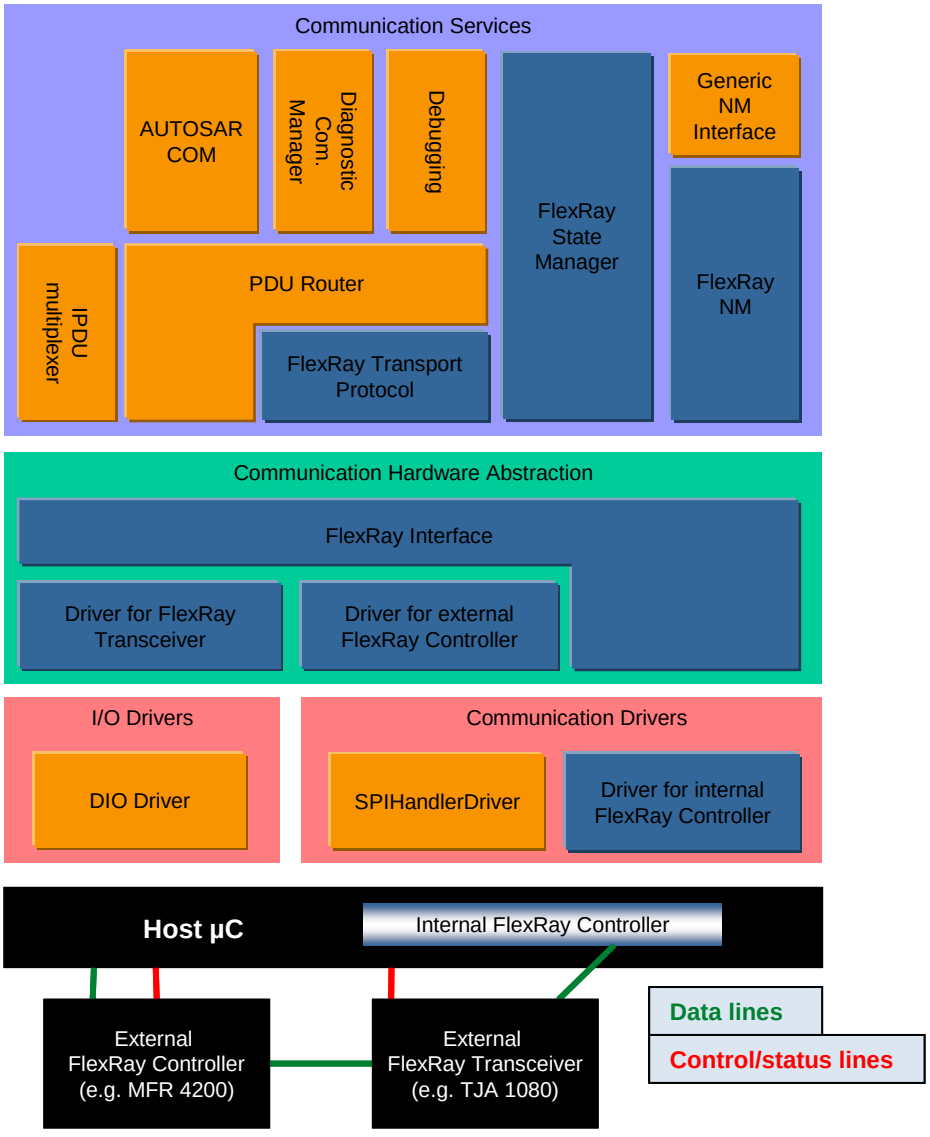


Figure 1 Software Architecture Overview

2 Acronyms and abbreviations

Acronym/ Abbrevation	*Description:*
API	Application Program Interface
AUTOSAR	Automotive Open System Architecture
BSW	Basic Software
CC	Communication Controller
CHI	Controller Host Interface
ComM	AUTOSAR Communication Manager
DCM	Diagnostic Communication Manager
Dem/DEM	Diagnostic Event Manager
Det/DET	Development Error Tracer
e.g.	[lat.] exempli gratia = [eng.] for example
ECU	Electronic Control Unit
EcuM	ECU State Manager
Fr	FlexRay Driver
FrIf	FlexRay Interface (AUTOSAR BSW module)
FrSM	FlexRay State Manager
FrTrcv	FlexRay Transceiver Driver
i.e.	[lat.] id est = [eng.] that is
Id/ID	Identifier
N/A	Not applicable
NM	Network Management
PDU	Protocol Data Unit
POC	Protocol Operation Control
POCState	Actual CC internal state of the POC. This state might differ from vPOC!State in certain cases, e.g. after FREEZE command invocation (see [11] for details).
RTE	Runtime Environment
RX	Reception
SchM	Schedule Manager
SW	Software
TX	Transmission
UML	Unified Modeling Language
vPOC	Data structure provided from the CC to the host at the CHI, which contains the actual POC status of the CC.
vPOC!Freeze	vPOC!Freeze denotes the Freeze bit that is part of the vPOC data structure. The Freeze bit is used by the CC to indicate that the HALT state has been entered due to an error condition.
vPOC!SlotMode	vPOC!SlotMode denotes the SlotMode field that is part of the vPOC data structure.
WUP	Wake-Up Pattern
XML	Extensible markup language

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

3 Related documentation

3.1 Input documents

[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 FrIf to initialize the FlexRay Communication Hardware and to control the operating modes of the FlexRay Controllers and FlexRay Transceivers assigned to the FlexRay Networks.

5.4 AUTOSAR Development Error Tracer

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

5.5 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

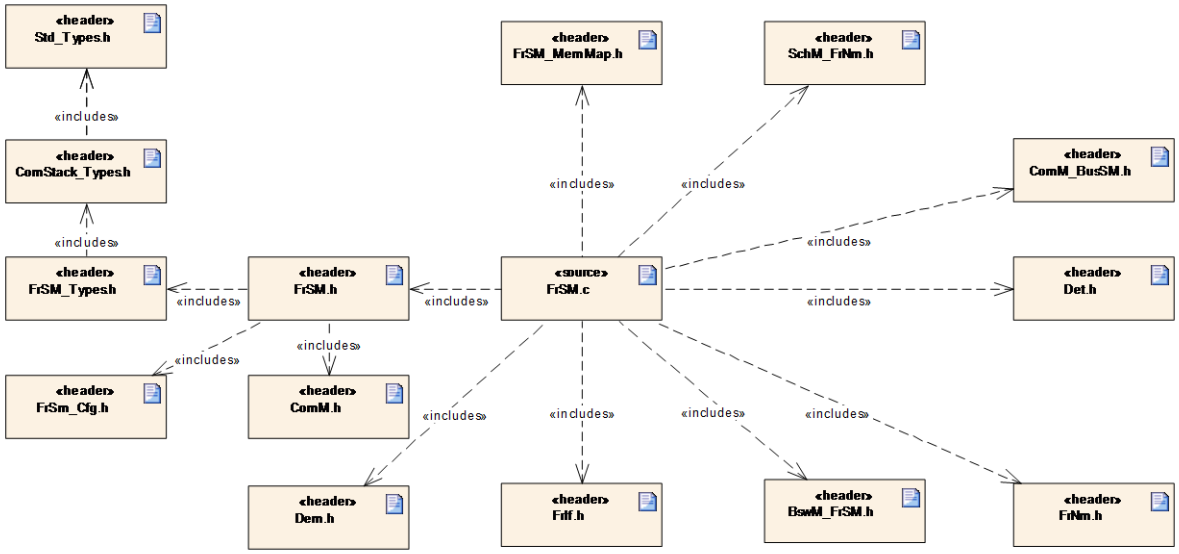
In order to be able to report startup failures the FlexRay State Manager has to have access to the FlexRay Network Management.

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_00120] ⌈The header file FrSM.h shall export the API of the FrSM module.⌋ ()

[SWS_FrSM_00121] ⌈The header file FrSM.h shall include FrSM_Types.h and FrSM_Cfg.h. ⌋ ()

[SWS_FrSM_00054] ⌈The header file FrSM_Types.h shall export the FrSM specific types. ⌋ ()

[SWS_FrSM_00055] ⌈The FrSM implementation (FrSM.c) shall include its header file FrSM.h to get access to its own API declaration and to its configuration parameters. ⌋ ()

[SWS_FrSM_00058] ⌈The FrSM implementation (FrSM.c) shall include the header file FrIf.h to get access to the FrIf API. ⌋ ()

[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
-	-	SWS_FrSM_00015
-	-	SWS_FrSM_00019
-	-	SWS_FrSM_00021
-	-	SWS_FrSM_00022
-	-	SWS_FrSM_00025
-	-	SWS_FrSM_00026
-	-	SWS_FrSM_00027
-	-	SWS_FrSM_00030
-	-	SWS_FrSM_00032
-	-	SWS_FrSM_00047
-	-	SWS_FrSM_00048
-	-	SWS_FrSM_00054
-	-	SWS_FrSM_00055
-	-	SWS_FrSM_00058
-	-	SWS_FrSM_00093
-	-	SWS_FrSM_00095
-	-	SWS_FrSM_00096
-	-	SWS_FrSM_00097
-	-	SWS_FrSM_00098
-	-	SWS_FrSM_00105
-	-	SWS_FrSM_00120
-	-	SWS_FrSM_00121
-	-	SWS_FrSM_00139
-	-	SWS_FrSM_00141
-	-	SWS_FrSM_00142
-	-	SWS_FrSM_00143
-	-	SWS_FrSM_00145
-	-	SWS_FrSM_00149
-	-	SWS_FrSM_00171
-	-	SWS_FrSM_00176
-	-	SWS_FrSM_00177
-	-	SWS_FrSM_00178
-	-	SWS_FrSM_00180
-	-	SWS_FrSM_00190
-	-	SWS_FrSM_00192
-	-	SWS_FrSM_00197

-	-	SWS_FrSM_00198
-	-	SWS_FrSM_00199
-	-	SWS_FrSM_00208
BSW00443	-	SWS_FrSM_00186
BSW00444	-	SWS_FrSM_00186
BSW00446	-	SWS_FrSM_00186
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 æC 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_00326	-	SWS_FrSM_00186
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_FrSM_00186

SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_FrSM_00099, SWS_FrSM_00100
SRS_BSW_00347	A Naming seperation 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_00370	-	SWS_FrSM_00186
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_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_FrSM_00013
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_FrSM_00186
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_FrSM_00013
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_FrSM_00060, SWS_FrSM_00061, SWS_FrSM_00169, SWS_FrSM_00179
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_FrSM_00029
SRS_BSW_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	SWS_FrSM_00186

	parameter is implemented as "const" it should be placed into a separate c-file	
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	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_FrSM_00186
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_FrSM_00186
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_00013, 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_00442	The AUTOSAR architecture shall support standardized debugging and tracing features	SWS_FrSM_00137
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_<Cluster Id> assigned to that cluster (see section 8.5.1). However, as defined in section 8.3.2 , some transitions of the state machine are processed in the context of the FrSM_RequestComMode function in order to achieve a deterministic behavior for shutdown. ⌋ ()

7.3.2 States

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

FrSM Cluster State	*Mapped FlexRay CC state*	*Description*
FRSM_READY	POC:ready	
FRSM_WAKEUP	POC:wake-up	FrSM performs wake-up
FRSM_STARTUP	POC:start-up	FrSM performs startup
FRSM_HALT_REQ	POC:normal active or POC:normal passive	FrSM performs a shutdown
FRSM_ONLINE	POC:normal active	Full Communication
FRSM_ONLINE_PASSIVE	POC:normal passive	Due to clock synchronization errors no data is transmitted or received.
FRSM_KEYSLOT_ONLY	POC:normal active ∧ vPOC!SlotMode ≠ AllSlots	Data can only be transmitted in the key slots.
FRSM_LOW_NUMB-ER_OF_COLDSTARTERS	POC: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:

Passive State	*Description*
FRSM_ECU_ACTIVE	When the FrSM is concurrently in state FRSM_READY, the transceivers are in set into mode FRTRCV_TRCVMODE_STANDBY, otherwise into mode FRTRCV_TRCVMODE_NORMAL
FRSM_ECU_PASSIVE	When the FrSM is concurrently in state FRSM_READY, the transceivers are in set into mode FRTRCV_TRCVMODE_STANDBY, otherwise into mode FRTRCV_TRCVMODE_RECEIVEONLY.

⌋ ()

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

FrSM Cluster State	*Passive State*	FrSM_BswM_StateType *value*
FRSM_READY	FRSM_ECU_ACTIVE	FRSM_READY
FRSM_READY	FRSM_ECU_PASSIVE	FRSM_READY_ECU_PASSIVE
FRSM_WAKEUP	FRSM_ECU_ACTIVE	FRSM_WAKEUP
FRSM_WAKEUP	FRSM_ECU_PASSIVE	FRSM_WAKEUP_ECU_PASSIVE
FRSM_STARTUP	FRSM_ECU_ACTIVE	FRSM_STARTUP
FRSM_STARTUP	FRSM_ECU_PASSIVE	FRSM_STARTUP_ECU_PASSIVE
FRSM_ONLINE	FRSM_ECU_ACTIVE	FRSM_ONLINE
FRSM_ONLINE	FRSM_ECU_PASSIVE	FRSM_ONLINE_ECU_PASSIVE
FRSM_ONLINE_PASSIVE	FRSM_ECU_ACTIVE	FRSM_ONLINE_PASSIVE
FRSM_ONLINE_PASSIVE	FRSM_ECU_PASSIVE	FRSM_ONLINE_PASSIVE_ECU_PASSIVE

FRSM_KEYSLOT_ONLY	FRSM_ECU_ACTIVE	FRSM_KEYSLOT_ONLY
FRSM_KEYSLOT_ONLY	FRSM_ECU_PASSIVE	FRSM_KEYSLOT_ONLY_ECU_PASSIVE
FRSM_HALT_REQUEST	FRSM_ECU_ACTIVE	FRSM_HALT_REQUEST
FRSM_HALT_REQUEST	FRSM_ECU_PASSIVE	FRSM_HALT_REQUEST_ECU_PASSIVE
FRSM_LOW_NUMBER_OF_COLD-STARTERS	FRSM_ECU_ACTIVE	FRSM_LOW_NUMBER_OF_COLDSTARTERS
FRSM_LOW_NUMBER_OF_COLD-STARTERS	FRSM_ECU_PASSIVE	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.

FrSM Variable	*Type*	*Description*
reqComMode	ComM_ModeType	The communication mode that has been requested by the ComM. The communication modes are abbreviated in this document as follows: NoCom: COMM_NO_COMMUNICATION SilentCom:COMM_SILENT_COMMUNICATION FullCom: COMM_FULL_COMMUNICATION According to the definition of ComM_ModeType these modes are ordered as follows: NoCom < SilentCom < FullCom
startupCounter	Integer	The number of startup attempts that have been performed
wakeupType	Enum	The following values are supported: • SingleChannelWakeup • DualChannelWakeup • DualChannelWakeupForward • NoWakeup
wakeupTransmitted	Boolean	True if vPOC!WakeupStatus = FR_WAKEUP_TRANSMITTED for at least attempt to transmit a wakeup pattern, false otherwise
busTrafficDetected	Boolean	True if vPOC!WakeupStatus = FR_WAKEUP_RECEIVED_HEADER or FR_WAKEUP_RECEIVED_WUP for at least attempt to transmit a wakeup pattern, false otherwise
wakeupCounter	Integer	The number of attempts that have been performed for transmitting a wakeup pattern.

Note that the silent communication mode 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:

FrSM Configuration Parameter	*Type*	*Description*
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 ∞
FrSMStartupRepetitions	Integer	Determines how often the ECU can repeat the startup procedure by reinitializing the FlexRay CC, see chapter 10.2. This value must not be smaller than FrSMStartupRepetitionsWithWakeup. If this optional configuration parameter is missing, there shall be no limitation, i.e. the configuration parameter shall be treated as having the value ∞
FrSMNumWakeupPatterns	Integer	Maximum number of Wakeup Patterns the node may send before going to FRSM_STARTUP.

FrSMDelayStartupWithoutWakeup	Boolean	If true, timer t1 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	If FrSMCheckWakeupReason is false, WUReason evaluates to NO_WU_BY_BUS. Otherwise if FrSMCheckWakeupReason is true, determine the wakeup reason by calling FrIf_GetTransceiverWUReason for each transceiver of the FlexRay cluster and check for FRTRCV_WU_BY_BUS and evaluate WUReason to • NO_WU_BY_BUS in case no wakeup has been detected. • PARTIAL_WU_BY_BUS in case the ECU is connected to both FlexRay channels of the cluster and wakeup has been detected for exactly one channel • ALL_WU_BY_BUS in case wakeup has been detected for all of the FlexRay channels of the cluster to which the ECU is connected.
AllChannelsAwake	boolean	Determine the WakeupRxStatus by calling FrIf_GetWakeupRxStatus for each of the FlexRay controllers of the FlexRay cluster and return TRUE if the wakeup status is 1 for that FlexRay channel which has not been woken up by this ECU; otherwise return FALSE.
t1_IsActive	boolean	Evaluates to true if t1 has been started and has not expired yet, otherwise to false
t3_IsNotActive	boolean	Evaluates to false if t3 is running and has not expired, otherwise to true.
t_TrcvStdby-Delay_IsActive	boolean	Evaluates to true if t_TrcvStdbyDelay has been started and has not expired yet, otherwise to false.
wakeupFinished	boolean	Evaluates to false if the wakeup pattern

		transmission as defined in section 7.3.8 is still in progress, otherwise to true.
lowNumberOfColdstarters	boolean	= FrIf_GetNumOfStartupFrames() < FrSMMinNumberOfColdstarter

7.3.6 Timers

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

Timer	*Description*
t1	The timer t1 models the delay of clearing the coldstart inhibit mode (i.e. calling FrIf_AllowColdstart). The duration of this timer can be statically configured with the configuration parameter FrSMDurationT1.
t2	The timer t2 models the time difference after which the FrSM will repeat the startup of the FlexRay cluster. The duration of this timer can be statically configured with the configuration parameter FrSMDurationT2.
t3	The timer t3 supervises the transition to FullCom. The duration of this timer can be statically configured with the configuration parameter FrSMDurationT3.
t_TrcvStdbyDelay	The timer t_TrcvStdbyDelay 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 t1shall 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. ⌋()

7.3.7 Functional Elements

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

Functional Element	*Description*
FE_WAKEUP	Call FrIf_SendWUP for each controller of the

	FlexRay cluster.
FE_SET_WU_CHANNEL_INITIAL	In case of a single channel node, do nothing. In case of a dual channel node, call FrIf_SetWakeupChannel for each controller of the FlexRay cluster in order to set the wakeup channel to the channel A.
FE_SET_WU_CHANNEL_FORWARD	In case of a single channel node, do nothing. In case of a dual channel node, call FrIf_SetWakeupChannel for each controller of the FlexRay cluster in order to set the wakeup channel to the channel on which no wakeup has been detected while evaluating WUReason.
FE_CONFIG	Call FrIf_ControllerInit for each controller of the FlexRay cluster.
FE_START	Call FrIf_StartCommunication for each controller of the FlexRay cluster.
FE_ALLOW_COLDSTART	Call FrIf_AllowColdstart for each controller of the FlexRay cluster if the configuration parameter FrSMIsColdstartEcu is true.
FE_HALT	Call FrIf_HaltCommunication for each controller of the FlexRay cluster.
FE_TRCV_STANDBY	Call FrIf_SetTransceiverMode with FrIf_TrcvMode as FRTRCV_TRCVMODE_STANDBY for each transceiver of the FlexRay cluster.
FE_TRCV_NORMAL	In case the FrSM state machine is in state FRSM_ECU_ACTIVE, call FrIf_SetTransceiverMode with FrIf_TrcvMode as FRTRCV_TRCVMODE_NORMAL and FrIf_ClearTransceiverWakeup for each transceiver of the FlexRay cluster. In case the FrSM state machine is in state FRSM_ECU_PASSIVE, call FrIf_SetTransceiverMode with FrIf_TrcvMode as FRTRCV_TRCVMODE_RECEIVEONLY and FrIf_ClearTransceiverWakeup for each transceiver of the FlexRay cluster.
FE_START_FRIF	Set the FrIf state to ONLINE by calling FrIf_SetState with FrIf_StateTransition as FRIF_GOTO_ONLINE for the cluster.
FE_STOP_FRIF	Set the FrIf state to OFFLINE by calling FrIf_SetState with FrIf_StateTransition as FRIF_GOTO_OFFLINE for the cluster.
FE_DEM_STATUS_FAILED	Report status of production error FRSM_E_CLUSTER_STARTUP as failed.
FE_DEM_STATUS_PASSED	Report status of production error FRSM_E_CLUSTER_STARTUP as passed.
FE_DEM_SYNC_LOSS	Report the status of the production error FRSM_E_CLUSTER_SYNC_LOSS as failed. If the name of an indication function (see section 8.6.3) is configured, call the indication function with the parameter SyncLossErrorStatus = true.
FE_DEM_SYNC_LOSS_PASSED	If the name of an indication function (see section 8.6.3) is configured, call the indication function with the parameter SyncLossErrorStatus = false.

	Additionally report the status of the production error FRSM_E_CLUSTER_SYNC_LOSS as passed.
FE_FULL_COM_IND	Indicate to the ComM that FullCom has been reached by calling ComM_BusSM_ModeIndication (FullCom)
FE_NO_COM_IND	Indicate to the ComM that FullCom has been left by calling ComM_BusSM_ModeIndication (NoCom).
FE_STARTUP_ERROR_IND	Call FrNm_StartupError.

7.3.8 Wakeup Pattern Transmission

[SWS_FrSM_00208]⌈ The 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`	busTrafficDetected := true
`FR_WAKEUP_TRANSMITTED`	wakeupTransmitted := true
`FR_WAKEUP_UNDEFINED FR_WAKEUP_COLLISION_HEADER FR_WAKEUP_COLLISION_WUP FR_WAKEUP_COLLISION_UNKNOWN`	No action

  - If busTrafficDetected = false and wakeupCounter < FrSMNumWakeup-Patterns, 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.

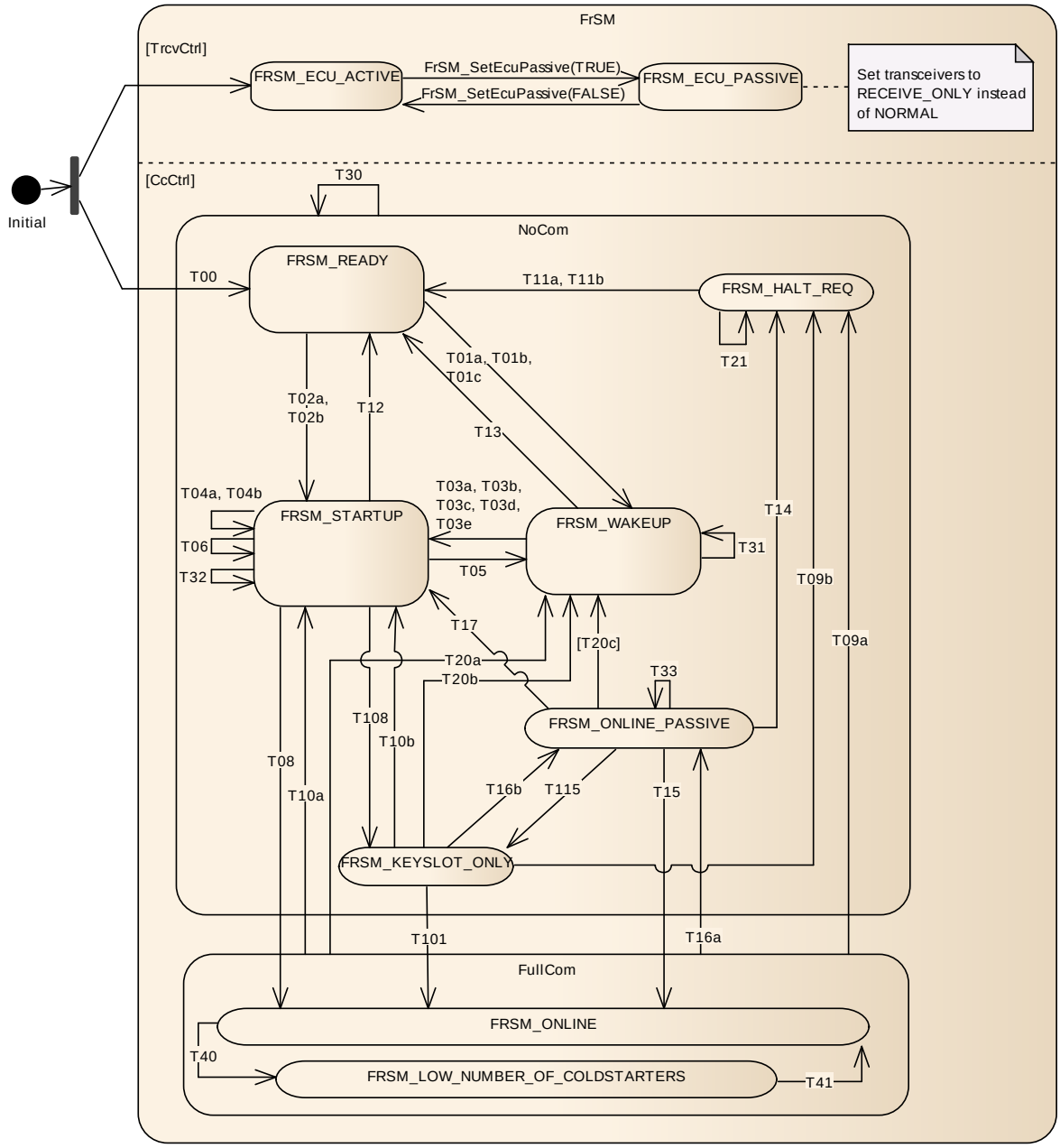


Figure 2 FrSM state machine of the FlexRay State Manager

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.

Tran-sition	***Event [Condition]***	***Actions***
T00	FrSM_Init()	FE_CONFIG
T01 (a)	[reqComMode = FullCom ∧ FrSMIsWakeupEcu ∧ WUReason = NO_WU_BY_BUS ∧ ¬ FrSMIsDualChannelNode]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := SingleChannelWakeup wakeupTransmitted := false FE_WAKEUP start t1 start t3
T01 (b)	[reqComMode = FullCom ∧ FrSMIsWakeupEcu ∧ WUReason = NO_WU_BY_BUS ∧FrSMIsDualChannelNode]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := DualChannelWakeup FE_SET_WU_CHANNEL_INITIAL wakeupTransmitted := false FE_WAKEUP start t3
T01 (c)	[reqComMode = FullCom ∧FrSMIsWakeupEcu ∧ WUReason = PARTIAL_WU_BY_BUS]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := DualChannelWakeupForward FE_SET_WU_CHANNEL_FORWARD FE_WAKEUPwakeupTransmitted := false FE_WAKEUP start t3
T02 (a)	[reqComMode = FullCom ∧ (¬ FrSMIsWakeupEcu ∨ WUReason = ALL_WU_BY_BUS) ∧ ¬ FrSMDelayStartupWithoutWakeup]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := NoWakeup FE_START FE_ALLOW_COLDSTART start t2 start t3
T02 (b)	[reqComMode = FullCom ∧ (¬ FrSMIsWakeupEcu ∨ WUReason = ALL_WU_BY_BUS) ∧ FrSMDelayStartupWithoutWakeup]	FE_TRCV_NORMAL startupCounter := 1 wakeupType := NoWakeup FE_START start t1 start t2 start t3
T03 (a)	[wakeupFinished ∧ reqComMode = FullCom ∧ FrSMNumWakeupPatterns = 1 ∧ wakeupType = SingleChannelWakeup]	FE_START cancel t1 start t1 start t2
T03 (b)	[wakeupFinished ∧ reqComMode = FullCom ∧ FrSMNumWakeupPatterns > 1 ∧ (wakeupTransmitted ∨ ¬ t1_IsActive) ∧ wakeupType = SingleChannelWakeup]	FE_START cancel t1 start t2 FE_ALLOW_COLDSTART

Tran-sition	Event [Condition]	Actions
T03 (c)	[wakeupFinished ∧ reqComMode = FullCom ∧ FrSMNumWakeupPatterns > 1 ∧ ¬ wakeupTransmitted ∧ wakeupType = SingleChannelWakeup]	FE_START start t2
T03 (d)	[wakeupFinished ∧ reqComMode = FullCom ∧ wakeupType = DualChannelWakeup]	FE_START start t2
T03 (e)	[wakeupFinished ∧ reqComMode = FullCom ∧ wakeupType = DualChannelWakeup-Forward]	FE_START FE_ALLOW_COLDSTART start t2
T04 (a)	t1 [reqComMode = FullCom ∧ vPOC!State ≠ Normal Active]	FE_ALLOW_COLDSTART
T04 (b)	[reqComMode = FullCom ∧ wakeupType = DualChannelWakeup ∧ *AllChannelsAwake* ∧ vPOC!State ≠ Normal Active]	FE_ALLOW_COLDSTART
T05	t2 [startupCounter ≤ FrSMStartupRepetitionsWithWakeup ∧ reqComMode = FullCom ∧ wakeupType ≠ NoWakeup ∧ vPOC!State ≠ Normal Active]	FE_TRCV_NORMAL FE_CONFIG FE_WAKEUP startupCounter := startupCounter + 1
T06	t2 [(FrSMStartupRepetitionsWithWakeup < startupCounter ∨ wakeupType = NoWakeup) ∧ startupCounter ≤ FrSMStartupRepetitions ∧ reqComMode = FullCom ∧ vPOC!State ≠ Normal Active]	FE_TRCV_NORMAL FE_CONFIG FE_START FE_ALLOW_COLDSTART startupCounter := startupCounter + 1 start t2
T08	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze ∧ vPOC!SlotMode = AllSlots ∧ reqComMode = FullCom]	cancel t1 cancel t2 FE_START_FRIF FE_DEM_STATUS_PASSED FE_DEM_SYNC_LOSS_PASSED FE_FULL_COM_IND cancel t3
T108	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze ∧ vPOC!SlotMode ≠ AllSlots ∧ reqComMode = FullCom]	cancel t1 cancel t2 FE_START_FRIF FE_DEM_STATUS_PASSED FE_DEM_SYNC_LOSS_PASSED cancel t3
T09a	FrSM_RequestComMode() [reqComMode = NoCom]	FE_STOP_FRIF FE_HALT FE_NO_COM_IND
T09b	FrSM_RequestComMode() [reqComMode = NoCom]	FE_STOP_FRIF FE_HALT
T10a	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom ∧ (*FrSmCheckWakeupReason* ∨ ¬ FrSMIsWakeupEcu)]	FE_DEM_SYNC_LOSS FE_STOP_FRIF FE_NO_COM_IND FE_CONFIG FE_START startupCounter := 1 start t2

Tran-sition	Event [Condition]	Actions
		start t3
T10b	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom ∧ (*FrSmCheckWakeupReason* ∨ ¬ FrSMIsWakeupEcu)]	FE_DEM_SYNC_LOSS FE_STOP_FRIF FE_CONFIG FE_START startupCounter := 1 start t2 start t3
T101	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze ∧ vPOC!SlotMode = AllSlots]	FE_FULL_COM_IND
T11a	t_TrcvStdbyDelay	FE_TRCV_STANDBY FE_CONFIG
T11b	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom]	cancel t_TrcvStdbyDelay FE_TRCV_STANDBY FE_CONFIG
T12	[reqComMode = NoCom]	cancel t1 cancel t2 cancel t3 FE_DEM_SYNC_LOSS_PASSED FE_TRCV_STANDBY FE_CONFIG
T13	[reqComMode = NoCom]	FE_DEM_SYNC_LOSS_PASSED FE_TRCV_STANDBY FE_CONFIG cancel t3 cancel t1
T14	FrSM_RequestComMode() [reqComMode = NoCom]	FE_DEM_SYNC_LOSS_PASSED FE_HALT cancel t3
T15	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze ∧ vPOC!SlotMode = AllSlots]	FE_DEM_SYNC_LOSS_PASSED FE_START_FRIF FE_FULL_COM_IND cancel t3
T115	[vPOC!State = Normal Active ∧ ¬ vPOC!Freeze ∧ vPOC!SlotMode ≠ AllSlots]	FE_DEM_SYNC_LOSS_PASSED FE_START_FRIF cancel t3
T16a	[vPOC!State = Normal Passive ∧ ¬ vPOC!Freeze]	FE_DEM_SYNC_LOSS FE_STOP_FRIF FE_NO_COM_IND start t3
T16b	[vPOC!State = Normal Passive ∧ ¬ vPOC!Freeze]	FE_DEM_SYNC_LOSS FE_STOP_FRIF start t3
T17	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom ∧ (*FrSmCheckWakeupReason* ∨ ¬ FrSMIsWakeupEcu)]	FE_CONFIG wakeupType := NoWakeup FE_START startupCounter := 1 start t2
T20a	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom ∧ ¬ *FrSmCheckWakeupReason* ∧ FrSMIsWakeupEcu]	wakeupType := SingleChannelWakeup FE_DEM_SYNC_LOSSFE_STOP_FRIF FE_NO_COM_IND FE_CONFIG FE_WAKEUP startupCounter := 1 start t1

Tran-sition	Event [Condition]	Actions
		start t3
T20b	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom ∧ ¬ *FrSmCheckWakeupReason* ∧ FrSMIsWakeupEcu	wakeupType := SingleChannelWakeup FE_DEM_SYNC_LOSSFE_STOP_FRIF FE_CONFIG FE_WAKEUP startupCounter := 1 start t1 start t3
T20c	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ reqComMode = FullCom ∧ ¬ *FrSmCheckWakeupReason* ∧ FrSMIsWakeupEcu]	wakeupType := SingleChannelWakeup FE_CONFIG FE_WAKEUP startupCounter := 1 start t1 start t3
T21	[(vPOC!State = Halt ∨ vPOC!Freeze) ∧ ¬ *t_TrcvStdbyDelay_IsActive]*	start t_TrcvStdbyDelay
T30	t3	FE_DEM_STATUS_FAILED FE_STARTUP_ERROR_IND
T31	[t3_IsNotActive]	FE_STARTUP_ERROR_IND
T32	[t3_IsNotActive]	FE_STARTUP_ERROR_IND
T33	[t3_IsNotActive]	FE_STARTUP_ERROR_IND
T40	[lowNumberOfColdstarters]	
T41	[¬ lowNumberOfColdstarters]	

Legend:

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

⌋ ()

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

Note: If resynchronization cannot be achieved before t2 expires (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 Production Errors

7.5.1 FRSM_E_CLUSTER_STARTUP

Error Name:	FRSM_E_CLUSTER_STARTUP
Short Description:	FlexRay cluster startup failure.
Long Description:	FlexRay controller has not reached the state *normal active* within the configured time after FlexRay startup.

Recommended DTC:	Assigned by DEM	
Detection Criteria:	Fail	FlexRay controller has not reached the state normal active within the time t3
	Pass	FlexRay controller has reached the state normal active
Secondary Parameters:	None	
Time Required:	FrSMDurationT3	
Monitor Frequency	Continuous	
MIL illumniation:	Assigned by DEM	

7.5.2 FRSM_E_CLUSTER_SYNC_LOSS

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

7.6 Error classification

Values for production code Event Ids are assigned in the configuration, see section 10.2.6.

Type of error	***Relevance***	***Related error code***	***Value [hex]***
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.	Development	FRSM_E_NULL_PTR	0x01
Invalid network handle parameter	Development	FRSM_E_INV_HANDLE	0x02
FrSM module was not initialized	Development	FRSM_E_UNINIT	0x03
Invalid communication mode requested	Development	FRSM_E_INV_MODE	0x04

()

7.7 Error detection

For details refer to the chapter 7.3 “Error Detection” in *SWS_BSWGeneral.*

7.8 Error notification

For details refer to the chapters 7.4 “Error notification” in *SWS_BSWGeneral.*

7.9 Debugging

[SWS_FrSM_00137] ⌈The states of FrSM state machine shall be available for debugging. ⌋(SRS_BSW_00442)

8 API specification

8.1 Imported types

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

[SWS_FrSM_00095] ⌈

Module	*Imported Type*
ComM	ComM_ModeType
ComStack_Types	NetworkHandleType
Dem	Dem_EventIdType
	Dem_EventStatusType
Fr	Fr_ChannelType
	Fr_POCStatusType
FrIf	FrIf_StateTransitionType
FrTrcv	FrTrcv_TrcvModeType
	FrTrcv_TrcvWUReasonType
Std_Types	Std_ReturnType
	Std_VersionInfoType

⌋ ()

8.2 Type definitions

8.2.1 FrSM_ConfigType

[SWS_FrSM_00198]⌈

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

⌋()

8.2.2 FrSM_BswM_StateType

[SWS_FrSM_00199]⌈

Name:	FrSM_BswM_StateType	
Type:	Enumeration	
Range:	FRSM_BSWM_READY	0
	FRSM_BSWM_READY_ECU_PASSIVE	1
	FRSM_BSWM_STARTUP	2
	FRSM_BSWM_STARTUP_ECU_PASSIVE	3
	FRSM_BSWM_WAKEUP	4
	FRSM_BSWM_WAKEUP_ECU_PASSIVE	5
	FRSM_BSWM_HALT_REQ	6
	FRSM_BSWM_HALT_REQ_ECU_PASSIVE	7

	FRSM_BSWM_KEYSLOT_ONLY	8
	FRSM_BSWM_KEYSLOT_ONLY_ECU_PASSIVE	9
	FRSM_BSWM_ONLINE	10
	FRSM_BSWM_ONLINE_ECU_PASSIVE	11
	FRSM_BSWM_ONLINE_PASSIVE	12
	FRSM_BSWM_ONLINE_PASSIVE_ECU_PASSIVE	13
	FRSM_LOW_NUMBER_OF_COLDSTARTERS	14
	FRSM_LOW_NUMBER_OF_COLDSTARTERS_ECU_PASSIVE	15
Description:	This type defines the states that are reported to the BswM using BswM_FrSM_CurrentState.	

⌋()

8.3 Function definitions

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

8.3.1 FrSM_Init

[SWS_FrSM_00013] ⌈

Service name:	FrSM_Init	

Syntax:	`void FrSM_Init(`
	`  const FrSM_ConfigType* FrSM_ConfigPtr`
	`)`	

Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	FrSM_ConfigPtr	Pointer to a selected configuration structure
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the FlexRay State Manager.	

⌋ (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)

[SWS_FrSM_00015] ⌈If development error detection is enabled (`FrSMDevErrorDetect` is ON) and FrSM_ConfigPtr equals NULL_PTR, the FrSM_Init function shall report the error FRSM_E_NULL_PTR to the DET and shall not perform the initialization. However, a value of NULL_PTR for FrSM_ConfigPtr shall not be treated as an error, if a configuration variant (see section 10.2.1) without post-build data is used.⌋ ()

8.3.2 FrSM_RequestComMode

[SWS_FrSM_00020] ⌈

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

⌋ (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] ⌈

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

⌋ (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_COM-MUNICATION.
- In any other case, the communication mode is COMM_NO_COMMUNI-CATION.

⌋ ()

[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_NULL_PTR 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] ⌈

Service name:	FrSM_GetVersionInfo	

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

Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	versioninfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	This service returns the version information of this module. The version information includes: - Module Id	

	- Vendor Id - Vendor specific version numbers (BSW00407). This function shall be pre compile time configurable On/Off by the configuration parameter: FRSM_VERSION_INFO_API Hint: If source code for caller and callee of this function is available this function should be realized as a macro. The macro should be defined in the modules header file.

」(SRS_BSW_00407)

8.3.5 FrSM_AllSlots

[SWS_FrSM_00172] ?

Service name:	FrSM_AllSlots	
Syntax:	`Std_ReturnType FrSM_AllSlots( NetworkHandleType NetworkHandle )`	
Service ID[hex]:	0x05	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant for different FlexRay clusters	
Parameters (in):	NetworkHandle	This parameter identifies the FlexRay cluster for which a communication mode is requested.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request not accepted
Description:	This API function can be used to leave the KeySlotOnlyMode.	

」()

[**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] ⌈

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

? ()

[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. Therefore, the header file FrSM_Cbk.h is not needed.

8.5 Scheduled functions

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

8.5.1 FrSM_MainFunction_<Cluster Id>

[SWS_FrSM_00118] ⌈

Service name:	FrSM_MainFunction_<Cluster Id>

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

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

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

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

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

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

[SWS_FrSM_00096] ⌈

API function	*Description*
BswM_FrSM_CurrentState	Function called by FrSM to indicate its current state.
ComM_BusSM_ModeIndication	Indication of the actual bus mode by the corresponding Bus State Manager. ComM shall propagate the indicated state to the users with means of the RTE and BswM.
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.
FrIf_AllowColdstart	Wraps the FlexRay Driver API function Fr_AllowColdstart().
FrIf_ClearTransceiverWakeup	Wraps the FlexRay Transceiver Driver API function FrTrcv_ClearTransceiverWakeup(). The enum value "FR_CHANNEL_AB" shall not be used.
FrIf_ControllerInit	Initialized a FlexRay CC.
FrIf_GetPOCStatus	Wraps the FlexRay Driver API function Fr_GetPOCStatus().
FrIf_GetTransceiverWUReason	Wraps the FlexRay Transceiver Driver API function FrTrcv_GetTransceiverWUReason(). The enum value "FR_CHANNEL_AB" shall not be used.
FrIf_HaltCommunication	Wraps the FlexRay Driver API function Fr_HaltCommunication().
FrIf_SendWUP	Wraps the FlexRay Driver API function Fr_SendWUP().
FrIf_SetState	Requests FrIf state machine transition.
FrIf_SetTransceiverMode	Wraps the FlexRay Transceiver Driver API function FrTrcv_SetTransceiverMode(). The enum value "FR_CHANNEL_AB" shall not be used.
FrIf_StartCommunication	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] ⌈

API function	*Description*
Det_ReportError	Service to report development errors.
FrIf_AllSlots	Wraps the FlexRay Driver API function Fr_AllSlots
FrIf_GetNumOfStartupFrames	Wraps the FlexRay Driver API function Fr_GetNumOfStartupFrames and gets a list of the the current number of startup frames seen on the cluster. See variable vStartupPairs of [12] for details.
FrIf_GetWakeupRxStatus	Wraps the FlexRay Driver API function Fr_GetWakeupRxStatus and gets the wakeup received information from the FlexRay controller.
FrIf_SetWakeupChannel	Wraps the FlexRay Driver API function Fr_SetWakeupChannel(). The enum value "FR_CHANNEL_AB" shall not be used.
FrNm_StartupError	This function is called by the FrSM when synchronization of the FlexRay cluster could not be achieved.

⌋ ()

8.6.3 Configurable Interfaces

8.6.3.1 <Cdd>_SyncLossErrorIndication

[SWS_FrSM_00190] ⌈

Service name:	<Cdd>_SyncLossErrorIndication	

Syntax:	`void <Cdd>_SyncLossErrorIndication(`
	`  NetworkHandleType NetworkHandle,`
	`  boolean SyncLossErrorStatus`
	`)`	

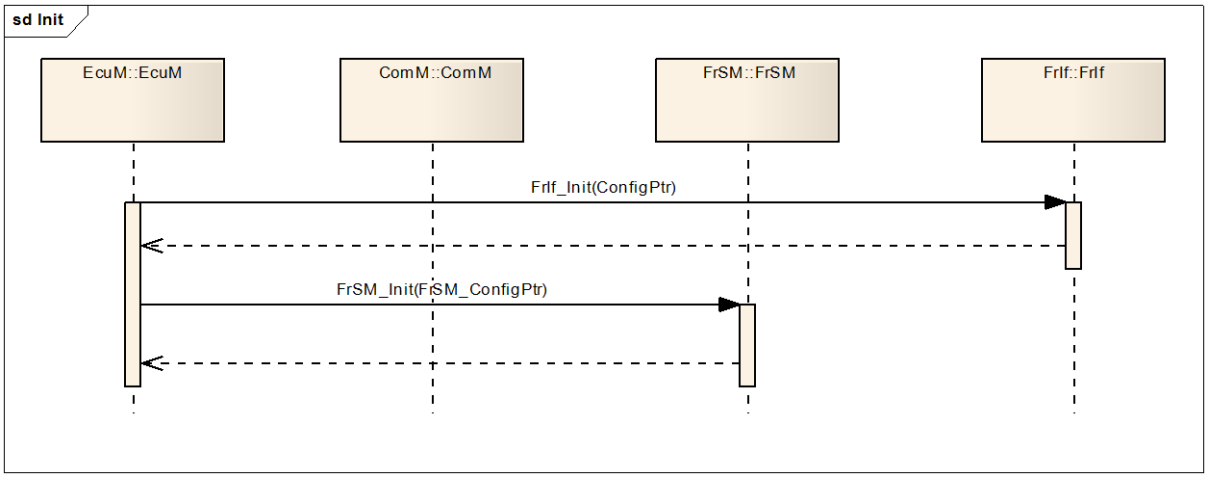
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different FlexRay clusters	
Parameters (in):	NetworkHandle	Handle of FlexRay cluster
	SyncLossErrorStatus	true: ECU lost synchronization to the FlexRay cluster. false: ECU can synchronize to the FlexRay cluster or request for full communication has been released after the ECU lost its synchronization to the FlexRay cluster.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This function is called with parameter SyncLossErrorStatus = true when the ECU loses its synchronization to the FlexRay cluster. The function is called with parameter 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.	

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 looses its synchronization to the FlexRay cluster, after it could synchronize to the FlexRay cluster or when the FullCom request is released after the ECU lost its synchronization to the FlexRay cluster.

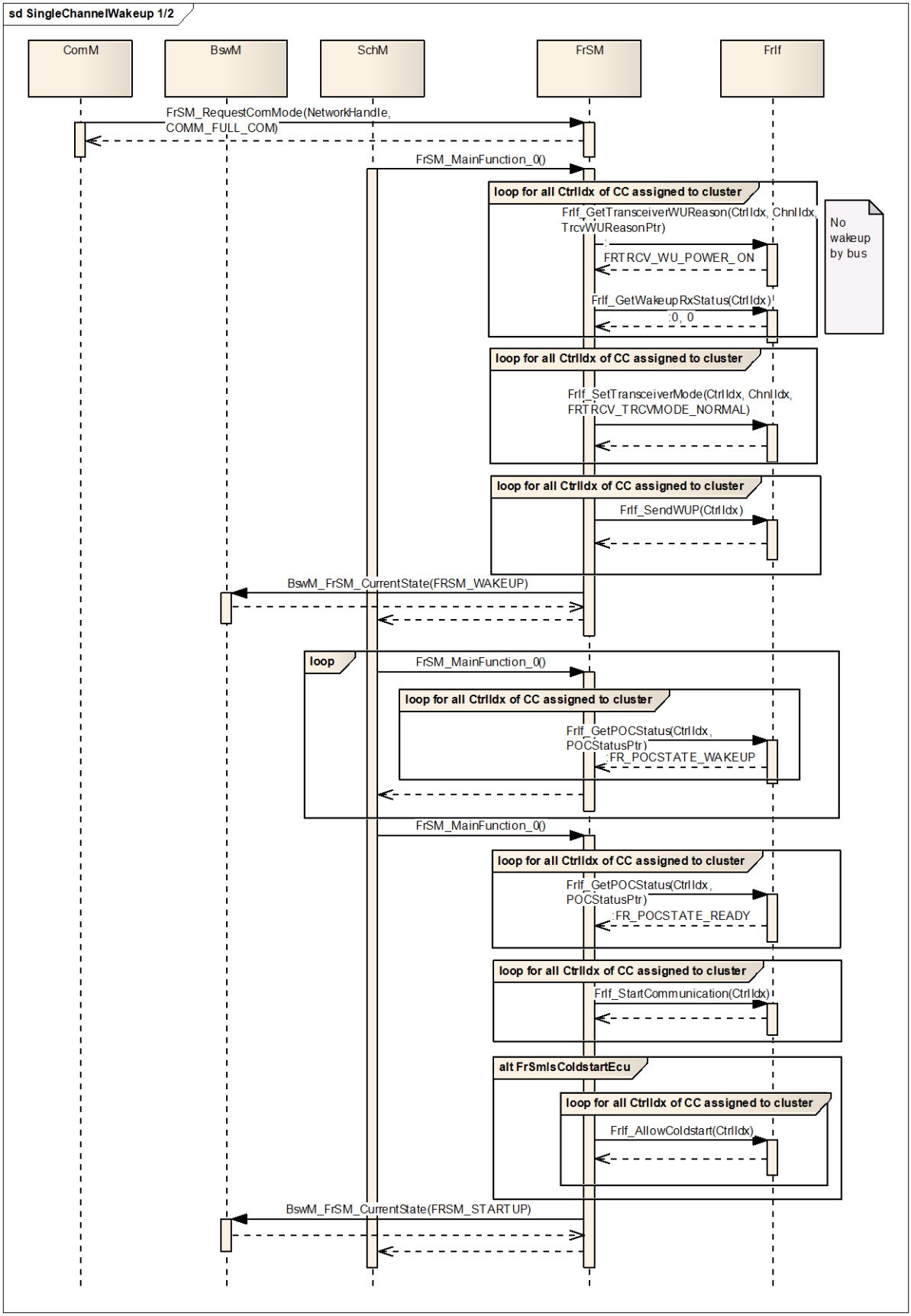
⌋ ()

9 Sequence diagrams

9.1 Initialization



9.2 Single Channel Wakeup



(continued on next page)

(continued)

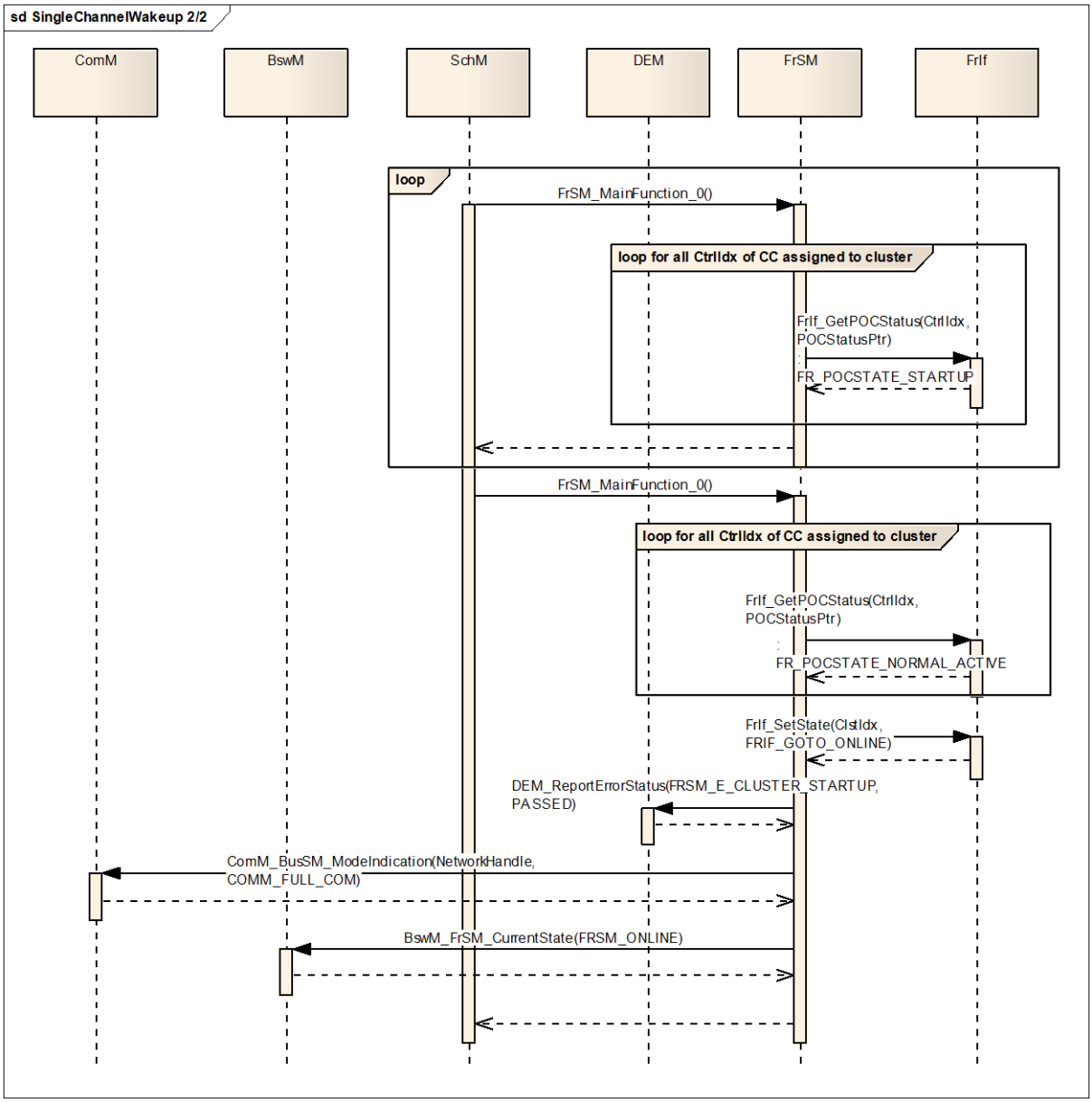
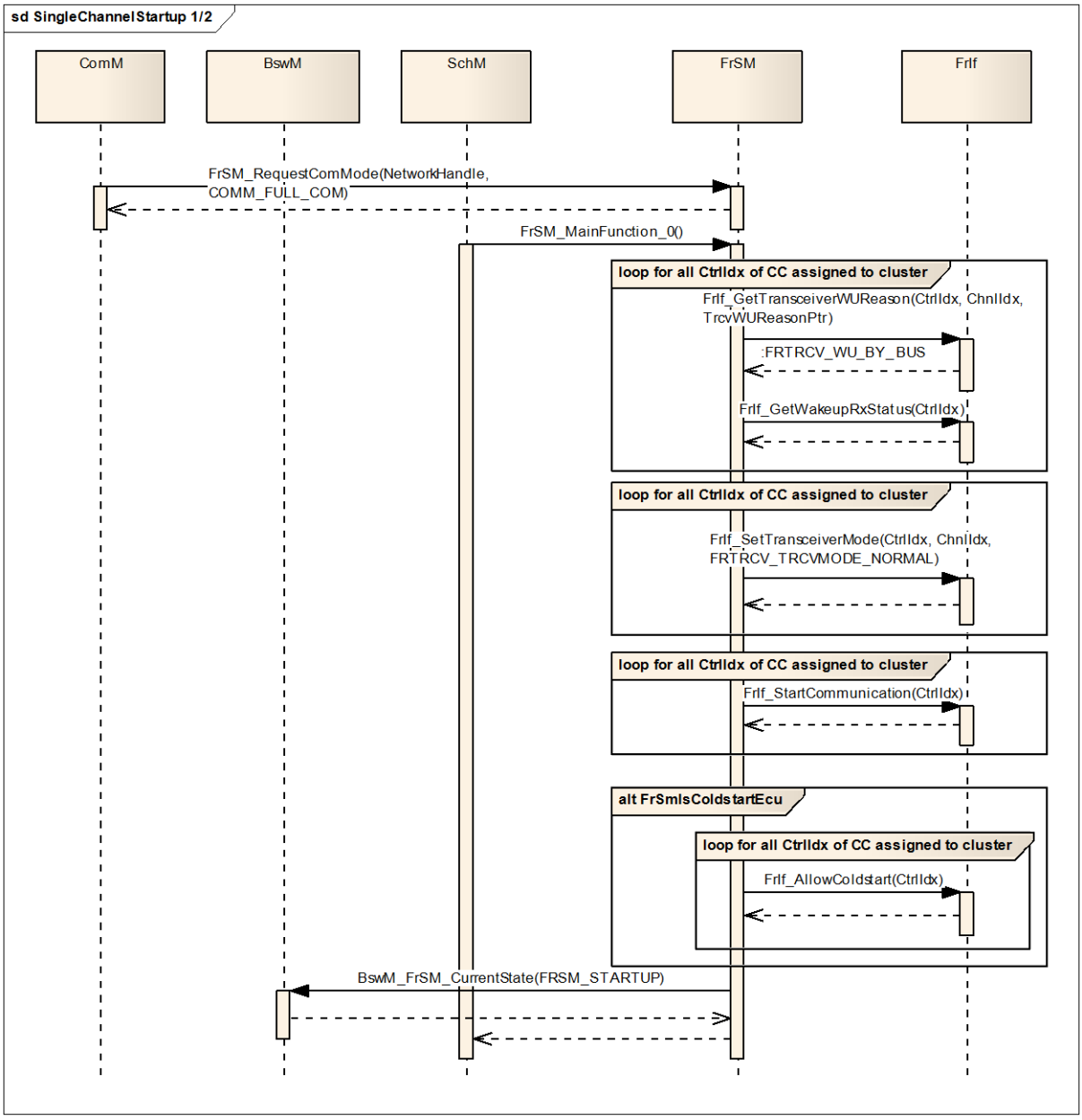


Figure 3 Transition from no communication to full communication for the case of an ECU that has a local wakeup reason.

9.3 Single Channel Passive Startup



continued on next page

(continued)

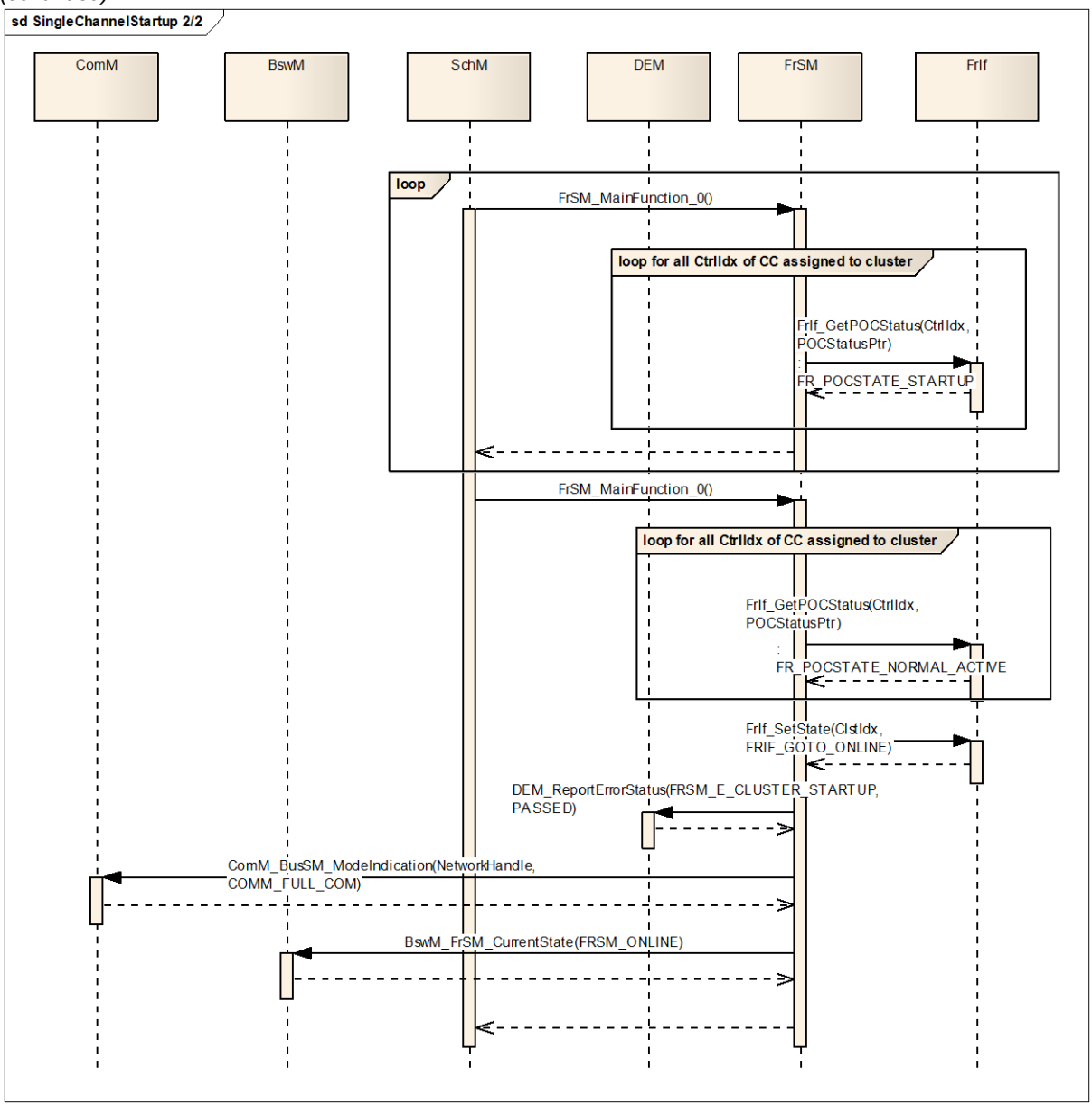
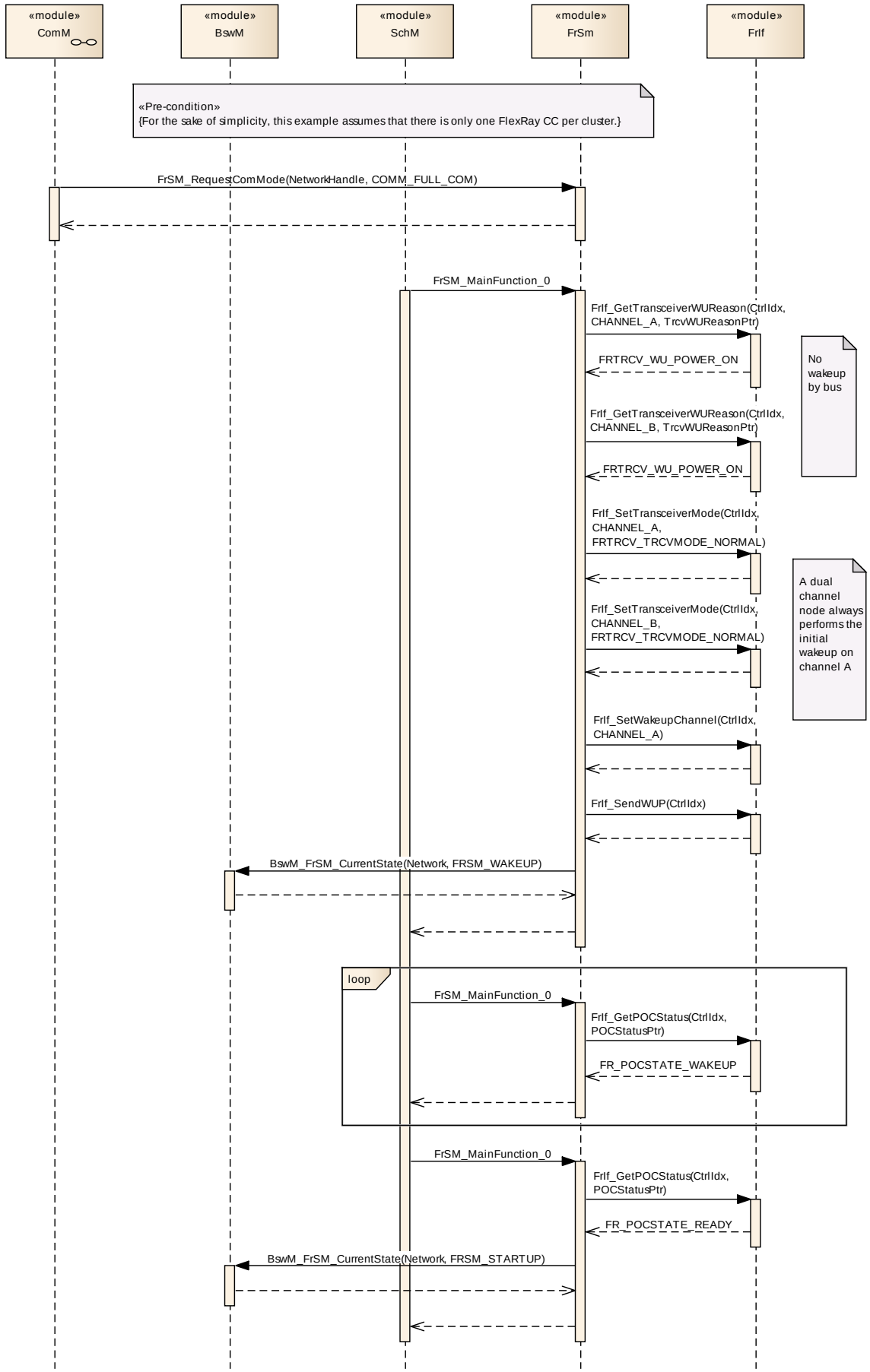


Figure 4 Transition from no communication to full communication for the case of an ECU that has been woken up by bus.

9.4 Dual Channel Wakeup


«module»
ComM
«module»
BswM
«module»
SchM
«module»
FrSm
«module»
Frlf
«Pre-condition»
{For the sake of simplicity, this example assumes that there is only one FlexRay CC per cluster.}
FrSM_RequestComMode(NetworkHandle, COMM_FULL_COM)
FrSM_MainFunction_0
Frlf_GetTransceiverWUReason(CtrlIdx, CHANNEL_A, TrcvWUReasonPtr)
FRTRCV_WU_POWER_ON
No wakeup by bus
Frlf_GetTransceiverWUReason(CtrlIdx, CHANNEL_B, TrcvWUReasonPtr)
FRTRCV_WU_POWER_ON
Frlf_SetTransceiverMode(CtrlIdx, CHANNEL_A, FRTRCV_TRCVMODE_NORMAL)
A dual channel node always performs the initial wakeup on channel A
Frlf_SetTransceiverMode(CtrlIdx, CHANNEL_B, FRTRCV_TRCVMODE_NORMAL)
Frlf_SetWakeupChannel(CtrlIdx, CHANNEL_A)
Frlf_SendWUP(CtrlIdx)
BswM_FrSM_CurrentState(Network, FRSM_WAKEUP)
loop
FrSM_MainFunction_0
Frlf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_WAKEUP
FrSM_MainFunction_0
Frlf_GetPOCStatus(CtrlIdx, POCStatusPtr)
FR_POCSTATE_READY
BswM_FrSM_CurrentState(Network, FRSM_STARTUP)

(continued on next page)

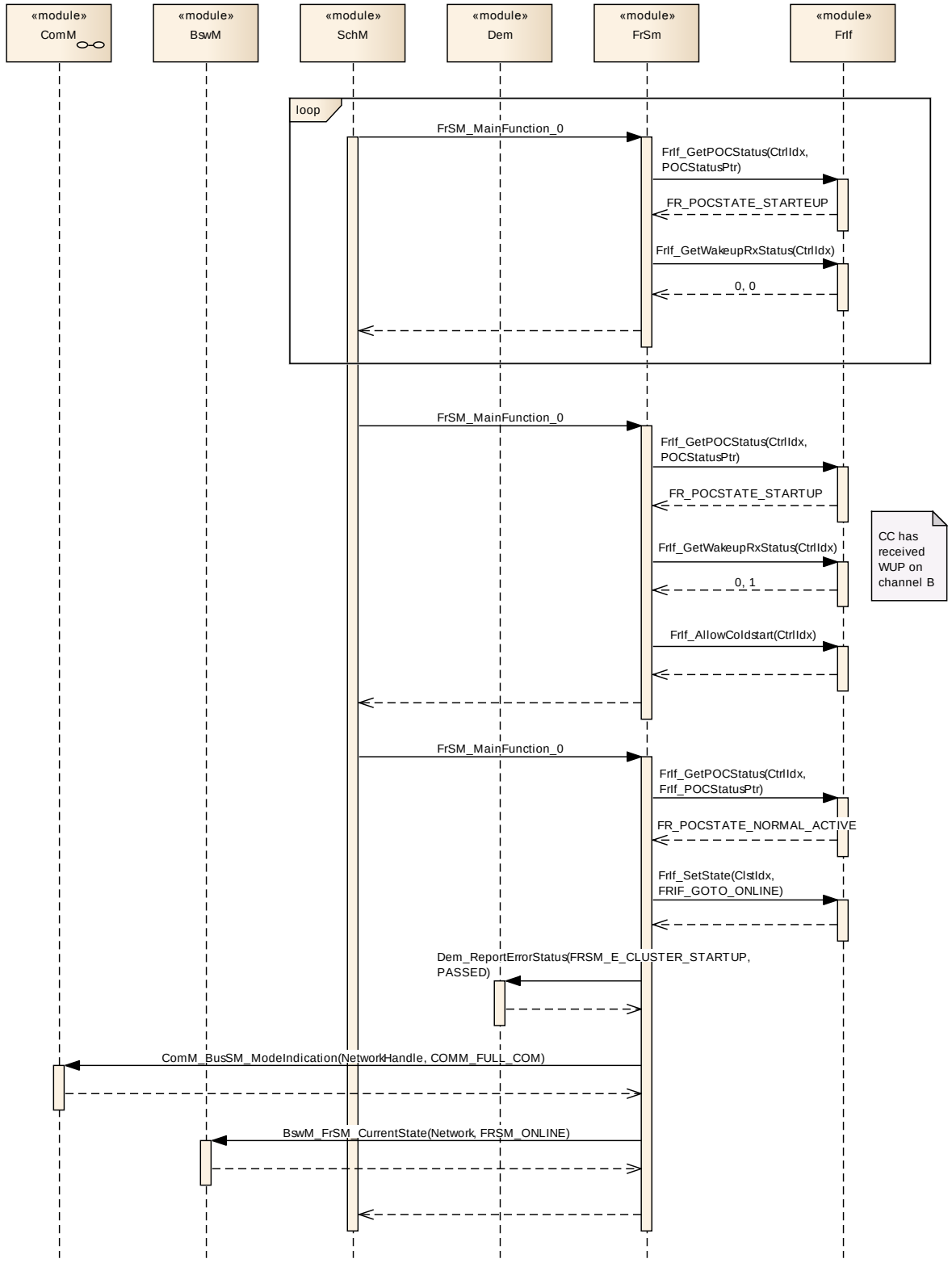
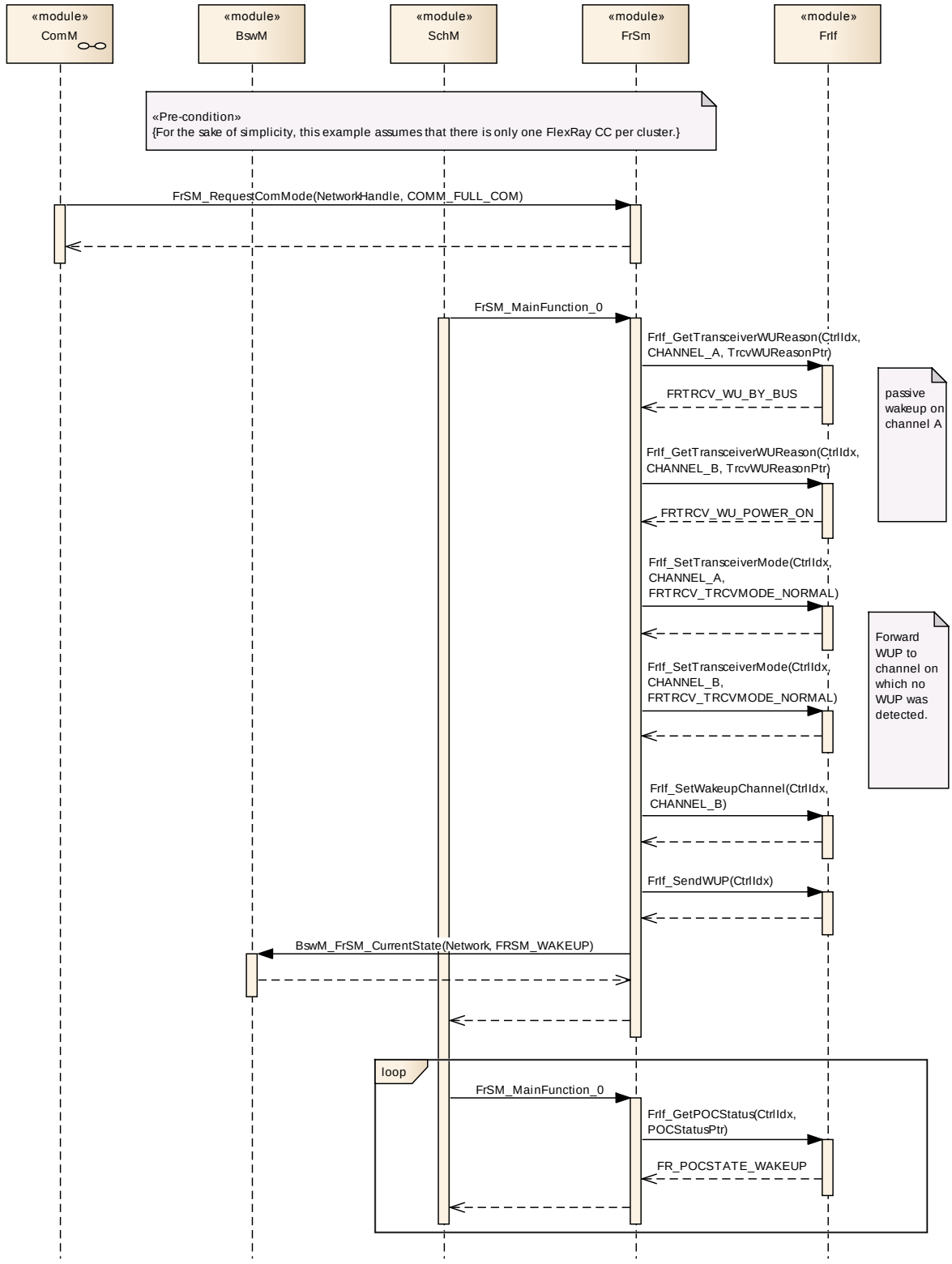


Figure 5 Transition from no communication to full communication for the case of a dual channel ECU with a local wakeup reason.

9.5 Dual Channel Wakeup Forward



(continued on next page)

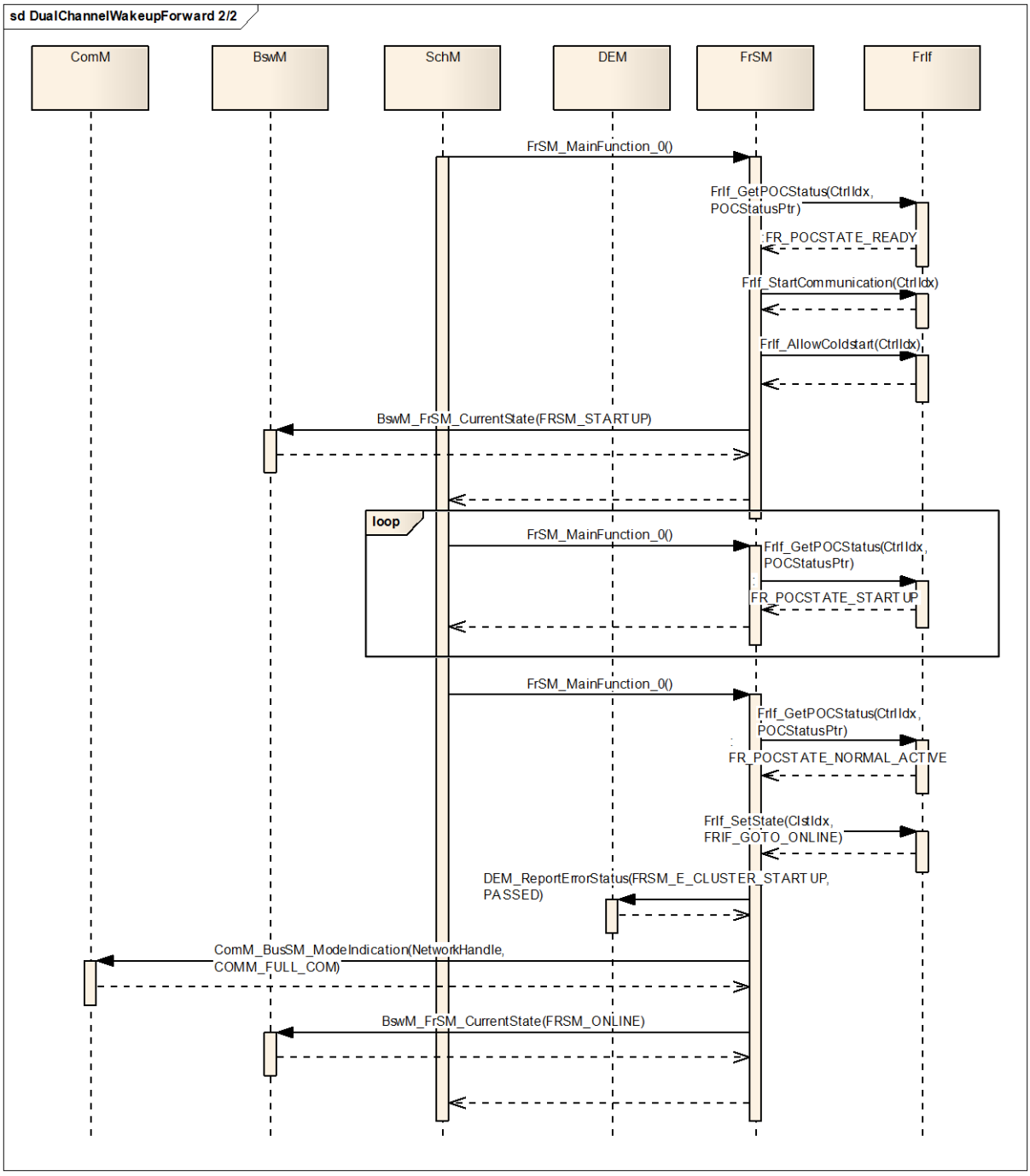
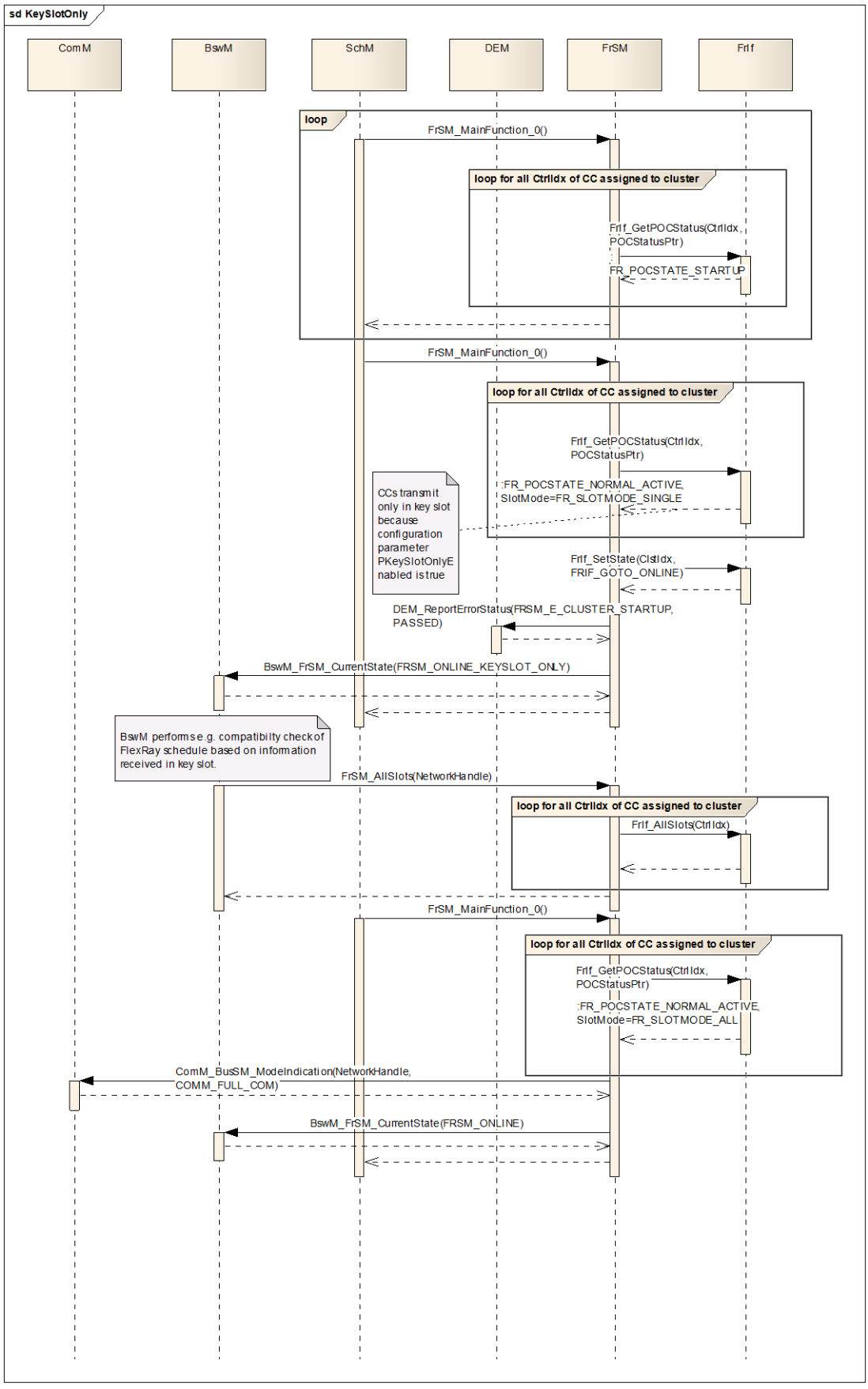
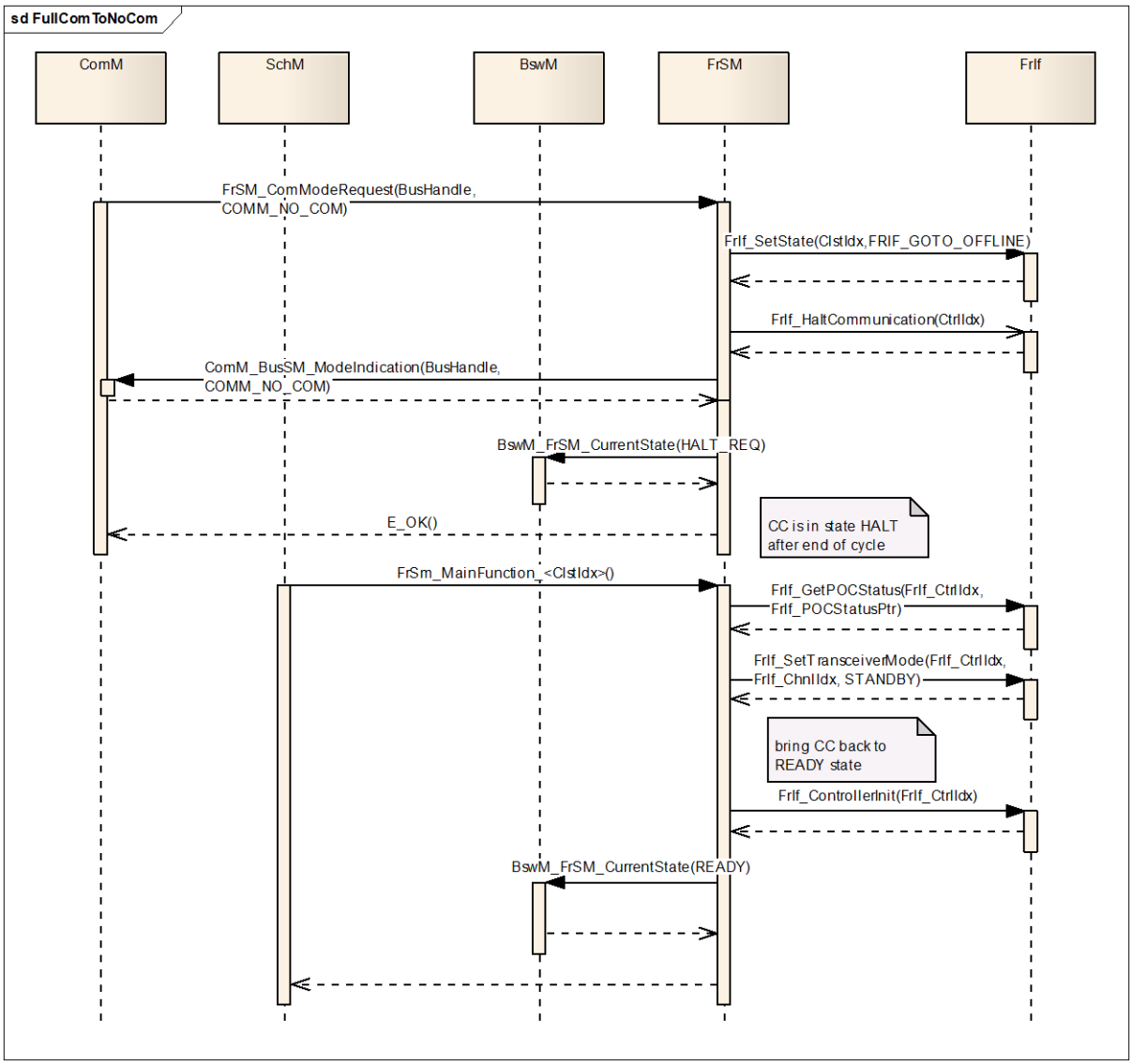


Figure 6 Transition from no communication to full communication for the case of a dual channel that has been woken up by bus.

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 7 and Chapter 8.

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

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

10.2.1 Variants

10.2.1.1 VARIANT-PRE-COMPILE (Pre-compile Configuration)

[SWS_FrSM_00098] ⌈In the variant VARIANT-PRE-COMPILE all parameters below that are marked as pre-compile configurable with “VARIANT-PRE-COMPILE“ shall be configurable in a pre-compile manner, for example as #defines.

The module is most likely delivered as source code. ⌋ ()

10.2.1.2 VARIANT-LINK-TIME (Link-time Configuration)

[SWS_FrSM_00099] ⌈The variant VARIANT-LINK-TIME shall include all configuration options of the variant VARIANT-PRE-COMPILE. Additionally all parameters that are marked as link-time configurable with “VARIANT-LINK-TIME“

shall be configurable at link time for example by linking a special configured parameter object file.

The module is most likely delivered as object code. ⌋(SRS_BSW_00342)

10.2.1.3 VARIANT-POST-BUILD (Post-build Configuration)

[SWS_FrSM_00100] ⌈The variant VARIANT-POST-BUILD shall include all configuration options of the variant VARIANT-LINK-TIME. Additionally all parameters that are marked as post-build configurable with “VARIANT-POST-BUILD“ shall be configurable post build for example by flashing configuration data.

The module is most likely delivered as object code. ⌋(SRS_BSW_00342)

10.2.2 FrSM

Module Name	*FrSM*
Module Description	Configuration of the FlexRay State Manager

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.

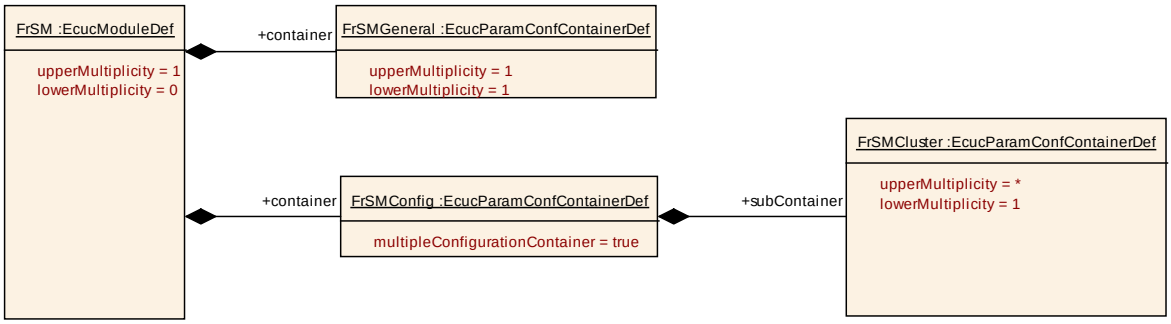


Figure 7 FlexRay State Manager Configuration

10.2.3 FrSMConfig

SWS Item	**ECUC_FrSM_00146 :**
Container Name	FrSMConfig{FRSM_CONFIG} [Multi Config Container]
Description	This container comprises the cluster specific configuration of the FlexRay State Manager.
Configuration Parameters	

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

10.2.4 FrSMGeneral

SWS Item	**ECUC_FrSM_00107 :**
Container Name	FrSMGeneral{FRSM_GENERAL}
Description	This container contains the general configuration parameters of the FlexRay State Manager.
Configuration Parameters	

SWS Item	**ECUC_FrSM_00172 :**		
Name	FrSMAllSlotsSupport {FRSM_ALL_SLOTS_SUPPORT}		
Description	Configuration parameter to enable/disable FrSM support to enable/disable the switching from key-slot/single-slot mode to all-slot mode.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00171 :**		
Name	FrSMCddHeaderFile {FRSM_CDD_HEADER_FILE}		
Description	This parameter defines header files for callback functions which are implemented by CDD, e.g. <Cdd>_SyncLossErrorIndication.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00066 :**		
Name	FrSMDevErrorDetect {FRSM_DEV_ERROR_DETECT}		
Description	Enables and disables the development error detection and notification mechanism.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00167 :**
Name	FrSMSyncLossErrorIndicationName {FRSM_SYNC_LOSS_ERROR_INDICATION_NAME}
Description	Name of <Cdd>_SyncLossErrorIndication function that shall be called on loss of synchronization. If this parameter is omitted no indication shall take place.

Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00108 :**		
Name	FrSMVersionInfoApi {FRSM_VERSION_INFO_API}		
Description	Enables and disables the version info API		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

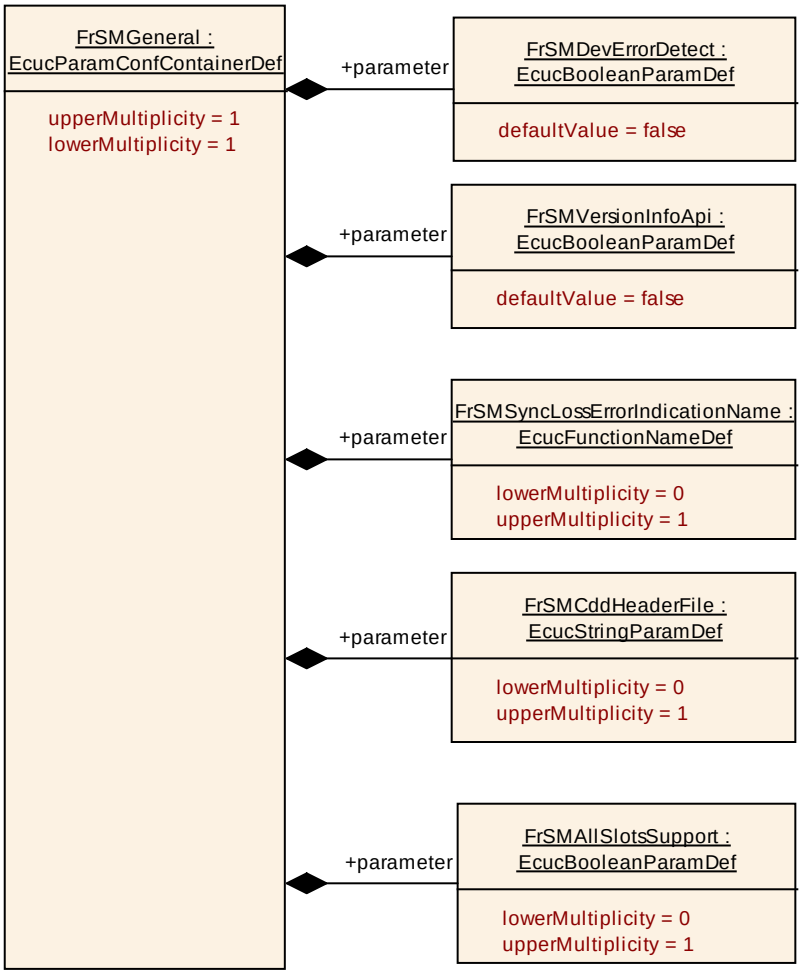


Figure 8 FrSMGeneral Container

10.2.5 FrSMCluster

SWS Item	**ECUC_FrSM_00067 :**
Container Name	FrSMCluster{FRSM_CLUSTER}
Description	This container specifies a FlexRay cluster and all related data. A FlexRay cluster may consist of more than one controller per ECU.
Configuration Parameters	

SWS Item	**ECUC_FrSM_00001 :**		
Name	FrSMCheckWakeupReason {FRSM_CHECK_WAKEUP_REASON}		
Description	If FrSMCheckWakeupReason is true, the FrSM will check the wakeup reason in order to skip the wakeup in case of wakeup by bus. If FrSMCheckWakeupReason is false, the FrSM will always try to perform a wakeup.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00166 :**		
Name	FrSMDelayStartupWithoutWakeup {FRSM_DELAY_STARTUP_WITHOUT_WAKEUP}		
Description	If true, timer t1 shall be started instead of immediately calling FrIf_AllowColdstart in case of a startup without wakeup.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00102 :**		
Name	FrSMDurationT1 {FRSM_DURATION_T1}		
Description	The duration of timer t1 in seconds. A value of 0 shall imply that the timer is not used.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSMMainFunctionCycleTime (As timers are checked during the call of FrSM_MainFunction, the effective timer duration will always be a multiple of FrSMMainFunctionCycleTime).		

SWS Item	**ECUC_FrSM_00089 :**
Name	FrSMDurationT2 {FRSM_DURATION_T2}

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

SWS Item	**ECUC_FrSM_00162 :**		
Name	FrSMDurationT3 {FRSM_DURATION_T3}		
Description	The duration of timer t3 in seconds. The value of this parameter shall be larger than the value of FrSMDurationT1 parameter. A value of 0 shall imply that the timer is not used. It shall only be possible to configure a value 0 if no FrNm is used.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSMMainFunctionCycleTime (As timers are checked during the call of FrSM_MainFunction, the effective timer duration will always be a multiple of FrSMMainFunctionCycleTime).		

SWS Item	**ECUC_FrSM_00068 :**		
Name	FrSMIsColdstartEcu {FRSM_IS_COLDSTART_ECU}		
Description	True: The ECU is a coldstart node for this FlexRay cluster. False: The ECU is no coldstart node for this FlexRay cluster.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00109 :**		
Name	FrSMIsWakeupEcu {FRSM_IS_WAKEUP_ECU}		
Description	True: FrSM shall perform a wakeup for this cluster. False: FrSM shall never perform a wakeup for this FlexRay cluster.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00115 :**		
Name	FrSMMainFunctionCycleTime {FRSM_MAIN_FUNCTION_CYCLE_TIME}		
Description	This parameter defines the cycle time in seconds of the periodic calling of FrSM main function.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME, VARIANT-POST-BUILD
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00168 :**		
Name	FrSMMinNumberOfColdstarter {FRSM_MIN_NUMBER_OF_COLDSTARTER}		
Description	This parameter defines the number of coldstarter that should not be underrun. If this parameter is not configured the mainfunction shall not check the number of startup frames.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00165 :**		
Name	FrSMNumWakeupPatterns {FRSM_NUM_WAKEUP_PATTERNS}		
Description	Maximum number of Wakeup Patterns the node may send before going to FRSM_STARTUP.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_FrSM_00069 :**		
Name	FrSMStartupRepetitions {FRSM_STARTUP_REPETITIONS}		
Description	The number of times an ECU may repeat the startup procedure for a FlexRay cluster.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: This value must be greater or equal to FrSMStartupRepetitionsWithWakeup		

SWS Item	ECUC_FrSM_00094 :		
Name	FrSMStartupRepetitionsWithWakeup {FRSM_STARTUP_REPETITIONS_WITH_WAKEUP}		
Description	The number of times an ECU may repeat the startup procedure including a wakeup for a FlexRay cluster.		
Multiplicity	0..1		
Type	EcucIntegerParamDef		
Range	0 .. 65535		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_FrSM_00170 :		
Name	FrSMTrcvStdbyDelay {FRSM_TRCV_STDBY_DELAY}		
Description	The duration of timer t_TrcvStdbyDelay in seconds. The granularity of this parameter shall be restricted to full FlexRay cycles (FrIfGdCycle). A value of 0 shall imply that the timer is not used.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. INF		
Default value	--		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: FrSmMainFunctionCycleTime		

SWS Item	ECUC_FrSM_00070 :		
Name	FrSMComMNetworkHandleRef {FRSM_COMM_NETWORK_HANDLE_REF}		
Description	Reference to the unique handle to identify one certain FlexRay network correspond to one of the network handles of the ComM configuration.		
Multiplicity	1		
Type	Symbolic name reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	ECUC_FrSM_00116 :		
Name	FrSMFrIfClusterRef {FRSM_FRIF_CLUSTER_REF}		
Description	References the cluster configuration in the FlexRay Interface configuration. Note that the assigned controllers and transceivers are defined in the FrIf configuration and can be accessed via this reference.		
Multiplicity	1		
Type	Symbolic name reference to [FrIfCluster]		
ConfigurationClass	Pre-compile time	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	Multiplicity	Scope / Dependency

FrSMClusterDemEventParameterRefs	0..1	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in this container and can be extended by vendor specific error references.

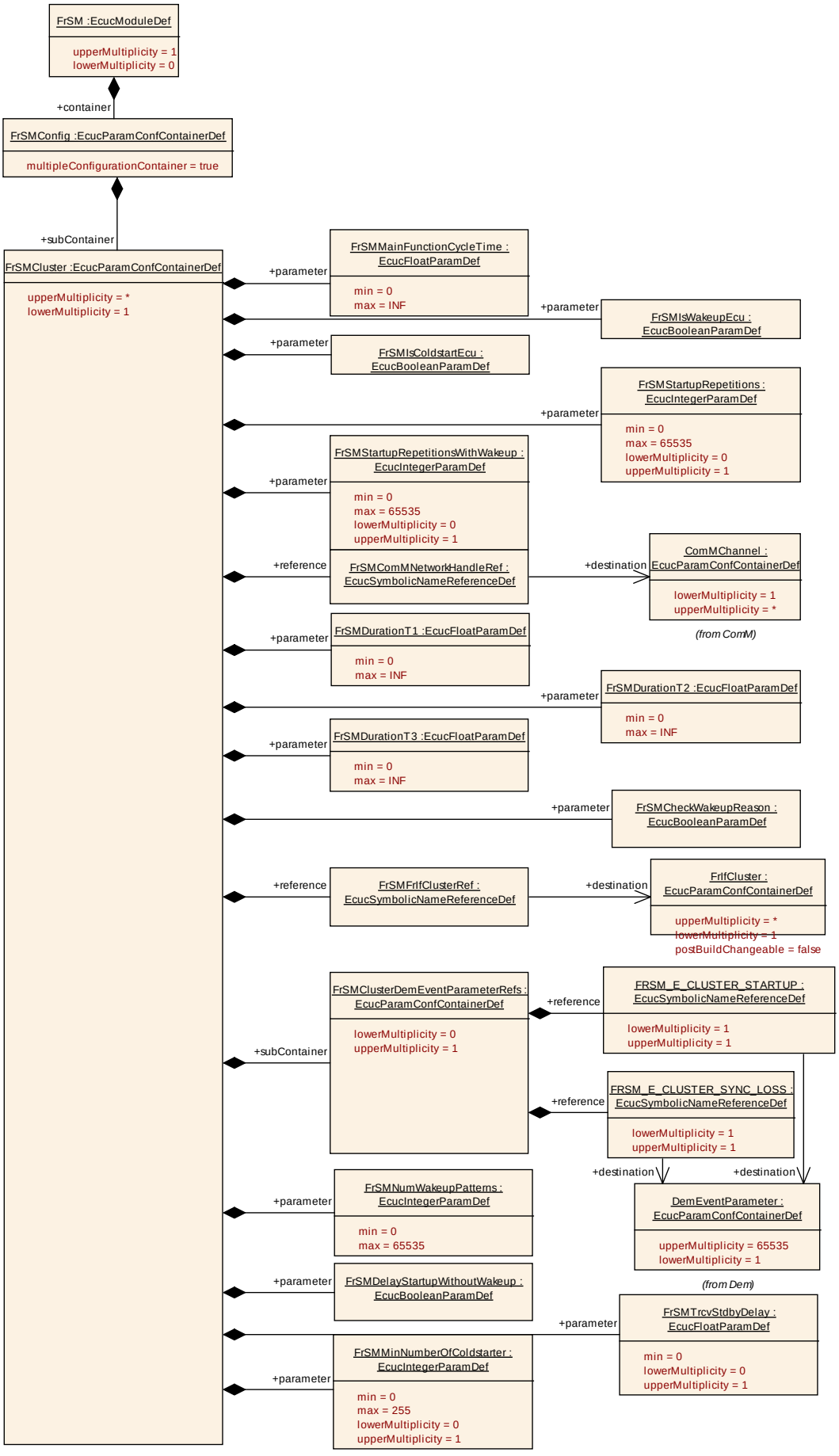

FrSM :EcucModuleDef
upperMultiplicity = 1
lowerMultiplicity = 0
+container
FrSMConfig :EcucParamConfContainerDef
multipleConfigurationContainer = true
+subContainer
FrSMCluster :EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
+parameter
FrSMMainFunctionCycleTime : EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMIsWakeupEcu : EcucBooleanParamDef
+parameter
FrSMIsColdstartEcu : EcucBooleanParamDef
+parameter
FrSMStartupRepetitions : EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrSMStartupRepetitionsWithWakeup : EcucIntegerParamDef
min = 0
max = 65535
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
FrSMComMNetworkHandleRef : EcucSymbolicNameReferenceDef
+destination
ComMChannel : EcucParamConfContainerDef
lowerMultiplicity = 1
upperMultiplicity = *
(from ComM)
+parameter
FrSMDurationT1 :EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMDurationT2 :EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMDurationT3 :EcucFloatParamDef
min = 0
max = INF
+parameter
FrSMCheckWakeupReason : EcucBooleanParamDef
+reference
FrSMFrIfClusterRef : EcucSymbolicNameReferenceDef
+destination
FrIfCluster : EcucParamConfContainerDef
upperMultiplicity = *
lowerMultiplicity = 1
postBuildChangeable = false
FrSMClusterDemEventParameterRefs : EcucParamConfContainerDef
lowerMultiplicity = 0
upperMultiplicity = 1
+reference
FRSM_E_CLUSTER_STARTUP : EcucSymbolicNameReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
+subContainer
+reference
FRSM_E_CLUSTER_SYNC_LOSS : EcucSymbolicNameReferenceDef
lowerMultiplicity = 1
upperMultiplicity = 1
+destination
+destination
DemEventParameter : EcucParamConfContainerDef
upperMultiplicity = 65535
lowerMultiplicity = 1
(from Dem)
+parameter
FrSMNumWakeupPatterns : EcucIntegerParamDef
min = 0
max = 65535
+parameter
FrSMDelayStartupWithoutWakeup : EcucBooleanParamDef
+parameter
FrSMTrcvStdbyDelay : EcucFloatParamDef
min = 0
lowerMultiplicity = 0
upperMultiplicity = 1
+parameter
FrSMMinNumberOfColdstarter : EcucIntegerParamDef
min = 0
max = 255
lowerMultiplicity = 0
upperMultiplicity = 1

Figure 9 FrSMCluster Container

10.2.6 FrSMClusterDemEventParameterRefs

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

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

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

No Included Containers

10.3 Published Information

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

11 Not applicable requirements

[SWS_FrSM_00186] ⌈ These requirements are not applicable to this specification. ⌋

(SRS_BSW_00170, SRS_BSW_00419, SRS_BSW_00387, 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_00326, SRS_BSW_00413, SRS_BSW_00347, SRS_BSW_00314, SRS_BSW_00370, SRS_BSW_00439, SRS_BSW_00449, SRS_BSW_00377, SRS_BSW_00359, SRS_BSW_00360, SRS_BSW_00440, BSW00443, BSW00444, BSW00446)