

<b>Document Title</b>	Specification of CAN State Manager
<b>Document Owner</b>	AUTOSAR
<b>Document Responsibility</b>	AUTOSAR
<b>Document Identification No</b>	253
<b>Document Status</b>	published
<b>Part of AUTOSAR Standard</b>	Classic Platform
<b>Part of Standard Release</b>	R19-11

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2019-11-28	R19-11	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Fixed Change_Baudrate-Statemachine for NoCom</li> <li>• Added GetPduMode-Interface to list.</li> <li>• Inconsistent behavior due to REPEAT_MAX / No Never-Give-Up Strategy fixed</li> <li>• Changed Document Status from Final to published</li> </ul>
2018-10-31	4.4.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Reclassification of some errors</li> <li>• Editorial changes</li> </ul>
2017-12-08	4.3.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Moved CANSM_E_MODE_REQUEST_TIM EOUT to Runtime Error</li> </ul>
2016-11-30	4.3.0	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Provide Delnit-API</li> <li>• ECU passive mode clarified and fixed</li> <li>• Editorial changes</li> </ul>
2015-07-31	4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Development Error Tracer replaced with Default Error Tracer</li> <li>• Bus-off recovery time dependencies specified more precisely</li> <li>• Optional interface to check and to change baudrate removed</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2014-10-31	4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• API for ECU passive mode activation</li> <li>• Baudrate change without reinitialisation, if possible</li> <li>• Interface handling to CanIf module improved</li> <li>• Interface handling to ComM module improved</li> </ul>
2014-03-31	4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Introduction of random delays</li> <li>• Re-Request of ComMode</li> <li>• Add WakeupValidation to avoid race conditions</li> <li>• Adapt Bus Off Recovery and NM state synchronization</li> </ul>
2013-10-31	4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> <li>• Dependency to DCM module removed</li> <li>• Mileading timing row removed in CanSM_MainFunction</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>• Support Pretended Networking mode handling</li> <li>• Changed concept to setup baudrate</li> <li>• Initialization Sequence between ComM and CanSM</li> <li>• Do not send WUF as First Message on the Bus after BusOff</li> <li>• CanSm_TxTimeoutExeption in case of BusOff</li> </ul>

<b>Document Change History</b>			
<b>Date</b>	<b>Release</b>	<b>Changed by</b>	<b>Change Description</b>
2011-12-22	4.0.3	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Added new handling to support partial networking</li> <li>• Changed handling for bus deinitialisation according to AR3.x behaviour</li> <li>• New API and handling to change the baudrate of a CAN network</li> <li>• Changed handling for bus-off recovery and related production error report</li> <li>• Comprehensive revision of all state machine diagrams and SWS-ID-items</li> <li>• Changed classification of production errors and development errors</li> <li>• Solve conflicts of SWS-ID items with the conformance test specification</li> </ul>
2009-12-18	4.0.1	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• Configurable Bus-Off recovery with CAN TX confirmation instead of time based recovery</li> <li>• Control of PDU channel modes completely shifted from CanIf to CanSM module</li> </ul>
2010-02-02	3.1.4	AUTOSAR Administration	<ul style="list-style-type: none"> <li>• VMM/AMM Concept related changes (PDU group control shifted to BswM)</li> <li>• Asynchronous handling of CAN network mode transitions (consideration of CAN Transceiver and CAN controller mode notifications)</li> <li>• Solution of Document Improvement issues reported by TO (e. g. split up of non atomic software requirements, textual requirements instead of only a state diagram)</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>

## Document Change History

Date	Release	Changed by	Change Description
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 .....	9
2	Acronyms and abbreviations .....	10
3	Related documentation.....	11
3.1	Input documents.....	11
3.2	Related standards and norms .....	12
3.3	Related specification .....	12
4	Constraints and assumptions .....	14
4.1	Limitations .....	14
4.2	Applicability to car domains.....	14
5	Dependencies to other modules.....	15
5.1	ECU State Manager (EcuM).....	15
5.2	BSW Scheduler (SchM) .....	15
5.3	Communication Manager (ComM) .....	16
5.4	CAN Interface (CanIf).....	16
5.5	Diagnostic Event Manager (DEM).....	16
5.6	Basic Software Mode Manager (BswM) .....	16
5.7	CAN Network Management (CanNm) .....	16
5.8	Default Error Tracer (DET).....	16
5.9	File structure .....	16
5.9.1	Code file structure.....	16
5.9.2	Header file structure.....	17
5.9.3	Version check.....	17
6	Requirements traceability .....	18
7	Functional specification .....	24
7.1	General requirements.....	25
7.2	State machine for each CAN network .....	32
7.2.1	Trigger: PowerOn.....	33
7.2.2	Trigger: CanSM_Init.....	34
7.2.3	Trigger: CanSM_DeInit .....	35
7.2.4	Trigger: T_START_WAKEUP_SOURCE .....	35
7.2.5	Trigger: T_STOP_WAKEUP_SOURCE .....	37
7.2.6	Trigger: T_FULL_COM_MODE_REQUEST .....	38
7.2.7	Trigger: T_SILENT_COM_MODE_REQUEST .....	38
7.2.8	Trigger: T_NO_COM_MODE_REQUEST.....	40
7.2.9	Trigger: T_BUS_OFF .....	40
7.2.10	Guarding condition: G_FULL_COM_MODE_REQUESTED .....	42
7.2.11	Guarding condition: G_SILENT_COM_MODE_REQUESTED .....	43
7.2.12	Effect: E_PRE_NOCOM .....	45
7.2.13	Effect: E_NOCOM.....	46
7.2.14	Effect: E_FULL_COM .....	49
7.2.15	Effect: E_FULL_TO_SILENT_COM .....	53
7.2.16	Effect: E_BR_END_FULL_COM.....	57
7.2.17	Effect: E_BR_END_SILENT_COM .....	58

7.2.18	Effect: E_SILENT_TO_FULL_COM .....	59
7.2.19	Sub state machine CANSM_BSM_WUVALIDATION.....	60
7.2.20	Sub state machine: CANSM_BSM_S_PRE_NOCOM .....	63
7.2.21	Sub state machine: CANSM_BSM_S_SILENTCOM_BOR.....	140
7.2.22	Sub state machine: CANSM_BSM_S_PRE_FULLCOM .....	142
7.2.23	Sub state machine CANSM_BSM_S_FULLCOM .....	156
7.2.24	Sub state machine: CANSM_BSM_S_CHANGE_BAUDRATE.....	165
7.3	Error classification .....	168
7.3.1	Development Errors .....	168
7.3.2	Runtime Errors .....	169
7.3.3	Transient Faults .....	169
7.3.4	Production Errors .....	169
7.3.5	Extended Production Errors .....	169
7.4	Pretended Networking function .....	170
7.4.1	Activation .....	170
7.4.2	Deactivation .....	170
7.5	ECU online active / passive mode.....	170
7.6	Error detection.....	171
7.7	Error notification .....	171
7.8	Non-functional design rules.....	171
8	API specification.....	172
8.1	Imported types.....	172
8.2	Type definitions .....	173
8.2.1	CanSM_StateType .....	173
8.2.2	CanSM_ConfigType .....	173
8.2.3	CanSM_BswMCurrentStateType .....	173
8.3	Function definitions .....	175
8.3.1	CanSM_Init .....	175
8.3.2	CanSM_Delnit.....	175
8.3.3	CanSM_RequestComMode .....	176
8.3.4	CanSM_GetCurrentComMode .....	177
8.3.5	CanSM_StartWakeupSource .....	179
8.3.6	CanSM_StopWakeupSource .....	180
8.3.7	Optional.....	182
8.3.8	Call-back notifications .....	186
8.3.9	CanSM_ControllerBusOff.....	186
8.3.10	CanSM_ControllerModeIndication .....	187
8.3.11	CanSM_TransceiverModeIndication .....	188
8.3.12	CanSM_TxTimeoutException.....	188
8.3.13	CanSM_ClearTrcvWufFlagIndication .....	189
8.3.14	CanSM_CheckTransceiverWakeFlagIndication .....	190
8.3.15	CanSM_ConfirmPnAvailability .....	191
8.3.16	CanSM_CurrentlcomConfiguration .....	192
8.4	Scheduled functions.....	193
8.4.1	CanSM_MainFunction.....	193
8.5	Expected Interfaces.....	193
8.5.1	Mandatory Interfaces .....	193
8.5.2	Optional Interfaces .....	194
8.5.3	Configurable Interfaces .....	195

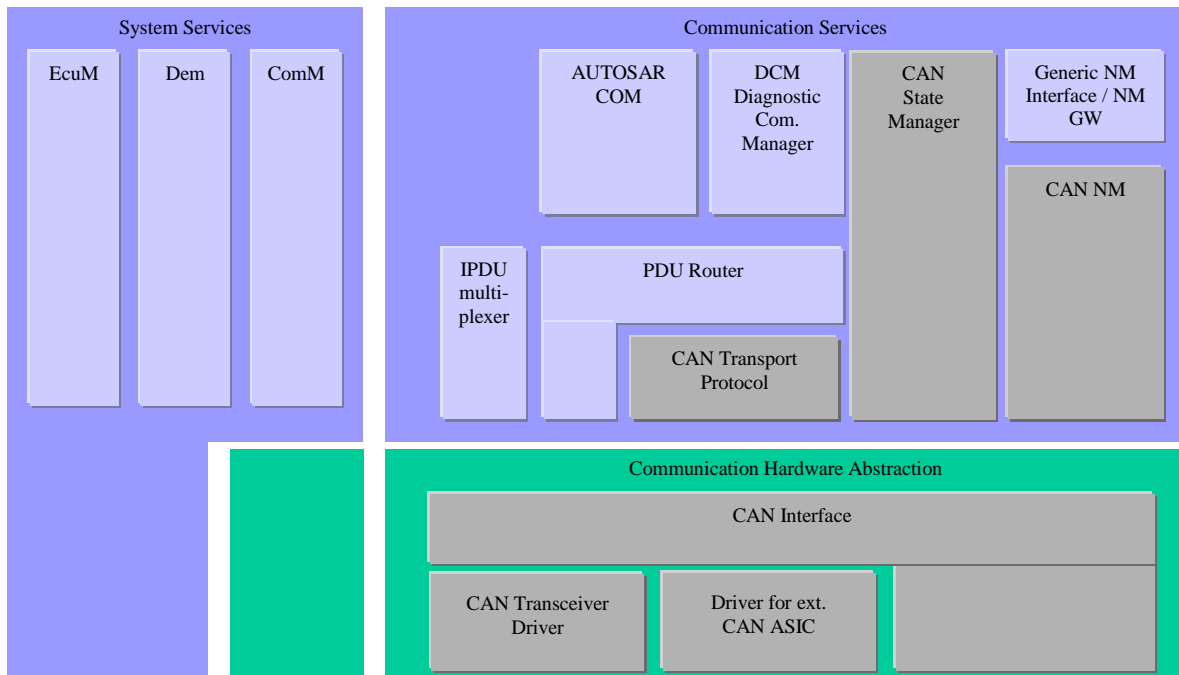
9	Sequence diagrams .....	196
9.1	Sequence diagram CanSm_StartCanController .....	197
9.2	Sequence diagram CanSm_StopCanController .....	198
10	Configuration specification .....	199
10.1	How to read this chapter .....	199
10.2	Containers and configuration parameters .....	199
10.2.1	CanSM .....	199
10.2.2	CanSMConfiguration .....	199
10.2.3	CanSMGeneral .....	200
10.2.4	CanSMManagerNetwork .....	204
10.2.5	CanSMController .....	206
10.2.6	CanSMDemEventParameterRefs .....	207
10.3	Published Information .....	208
11	CanSM unspecific / not applicable requirements .....	209



# 1 Introduction and functional overview

This specification describes the functionality, API and the configuration for the AUTOSAR Basic Software module CAN State Manager.

The AUTOSAR BSW stack specifies for each communication bus a bus specific state manager. This module shall implement the control flow for the respective bus. Like shown in the figure below, the CAN State Manager (CanSM) is a member of the Communication Service Layer. It interacts with the Communication Hardware Abstraction Layer and the System Service Layer.



**Figure 1-1: Layered Software Architecture from CanSM point of view**

## 2 Acronyms and abbreviations

<b>Abbreviation / Acronym:</b>	<b>Description:</b>
API	Application Program Interface
BSW	Basic Software
CAN	Controller Area Network
CanIf	CAN Interface
CanSM	CAN State Manager
ComM	Communication Manager
DEM	Diagnostic Event Manager
DET	Default Error Tracer
EcuM	ECU State Manager
PDU	Protocol Data Unit
RX	Receive
TX	Transmit
SchM	BSW Scheduler
SWC	Software Component
BswM	Basic Software Mode Manager

## 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

AUTOSAR\_TPS\_ECUConfiguration.pdf

[5] Specification of Standard Types

AUTOSAR\_SWS\_StandardTypes.pdf

[6] Specification of Communication Stack Types

AUTOSAR\_SWS\_CommunicationStackTypes.pdf

[7] Requirements on CAN

AUTOSAR\_SRS\_CAN.pdf

[8] Requirements on Mode Management

AUTOSAR\_SRS\_ModeManagement.pdf

[9] Specification of CAN Transceiver Driver

AUTOSAR\_SWS\_CANTransceiverDriver.pdf

[10] Specification of Communication Manager

AUTOSAR\_SWS\_COMMManager.pdf

[11] Specification of ECU State Manager

AUTOSAR\_SWS\_ECUStateManager.pdf

[12] Specification of Diagnostics Event Manager

AUTOSAR\_SWS\_DiagnosticEventManager.pdf

[13] Specification of CAN Interface

AUTOSAR\_SWS\_CANInterface.pdf

[14] Specification of BSW Scheduler

AUTOSAR\_SWS\_BSW\_Scheduler.pdf

[15] Specification of Default Error Tracer

AUTOSAR\_SWS\_DefaultErrorTracer.pdf

[16] Debugging Concept (internal)

[17] Vehicle and Application Mode Management Concept (internal)

[18] Specification of Basic Software Mode Manager

AUTOSAR\_SWS\_BSWModeManager.pdf

[19] Specification of CAN Network Management, AUTOSAR\_SWS\_Can\_NM.pdf

[20] Specification of Diagnostic Communication Manager

AUTOSAR\_SWS\_DiagnosticCommunicationManager.pdf

[21] General Specification of Basic Software Modules

AUTOSAR\_SWS\_BSWGeneral.pdf

## 3.2 Related standards and norms

None

## 3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [21] (SWS BSW General), which is also valid for CAN State Manager.

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

## **4 Constraints and assumptions**

### **4.1 Limitations**

The CanSM module can be used for CAN communication only. Its task is to operate with the CanIf module to control one or multiple underlying CAN Controllers and CAN Transceiver Drivers. Other protocols than CAN (i.e. LIN or FlexRay) are not supported.

### **4.2 Applicability to car domains**

The CAN State Manager module can be used for all domain applications whenever the CAN protocol is used.

## 5 Dependencies to other modules

The next sections give a brief description of configuration information and services the CanSM module requires from other modules.

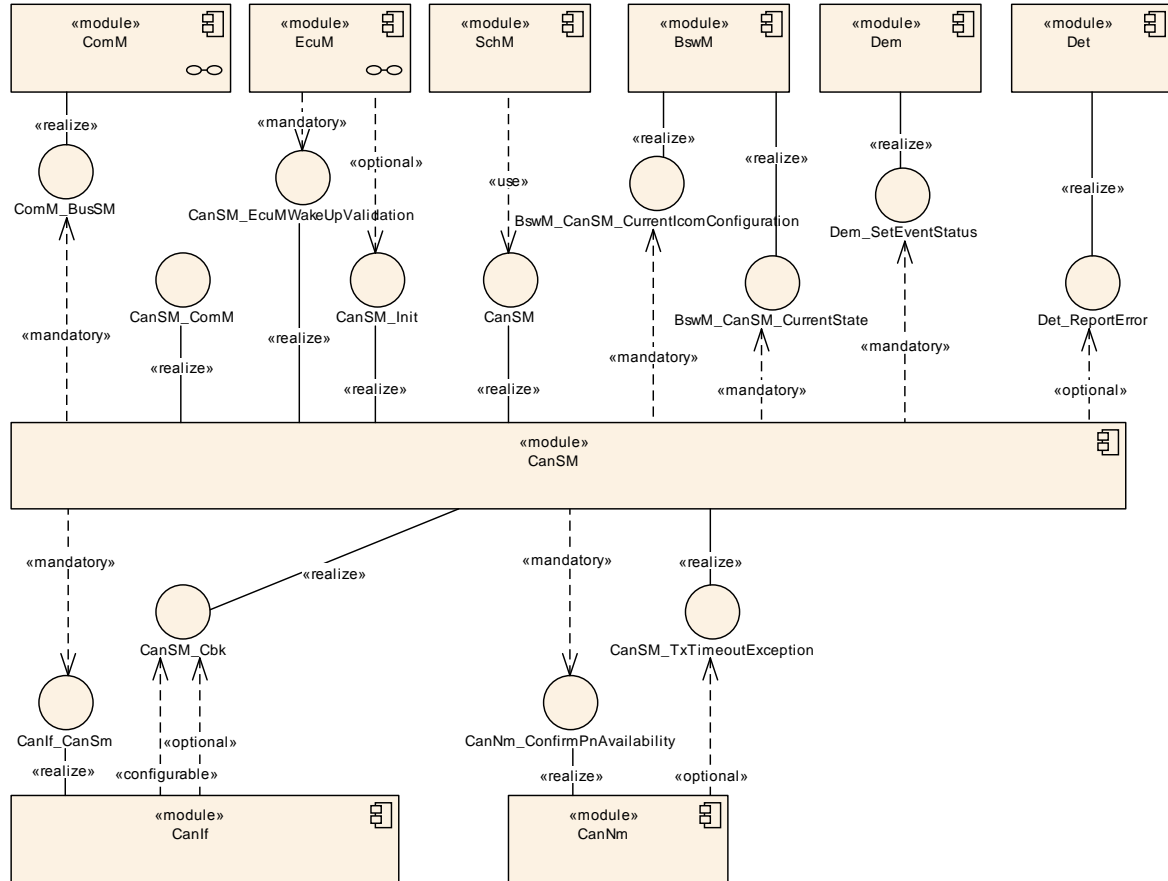


Figure 5-1: Module dependencies of the CanSM module

### 5.1 ECU State Manager (EcuM)

The EcuM module initializes the CanSM module and interacts with the CanSM module for the CAN wakeup validation (refer to [11] for a detailed specification of this module).

### 5.2 BSW Scheduler (SchM)

The BSW Scheduler module calls the main function of the CanSM module, which is necessary for the cyclic processes of the CanSM module (refer to [14] for a detailed specification of this module).

### **5.3 Communication Manager (ComM)**

The ComM module uses the API of the CanSM module to request communication modes of CAN networks, which are identified with unique network handles (refer to [10] for a detailed specification of this module).

The CanSM module notifies the current communication mode of its CAN networks to the ComM module.

### **5.4 CAN Interface (CanIf)**

The CanSM module uses the API of the CanIf module to control the operating modes of the CAN controllers and CAN transceivers assigned to the CAN networks (refer to [13] for a detailed specification of this module).

The CanIf module notifies the CanSM module about peripheral events.

### **5.5 Diagnostic Event Manager (DEM)**

The CanSM module reports bus specific production errors to the DEM module (refer to [12] for a detailed specification of this module).

### **5.6 Basic Software Mode Manager (BswM)**

The CanSM need to notify bus specific mode changes to the BswM module (refer to [18] for a detailed specification of this module).

### **5.7 CAN Network Management (CanNm)**

The CanSM module needs to notify the partial network availability to the CanNm module and shall handle notified CanNm timeout exceptions in case of partial networking (ref. to [19] for a detailed specification of this module).

### **5.8 Default Error Tracer (DET)**

The CanSM module reports development and runtime errors to the DET module. Development Errors are only reported if development error handling is switched on by configuration (refer to [15] for a detailed specification of this module).

## **5.9 File structure**

### **5.9.1 Code file structure**

For details refer to the chapter 5.1.6 “Code file structure” in *SWS\_BSWGeneral*



### 5.9.2 Header file structure

**[SWS\_CanSM\_00008]** [ The header file `CanSM.h` shall export CanSM module specific types and the APIs `CanSM_GetVersionInfo` and `CanSM_Init.`] (SRS\_BSW\_00447)

### 5.9.3 Version check

For details refer to the chapter 5.1.8 “Version Check” in *SWS\_BSWGeneral*.

## 6 Requirements traceability

Requirement	Description	Satisfied by
SRS_BSW_00003	All software modules shall provide version and identification information	SWS_CanSM_00024, SWS_CanSM_00374
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_CanSM_00023, SWS_CanSM_00596
SRS_BSW_00327	Error values naming convention	noname
SRS_BSW_00333	For each callback function it shall be specified if it is called from interrupt context or not	SWS_CanSM_00064, SWS_CanSM_00189, SWS_CanSM_00190, SWS_CanSM_00235
SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_CanSM_91001
SRS_BSW_00337	Classification of development errors	noname, SWS_CanSM_00654
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_CanSM_00023, SWS_CanSM_00596
SRS_BSW_00359	All AUTOSAR Basic Software Modules callback functions shall avoid return types other than void if possible	SWS_CanSM_00064, SWS_CanSM_00189, SWS_CanSM_00190, SWS_CanSM_00235
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	noname, SWS_CanSM_00660
SRS_BSW_00385	List possible error notifications	noname
SRS_BSW_00400	Parameter shall be selected from multiple sets of parameters after code has been loaded and started	SWS_CanSM_00023, SWS_CanSM_00597
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_CanSM_00023, SWS_CanSM_00596
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_CanSM_00023, SWS_CanSM_00596

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_CanSM_00023, SWS_CanSM_00184, SWS_CanSM_00596
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_CanSM_00024, SWS_CanSM_00374
SRS_BSW_00414	Init functions shall have a pointer to a configuration structure as single parameter	SWS_CanSM_00023, SWS_CanSM_00596
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_CanSM_00498, SWS_CanSM_00522, SWS_CanSM_00605
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_CanSM_00065, SWS_CanSM_00167
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_CanSM_00065, SWS_CanSM_00167
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_CanSM_00023, SWS_CanSM_00597
SRS_BSW_00447	Standardizing Include file structure of BSW Modules Implementing Autosar Service	SWS_CanSM_00008
SRS_BSW_00466	Classification of extended production errors	SWS_CanSM_00664
SRS_Can_01142	The CAN State Manager shall offer a network abstract API to upper layer	SWS_CanSM_00062, SWS_CanSM_00065, SWS_CanSM_00167, SWS_CanSM_00182, SWS_CanSM_00183, SWS_CanSM_00186, SWS_CanSM_00187, SWS_CanSM_00188, SWS_CanSM_00266, SWS_CanSM_00278, SWS_CanSM_00282, SWS_CanSM_00284, SWS_CanSM_00360, SWS_CanSM_00369, SWS_CanSM_00370, SWS_CanSM_00371, SWS_CanSM_00372, SWS_CanSM_00385, SWS_CanSM_00399, SWS_CanSM_00410, SWS_CanSM_00422, SWS_CanSM_00423, SWS_CanSM_00425, SWS_CanSM_00426, SWS_CanSM_00427, SWS_CanSM_00428, SWS_CanSM_00429, SWS_CanSM_00430, SWS_CanSM_00431, SWS_CanSM_00432, SWS_CanSM_00433, SWS_CanSM_00434, SWS_CanSM_00436, SWS_CanSM_00437, SWS_CanSM_00438, SWS_CanSM_00439,

		<p>SWS_CanSM_00440, SWS_CanSM_00441,  SWS_CanSM_00442, SWS_CanSM_00443,  SWS_CanSM_00444, SWS_CanSM_00445,  SWS_CanSM_00446, SWS_CanSM_00447,  SWS_CanSM_00448, SWS_CanSM_00449,  SWS_CanSM_00450, SWS_CanSM_00451,  SWS_CanSM_00452, SWS_CanSM_00453,  SWS_CanSM_00454, SWS_CanSM_00455,  SWS_CanSM_00456, SWS_CanSM_00457,  SWS_CanSM_00458, SWS_CanSM_00459,  SWS_CanSM_00460, SWS_CanSM_00461,  SWS_CanSM_00462, SWS_CanSM_00464,  SWS_CanSM_00465, SWS_CanSM_00466,  SWS_CanSM_00467, SWS_CanSM_00468,  SWS_CanSM_00469, SWS_CanSM_00470,  SWS_CanSM_00471, SWS_CanSM_00472,  SWS_CanSM_00473, SWS_CanSM_00474,  SWS_CanSM_00475, SWS_CanSM_00476,  SWS_CanSM_00477, SWS_CanSM_00478,  SWS_CanSM_00479, SWS_CanSM_00483,  SWS_CanSM_00484, SWS_CanSM_00485,  SWS_CanSM_00486, SWS_CanSM_00487,  SWS_CanSM_00488, SWS_CanSM_00489,  SWS_CanSM_00490, SWS_CanSM_00491,  SWS_CanSM_00492, SWS_CanSM_00493,  SWS_CanSM_00494, SWS_CanSM_00496,  SWS_CanSM_00497, SWS_CanSM_00499,  SWS_CanSM_00500, SWS_CanSM_00502,  SWS_CanSM_00503, SWS_CanSM_00504,  SWS_CanSM_00505, SWS_CanSM_00506,  SWS_CanSM_00507, SWS_CanSM_00508,  SWS_CanSM_00509, SWS_CanSM_00510,  SWS_CanSM_00511, SWS_CanSM_00512,  SWS_CanSM_00514, SWS_CanSM_00515,  SWS_CanSM_00517, SWS_CanSM_00518,  SWS_CanSM_00521, SWS_CanSM_00524,  SWS_CanSM_00525, SWS_CanSM_00526,  SWS_CanSM_00527, SWS_CanSM_00528,  SWS_CanSM_00529, SWS_CanSM_00530,  SWS_CanSM_00531, SWS_CanSM_00532,  SWS_CanSM_00533, SWS_CanSM_00534,  SWS_CanSM_00535, SWS_CanSM_00538,  SWS_CanSM_00540, SWS_CanSM_00541,  SWS_CanSM_00542, SWS_CanSM_00543,  SWS_CanSM_00550, SWS_CanSM_00555,  SWS_CanSM_00556, SWS_CanSM_00557,  SWS_CanSM_00558, SWS_CanSM_00561,  SWS_CanSM_00569, SWS_CanSM_00576,  SWS_CanSM_00577, SWS_CanSM_00578,  SWS_CanSM_00579, SWS_CanSM_00580,  SWS_CanSM_00581, SWS_CanSM_00582,  SWS_CanSM_00584, SWS_CanSM_00586,  SWS_CanSM_00587, SWS_CanSM_00588,  SWS_CanSM_00589, SWS_CanSM_00590,  SWS_CanSM_00591, SWS_CanSM_00593,  SWS_CanSM_00594, SWS_CanSM_00595,  SWS_CanSM_00599, SWS_CanSM_00600,  SWS_CanSM_00602, SWS_CanSM_00603,  SWS_CanSM_00604, SWS_CanSM_00607,  SWS_CanSM_00608, SWS_CanSM_00623,</p>
--	--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

		SWS_CanSM_00624, SWS_CanSM_00625, SWS_CanSM_00626, SWS_CanSM_00627, SWS_CanSM_00628, SWS_CanSM_00629, SWS_CanSM_00630, SWS_CanSM_00631, SWS_CanSM_00632, SWS_CanSM_00633, SWS_CanSM_00634, SWS_CanSM_00635, SWS_CanSM_00636, SWS_CanSM_00639, SWS_CanSM_00641, SWS_CanSM_00642, SWS_CanSM_00651, SWS_CanSM_00653
SRS_Can_01144	The CAN State Manager shall support a configurable BusOff recovery time	SWS_CanSM_00600, SWS_CanSM_00602, SWS_CanSM_00603, SWS_CanSM_00604, SWS_CanSM_00606, SWS_CanSM_00637
SRS_Can_01145	The CAN State Manager shall control the assigned CAN Devices	SWS_CanSM_00062, SWS_CanSM_00065, SWS_CanSM_00167, SWS_CanSM_00182, SWS_CanSM_00183, SWS_CanSM_00369, SWS_CanSM_00370, SWS_CanSM_00396, SWS_CanSM_00397, SWS_CanSM_00398, SWS_CanSM_00399, SWS_CanSM_00400, SWS_CanSM_00401, SWS_CanSM_00410, SWS_CanSM_00411, SWS_CanSM_00412, SWS_CanSM_00413, SWS_CanSM_00414, SWS_CanSM_00415, SWS_CanSM_00416, SWS_CanSM_00417, SWS_CanSM_00418, SWS_CanSM_00419, SWS_CanSM_00420, SWS_CanSM_00421, SWS_CanSM_00423, SWS_CanSM_00425, SWS_CanSM_00426, SWS_CanSM_00427, SWS_CanSM_00428, SWS_CanSM_00429, SWS_CanSM_00430, SWS_CanSM_00431, SWS_CanSM_00432, SWS_CanSM_00433, SWS_CanSM_00434, SWS_CanSM_00436, SWS_CanSM_00437, SWS_CanSM_00438, SWS_CanSM_00439, SWS_CanSM_00440, SWS_CanSM_00441, SWS_CanSM_00442, SWS_CanSM_00443, SWS_CanSM_00444, SWS_CanSM_00445, SWS_CanSM_00446, SWS_CanSM_00447, SWS_CanSM_00448, SWS_CanSM_00449, SWS_CanSM_00450, SWS_CanSM_00451, SWS_CanSM_00452, SWS_CanSM_00453, SWS_CanSM_00454, SWS_CanSM_00455, SWS_CanSM_00456, SWS_CanSM_00457, SWS_CanSM_00458, SWS_CanSM_00459, SWS_CanSM_00460, SWS_CanSM_00461, SWS_CanSM_00462, SWS_CanSM_00464, SWS_CanSM_00465, SWS_CanSM_00466, SWS_CanSM_00467, SWS_CanSM_00468, SWS_CanSM_00469, SWS_CanSM_00470, SWS_CanSM_00471, SWS_CanSM_00472, SWS_CanSM_00473, SWS_CanSM_00474, SWS_CanSM_00475, SWS_CanSM_00476, SWS_CanSM_00477, SWS_CanSM_00478, SWS_CanSM_00479, SWS_CanSM_00483, SWS_CanSM_00484, SWS_CanSM_00485, SWS_CanSM_00486, SWS_CanSM_00487, SWS_CanSM_00488, SWS_CanSM_00489, SWS_CanSM_00490, SWS_CanSM_00491, SWS_CanSM_00492, SWS_CanSM_00493, SWS_CanSM_00494, SWS_CanSM_00496,

		<p>SWS_CanSM_00497, SWS_CanSM_00499, SWS_CanSM_00500, SWS_CanSM_00507, SWS_CanSM_00508, SWS_CanSM_00509, SWS_CanSM_00510, SWS_CanSM_00511, SWS_CanSM_00512, SWS_CanSM_00514, SWS_CanSM_00515, SWS_CanSM_00517, SWS_CanSM_00518, SWS_CanSM_00521, SWS_CanSM_00524, SWS_CanSM_00525, SWS_CanSM_00526, SWS_CanSM_00527, SWS_CanSM_00528, SWS_CanSM_00529, SWS_CanSM_00531, SWS_CanSM_00532, SWS_CanSM_00533, SWS_CanSM_00534, SWS_CanSM_00535, SWS_CanSM_00538, SWS_CanSM_00540, SWS_CanSM_00541, SWS_CanSM_00542, SWS_CanSM_00543, SWS_CanSM_00546, SWS_CanSM_00550, SWS_CanSM_00555, SWS_CanSM_00556, SWS_CanSM_00557, SWS_CanSM_00558, SWS_CanSM_00560, SWS_CanSM_00576, SWS_CanSM_00577, SWS_CanSM_00578, SWS_CanSM_00579, SWS_CanSM_00580, SWS_CanSM_00581, SWS_CanSM_00582, SWS_CanSM_00584, SWS_CanSM_00588, SWS_CanSM_00589, SWS_CanSM_00590, SWS_CanSM_00591, SWS_CanSM_00600, SWS_CanSM_00602, SWS_CanSM_00603, SWS_CanSM_00604, SWS_CanSM_00607, SWS_CanSM_00608, SWS_CanSM_00609, SWS_CanSM_00610, SWS_CanSM_00611, SWS_CanSM_00612, SWS_CanSM_00613, SWS_CanSM_00616, SWS_CanSM_00617, SWS_CanSM_00618, SWS_CanSM_00619, SWS_CanSM_00620, SWS_CanSM_00621, SWS_CanSM_00622, SWS_CanSM_00623, SWS_CanSM_00624, SWS_CanSM_00625, SWS_CanSM_00626, SWS_CanSM_00627, SWS_CanSM_00628, SWS_CanSM_00629, SWS_CanSM_00630, SWS_CanSM_00631, SWS_CanSM_00632, SWS_CanSM_00633, SWS_CanSM_00634, SWS_CanSM_00636, SWS_CanSM_00638, SWS_CanSM_00639, SWS_CanSM_00641, SWS_CanSM_00642, SWS_CanSM_00651, SWS_CanSM_00653</p>
SRS_Can_01146	The CAN State Manager shall contain a CAN BusOff recovery algorithm for each used CAN Controller	SWS_CanSM_00600, SWS_CanSM_00602, SWS_CanSM_00603, SWS_CanSM_00604, SWS_CanSM_00606, SWS_CanSM_00637
SRS_Can_01158	The CAN stack shall provide a TX offline active mode for ECU passive mode	SWS_CanSM_00435, SWS_CanSM_00516, SWS_CanSM_00539, SWS_CanSM_00644, SWS_CanSM_00645, SWS_CanSM_00646, SWS_CanSM_00647, SWS_CanSM_00649, SWS_CanSM_00650, SWS_CanSM_00656
SRS_Can_01164	-	SWS_CanSM_00658, SWS_CanSM_91001
SRS_ModeMgm_09084	The Communication Manager shall provide an API which allows application to query the	SWS_CanSM_00063

	current communication mode	
SRS_ModeMgm_09251	PNC communication state shall be forwarded to the BswM	SWS_CanSM_00598

## 7 Functional specification

This chapter specifies the different functions of the CanSM module in the AUTOSAR BSW architecture.

An ECU can have different communication networks. Each network has to be identified with an unique network handle. The ComM module requests communication modes from the networks. It knows by its configuration, which handle is assigned to what kind of network. In case of CAN, it uses the CanSM module.

The CanSM module is responsible for the control flow abstraction of CAN networks:

It changes the communication modes of the configured CAN networks depending on the mode requests from the ComM module.

Therefore the CanSM module uses the API of the CanIf module. The CanIf module is responsible for the control flow abstraction of the configured CAN Controllers and CAN Transceivers (the data flow abstraction of the CanIf module is not relevant for the CanSM module). Any change of the CAN Controller modes and CAN Transceiver modes will be notified by the CanIf module to the CanSM module. Depending on this notifications and state of the CAN network state machine, which the CanSM module shall implement for each configured CAN network, the CanSM module notifies the ComM and the BswM (ref. to chapter 7.2 for details).



### 7.1 General requirements

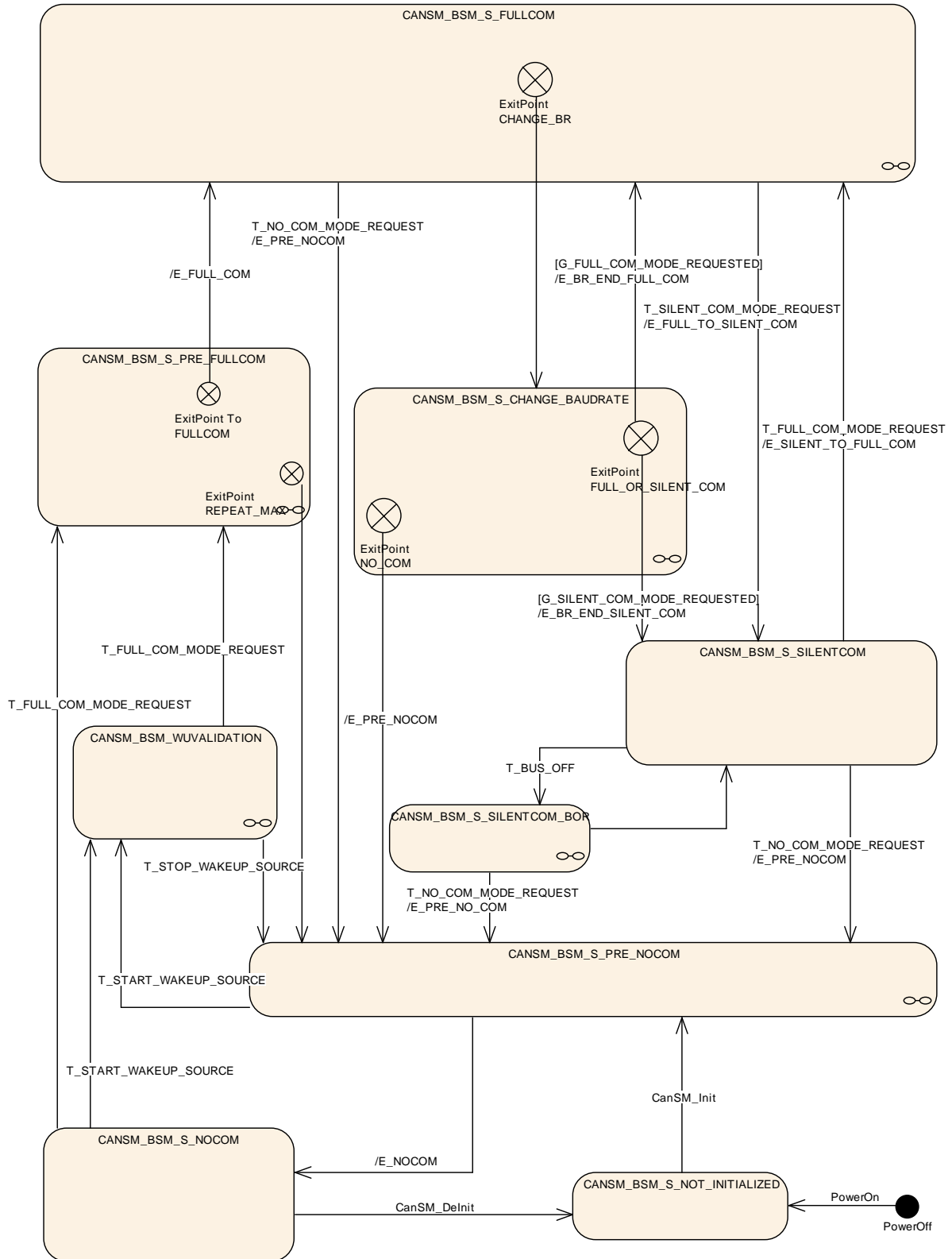
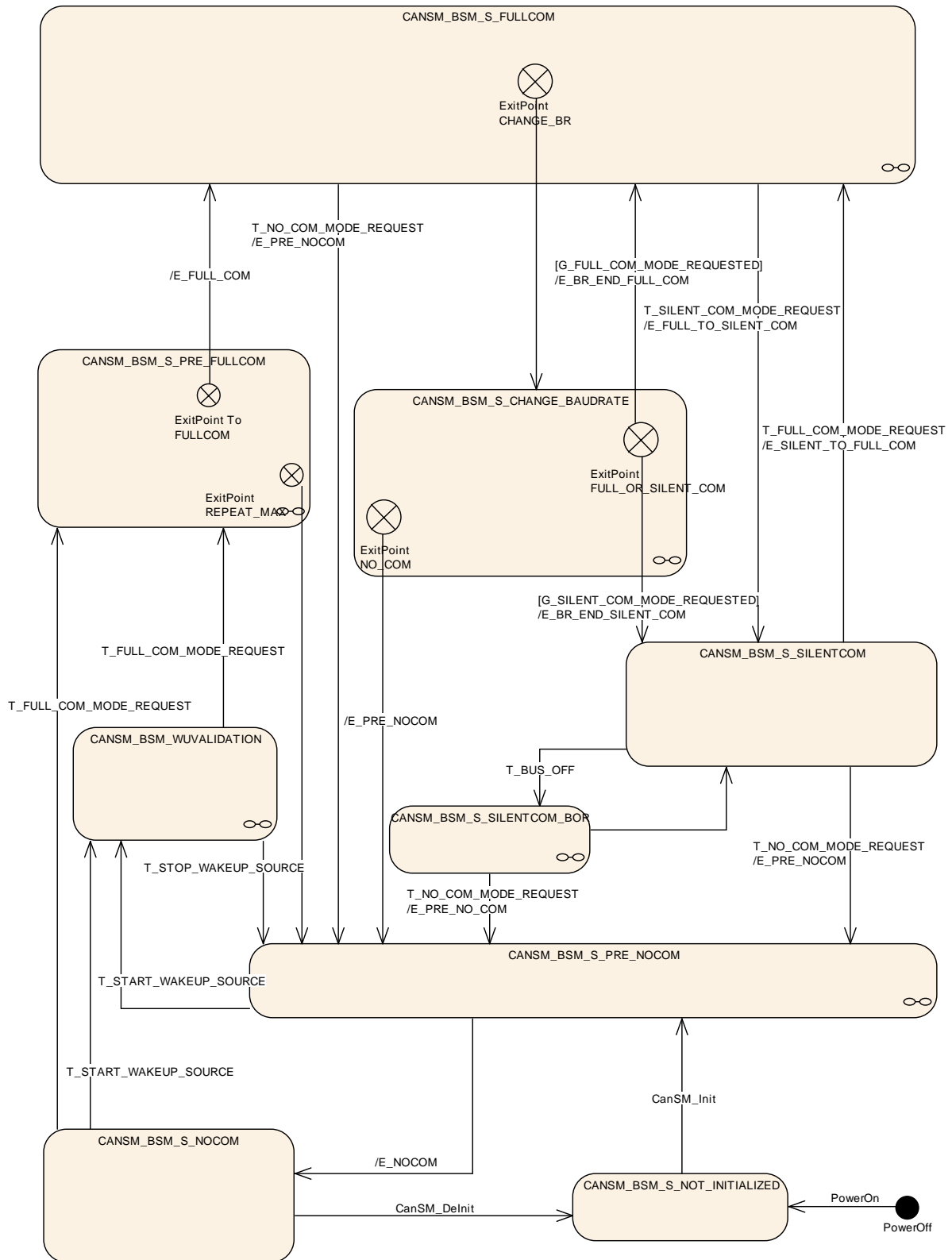


Figure 7-1 : CANSM\_BSM, state machine diagram for one CAN network

**[SWS\_CanSM\_00266]** [ The CanSM module shall store the current network mode for each configured CAN network internally (ref. to to [ECUC\\_CanSM\\_00126](#)).] (SRS\_Can\_01142)

**[SWS\_CanSM\_00284]** [ The internally stored network modes of the CanSM module can have the values `COMM_NO_COMMUNICATION`, `COMM_SILENT_COMMUNICATION`, `COMM_FULL_COMMUNICATION`.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00428]** [ All effects of the CanSM state machine `CANSM_BSM` (ref. to



**Figure 7-1)** shall be operated in the context of the CanSM main function (ref. to SWS\_CanSM\_00065).|(SRS\_Can\_01142, SRS\_Can\_01145)

[SWS\_CanSM\_00278] [ If the CanSM state machine CANSM\_BSM (ref. to

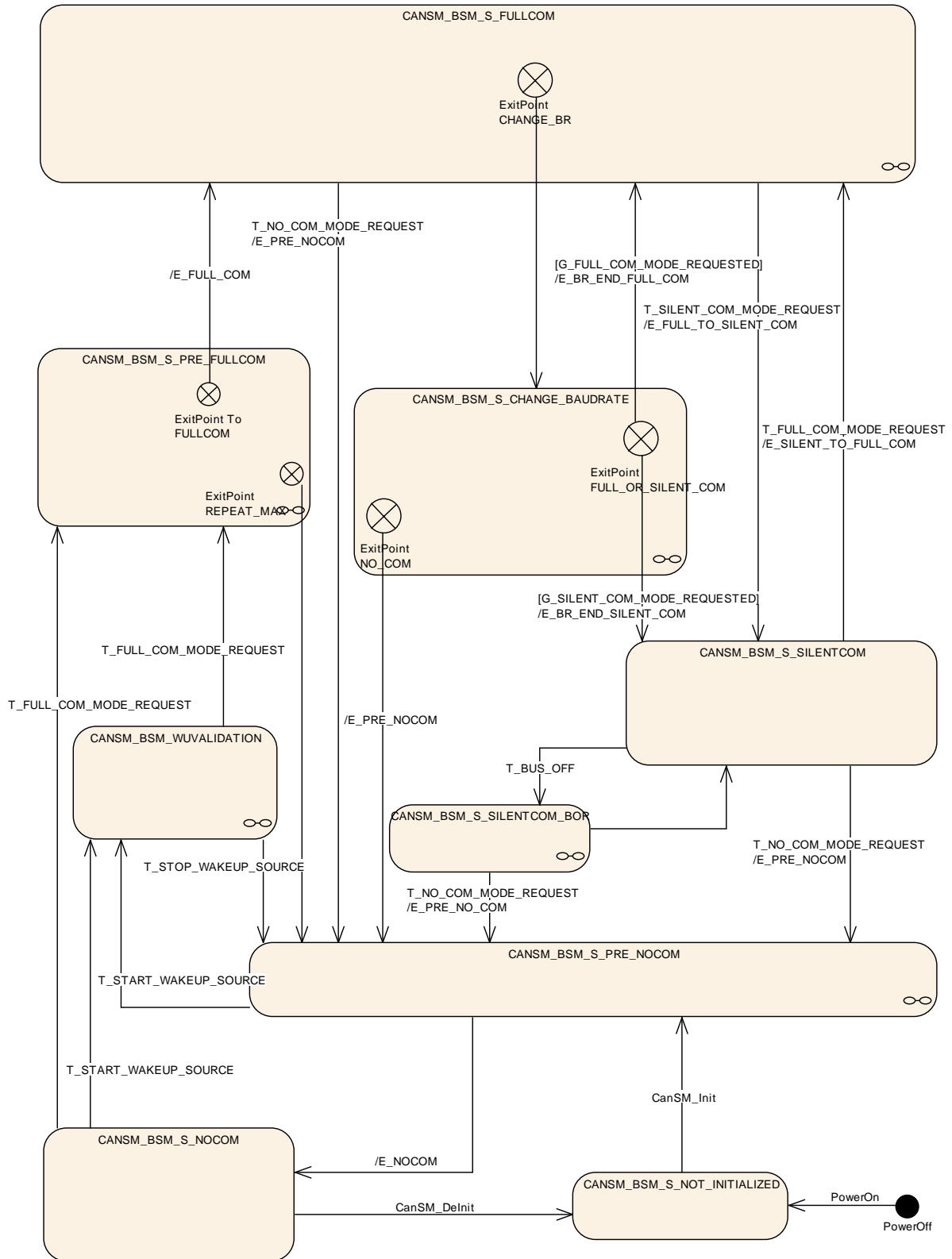


Figure 7-1) is in the state CANSM\_BSM\_S\_NOT\_INITIALIZED, it shall deny network mode requests from the ComM module (ref. to [SWS\\_CanSM\\_00062](#)).] (SRS\_Can\_01142)

**[SWS\_CanSM\_00385]** [ If CanSM has repeated one of the CanIf API calls `CanIf_SetControllerMode`, `CanIf_SetTrcvMode`, `CanIf_ClearTrcvWufFlag` or `CanIf_CheckTrcvWakeFlag` more often than `CanSMModeRequestRepetitionMax` (ref. to [ECUC\\_CanSM\\_00335](#)) without getting the return value `E_OK` or without getting the corresponding mode indication callbacks `CanSM_ControllerModeIndication`, `CanSM_TransceiverModeIndication`, `CanSM_ClearTrcvWufFlagIndication` or `CanSM_CheckTransceiverWakeFlagIndication`, CanSM shall call the function `Det_ReportRuntimeError` with `ErrorId` parameter `CANSM_E_MODE_REQUEST_TIMEOUT`.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00422]** [ If the CanIf module notifies PN availability for a configured CAN Transceiver to the CanSM module with the callback function `CanSM_ConfirmPnAvailability` (ref. to [SWS\\_CanSM\\_00419](#)), then the CanSM module shall call the API `CanNm_ConfirmPnAvailability` (ref. to chapter 8.5.1) with the related CAN network as `channel` to confirm the PN availability to the `CanNm` module.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00560]** [ If no `CanSMTransceiverId` (ref. to [ECUC\\_CanSM\\_00137](#)) is configured for a CAN Network, then the CanSM module shall bypass all specified `CanIf_SetTrcvMode` (e. g. [SWS\\_CanSM\\_00446](#)) calls for the CAN Network and proceed in the different state transitions as if it has got the supposed `CanSM_TransceiverModeIndication` already (e. g. [SWS\\_CanSM\\_00448](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00635]**[ The CanSM module shall store for each configured CAN network (ref. to [ECUC\\_CanSM\\_00126](#)) the latest communication mode request, which has been accepted by returning `E_OK` in the API request `CanSM_RequestComMode` (ref. to [SWS\\_CANSM\\_00062](#), [SWS\\_CANSM\\_00182](#)) and use it as trigger for the state machine of the related CAN network (ref. to

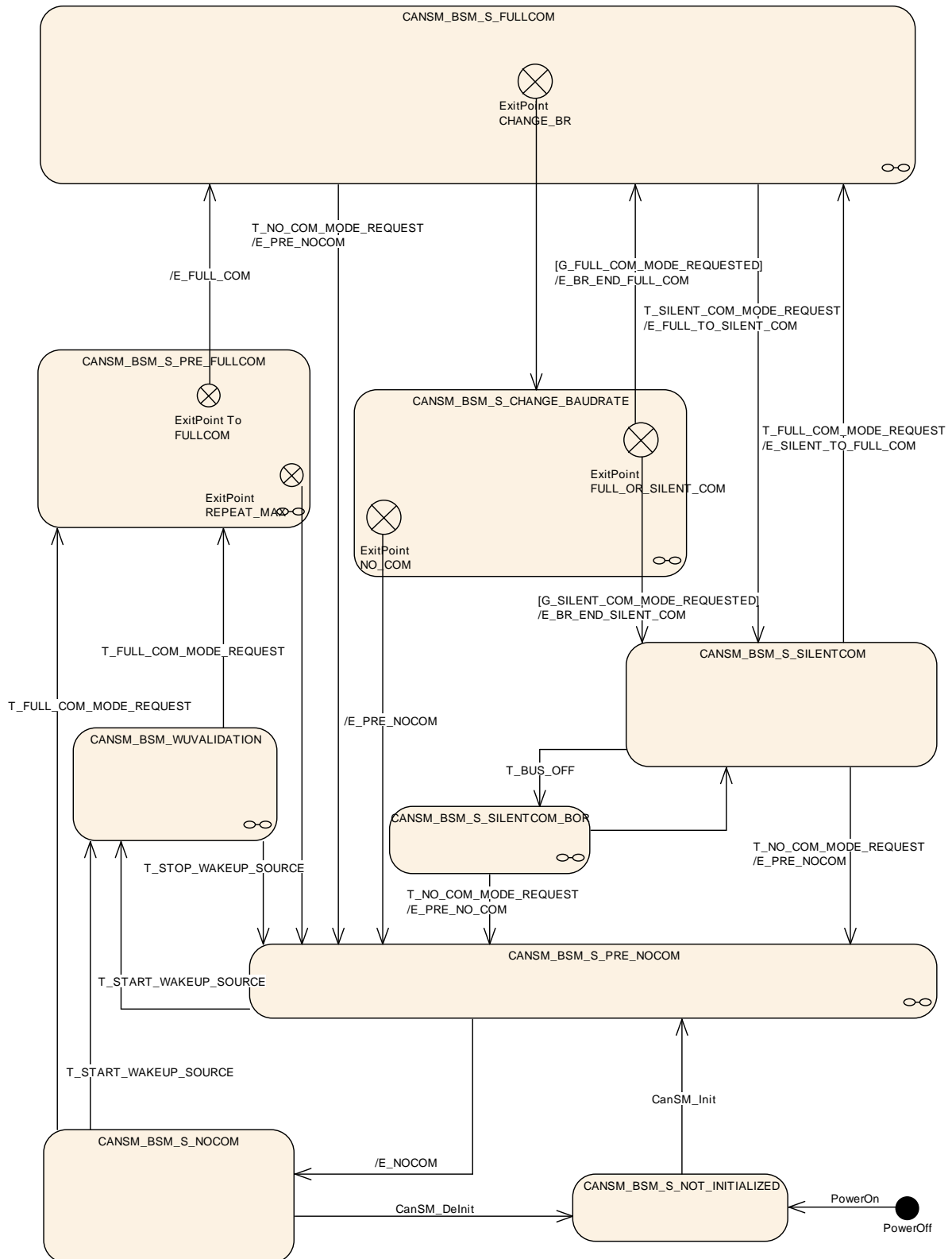


Figure 7-1), [SWS CanSM 00427](#), [SWS CanSM 00429](#), [SWS CanSM 00499](#), [SWS CanSM 00542](#), [SWS CanSM 00543](#), [SWS CANSM 00425](#), [SWS CANSM 00426](#), [SWS CANSM 00554](#).] (SRS\_Can\_01142)

[SWS\_CanSM\_00638] The CanSM module shall store after every successful CAN controller mode change (ref. to [SWS CANSM 00396](#)) or bus-off conditioned change

to CAN\_CS\_STOPPED (ref. to [SWS\\_CANSM\\_00064](#)), the changed mode internally for each CAN controller.] (SRS\_Can\_01145)

### 7.2 State machine for each CAN network

The diagram (ref. to

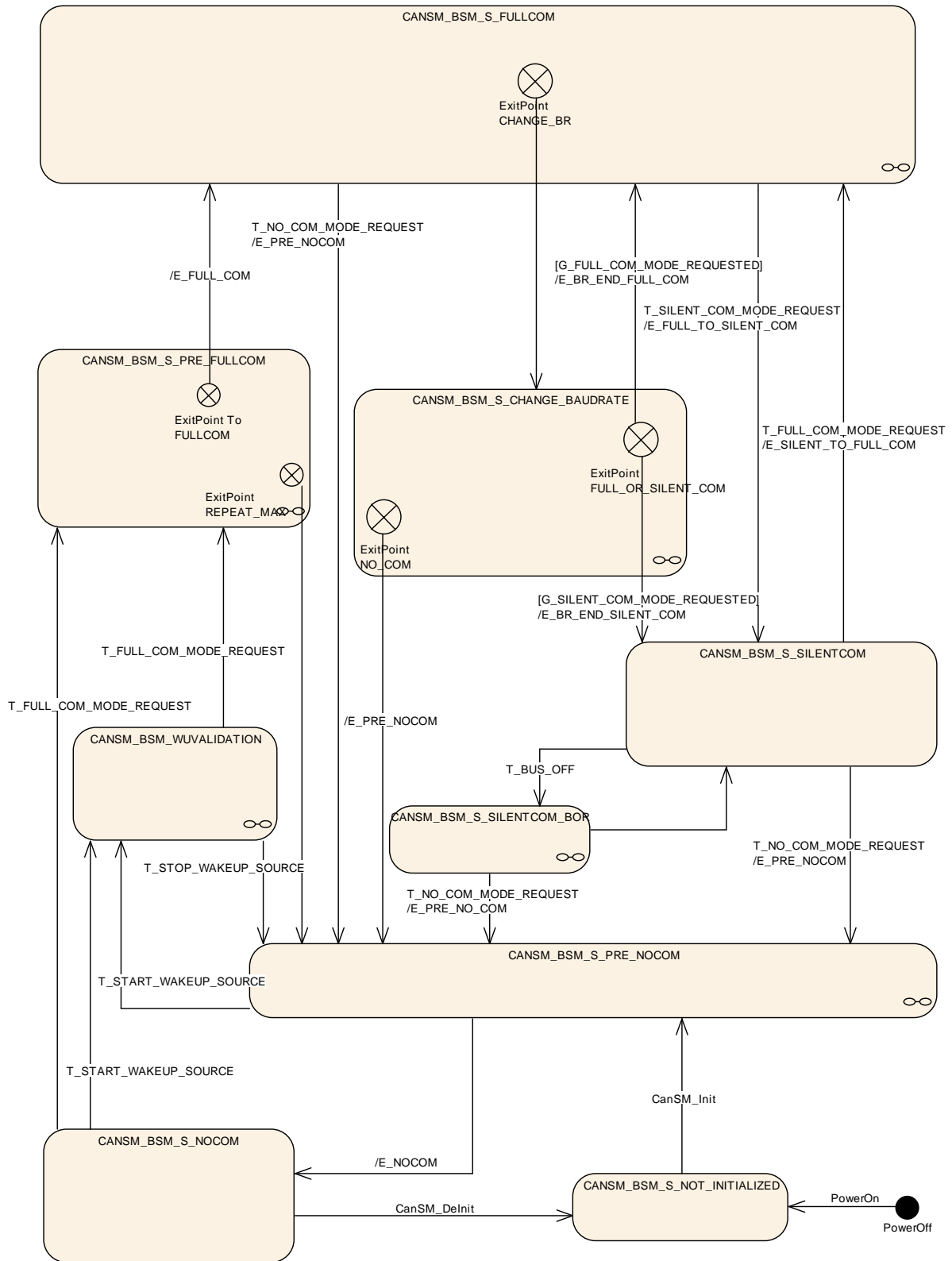


Figure 7-1) specifies the behavioral state machine of the CanSM module, which shall be implemented for each configured CAN network (ref. to [ECUC CanSM\\_00126](#))



7.2.1 Trigger: PowerOn

[SWS\_CanSM\_00424] [ After PowerOn the CanSM state machines (ref. to

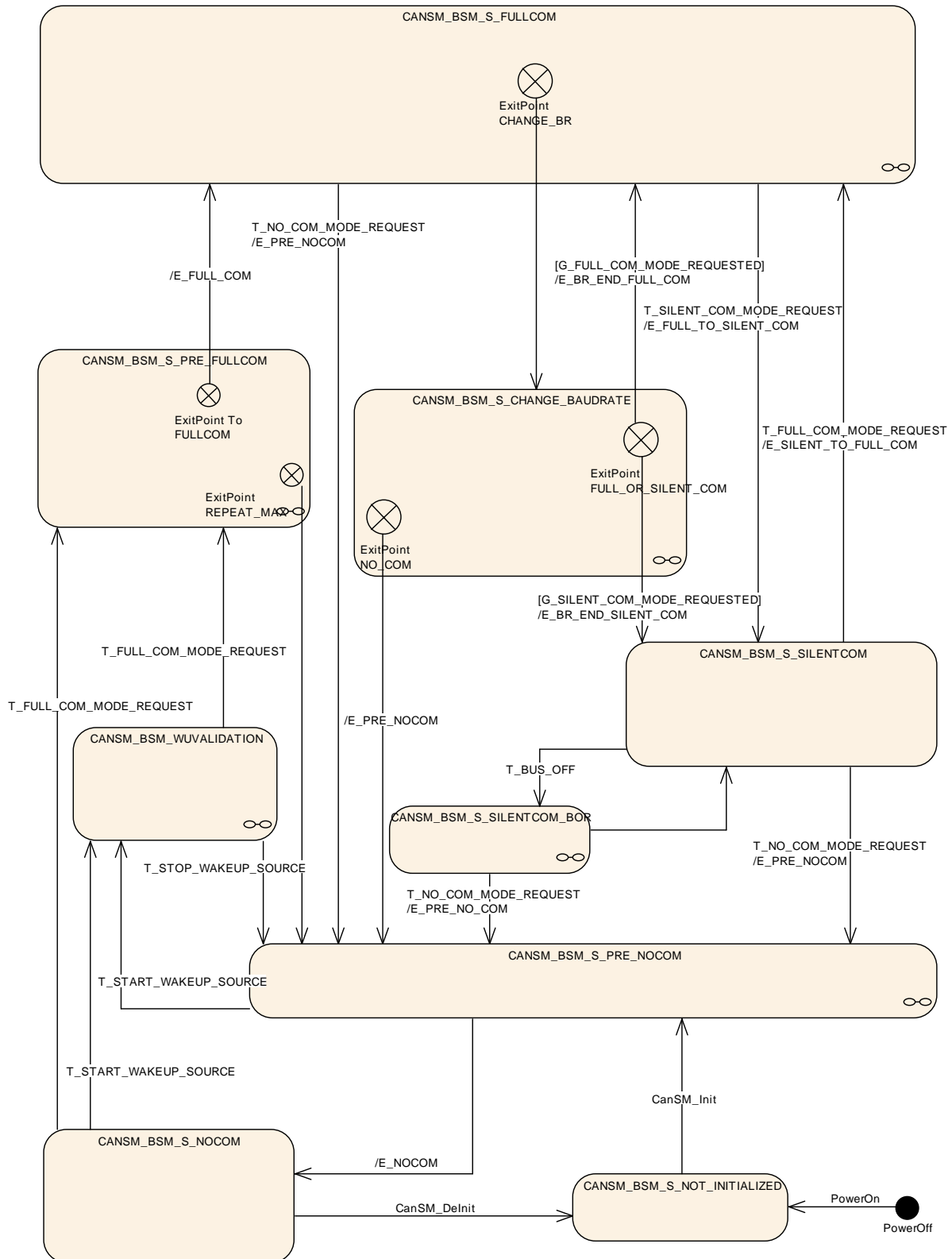


Figure 7-1) shall be in the state CANSM\_BSM\_NOT\_INITIALIZED.]

**7.2.2 Trigger: CanSM\_Init**

**[SWS\_CanSM\_00423]** [ If the CanSM module is requested with the function CanSM\_Init (ref. to chapter 8.3.1), this shall trigger the CanSM state machines (ref. to

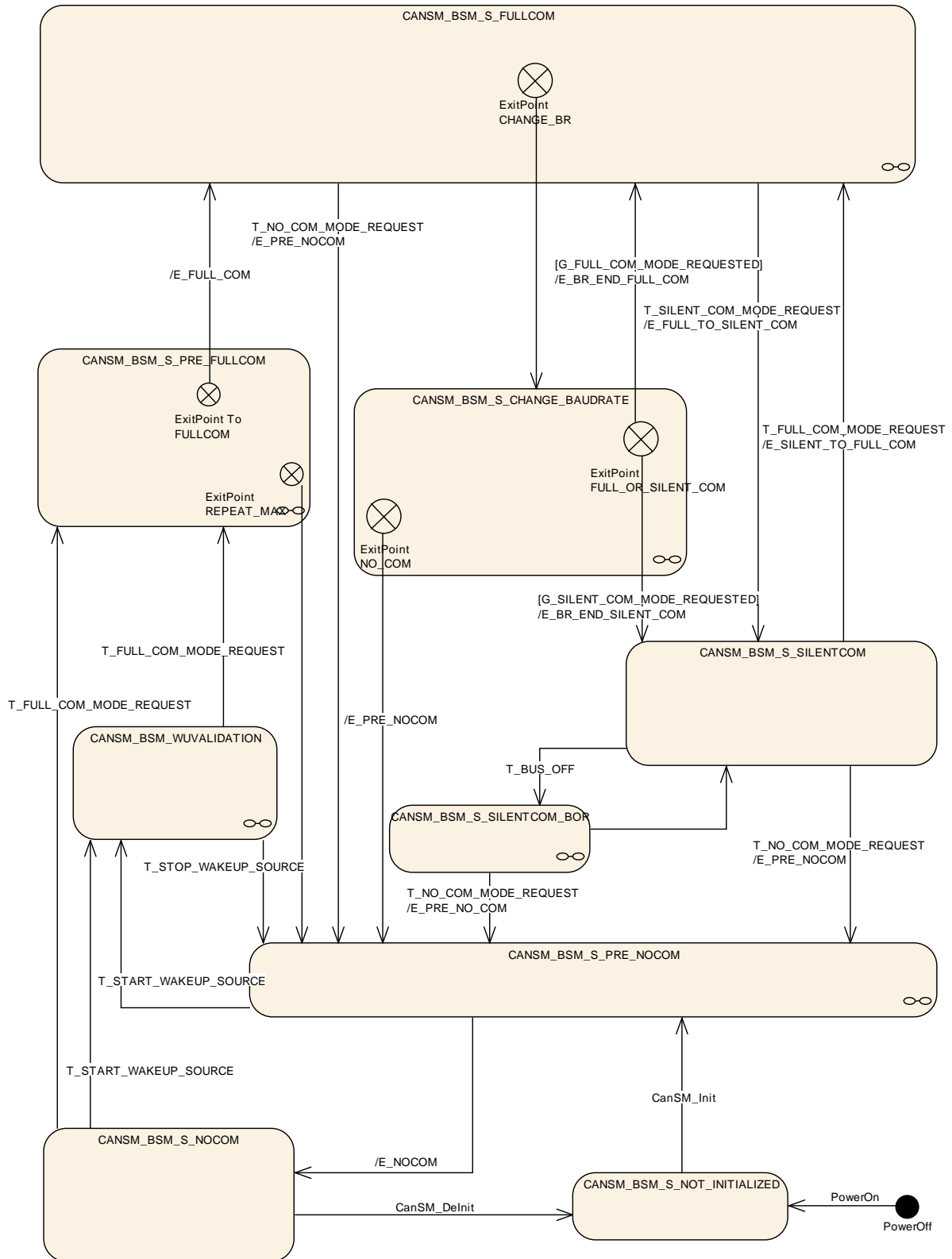


Figure 7-1) for all configured CAN Networks (ref. to [ECUC\\_CanSM\\_00126](#)) with the trigger `CanSM_Init. ]` (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.3 Trigger: `CanSM_Delnit`

[SWS\_CanSM\_00658] If the CanSM module is requested with the function `CanSM_Delnit`, this shall trigger the CanSM state machines (ref. to Figure 7-1) for all configured CAN Networks (ref. to [ECUC\\_CanSM\\_00126](#)) with the trigger `CanSM_Delnit. ]` (SRS\_Can\_01164)

Note: Caller of the `CanSM_Delnit` function has to ensure all CAN networks are in the state `CANSM_NO_COMMUNICATION`

### 7.2.4 Trigger: `T_START_WAKEUP_SOURCE`

[SWS\_CanSM\_00607] If the API request `CanSM_StartWakeUpSource` (ref. to [SWS\\_CanSM\\_00609](#)) returns `E_OK` (ref. to [SWS\\_CanSM\\_00616](#)), it shall trigger the state machine (ref. to

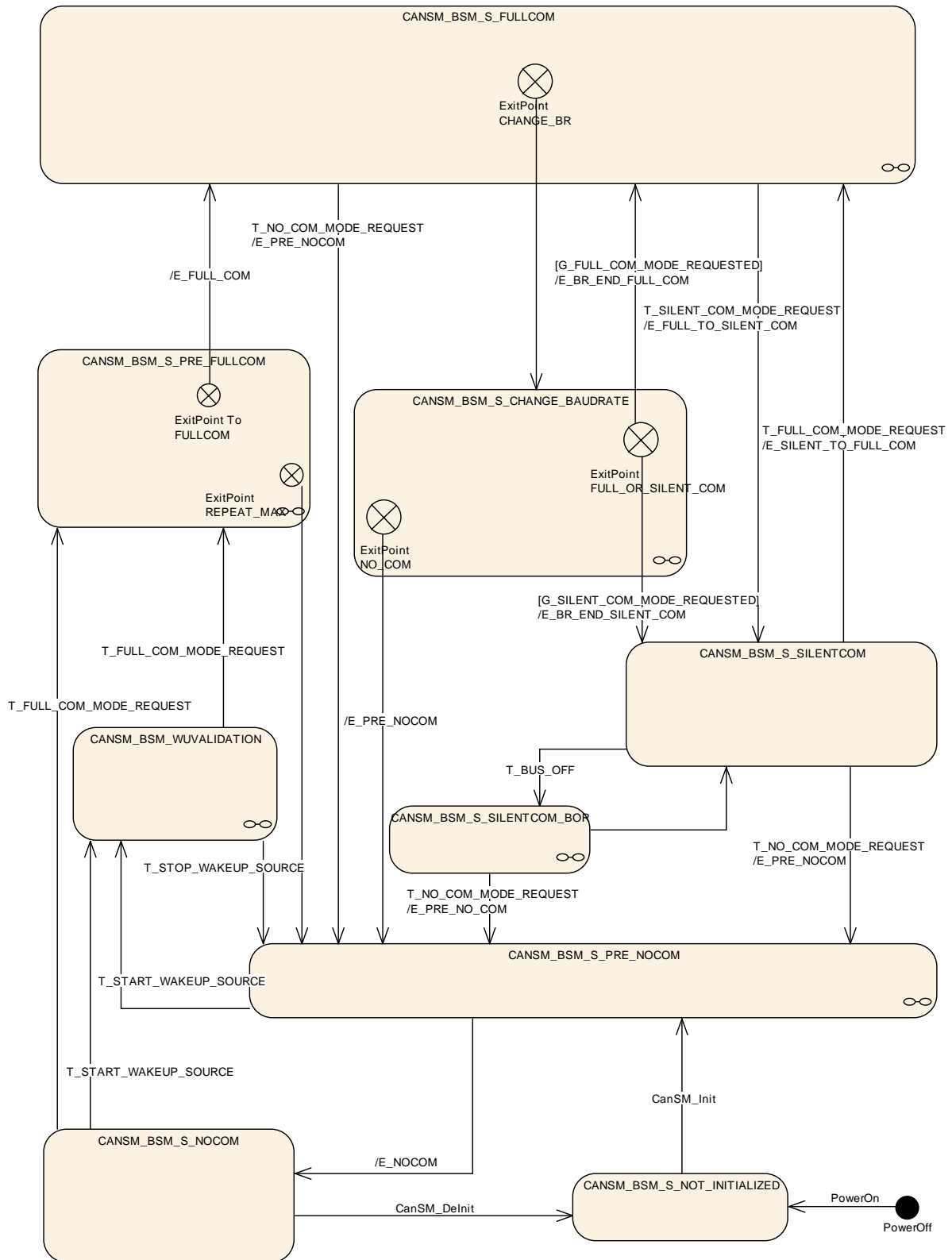


Figure 7-1) with T\_START\_WAKEUP\_SOURCE.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.5 Trigger: T\_STOP\_WAKEUP\_SOURCE**

[SWS\_CanSM\_00608] If the API request CanSM\_StopWakeUpSource (ref. to [SWS\\_CanSM\\_00610](#)) returns E\_OK (ref. to [SWS\\_CanSM\\_00622](#)), it shall trigger the state machine (ref. to

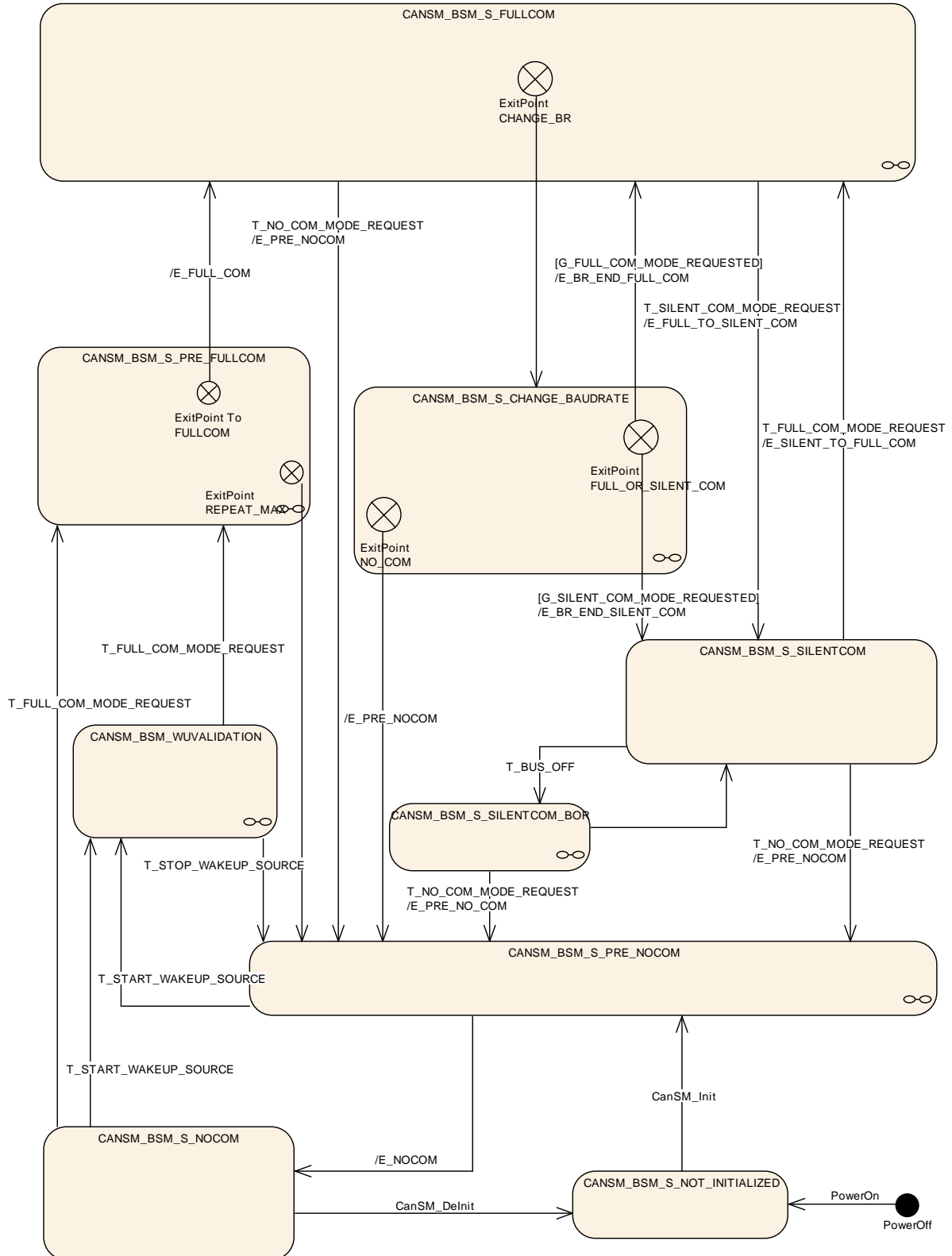


Figure 7-1) with T\_STOP\_WAKEUP\_SOURCE.] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.6 Trigger: T\_FULL\_COM\_MODE\_REQUEST

**[SWS\_CanSM\_00425]** [ The API request `CanSM_RequestComMode` (ref. to [SWS\\_CanSM\\_00635](#)) with the parameter `ComM_Mode` equal to `COMM_FULL_COMMUNICATION` shall trigger the state machine with `T_FULL_COM_MODE_REQUEST`, if the function parameter `network` matches the configuration parameter `CANSM_NETWORK_HANDLE` (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.7 Trigger: T\_SILENT\_COM\_MODE\_REQUEST

**[SWS\_CanSM\_00499]** [ The API request `CanSM_RequestComMode` (ref. to [SWS\\_CanSM\\_00635](#)) with the parameter `ComM_Mode` equal to `COMM_SILENT_COMMUNICATION` shall trigger the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to

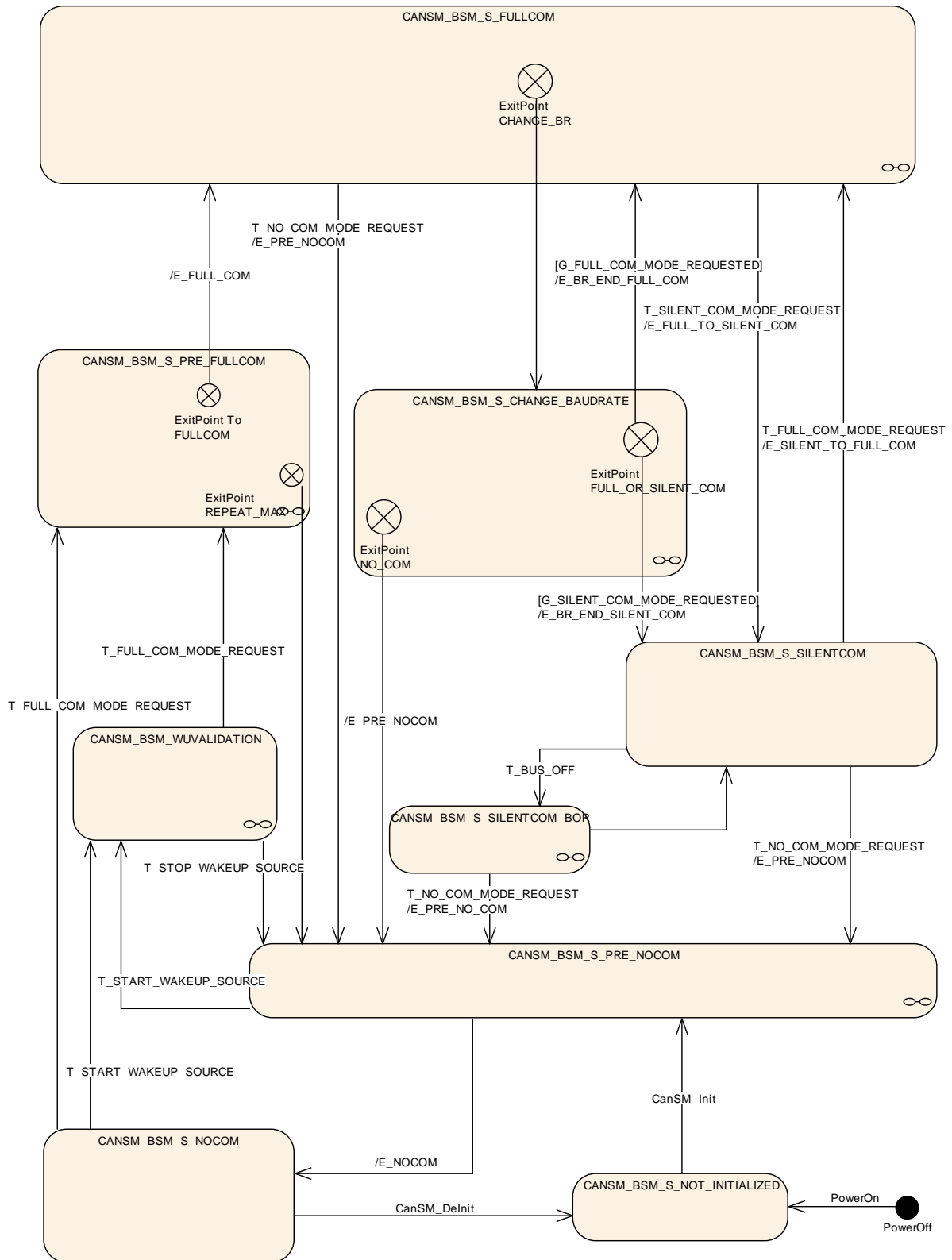


Figure 7-1) with `T_SILENT_COM_MODE_REQUEST`, which corresponds to the function parameter `network` and the configuration parameter `CANSM_NETWORK_HANDLE` (ref. to [ECUC CanSM 00161](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

Rationale: Regular use case for the transition of the CanNm Network mode to the CanNm Prepare Bus-Sleep mode.

### 7.2.8 Trigger: T\_NO\_COM\_MODE\_REQUEST

**[SWS\_CanSM\_00426]** [ The API request `CanSM_RequestComMode` (ref. to [SWS\\_CanSM\\_00635](#)) with the parameter `ComM_Mode` equal to `COMM_NO_COMMUNICATION` shall trigger the state machine with `T_NO_COM_MODE_REQUEST`, if the function parameter `network` matches the configuration parameter `CANSM_NETWORK_HANDLE` (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01142, SRS\_Can\_01145)

*Remark: Depending on the ComM configuration, the ComM module will request `COMM_SILENT_COMMUNICATION` first and then `COMM_NO_COMMUNICATION` or `COMM_NO_COMMUNICATION` directly (`ComMNmVariant=LIGHT`).*

### 7.2.9 Trigger: T\_BUS\_OFF

**[SWS\_CanSM\_00606]**[ The callback function `CanSM_ControllerBusOff` (ref. to [SWS\\_CanSM\\_00064](#)) shall trigger the state machine `CANSM_BSM` (ref. to



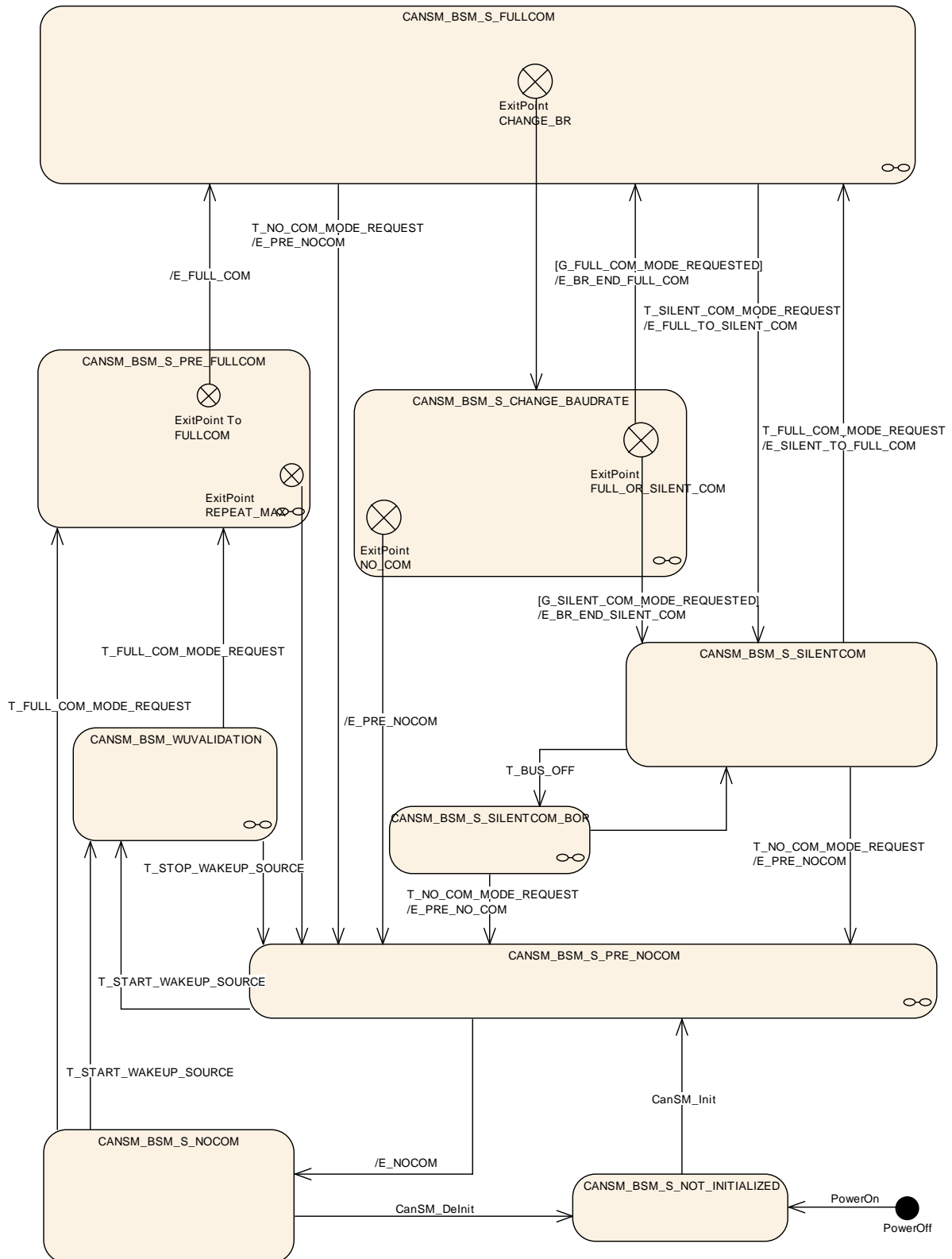


Figure 7-1) for the CAN network with `T_BUS_OFF`, if one of its configured CAN controllers matches to the function parameter `ControllerId` of the callback function `CanSM_ControllerBusOff`.] (SRS\_Can\_01144, SRS\_Can\_01146)



handle of the state machine has been with the parameter `ComM_Mode` equal to `COMM_FULL_COMMUNICATION`.] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.11 Guarding condition: `G_SILENT_COM_MODE_REQUESTED`

[SWS\_CanSM\_00429] | The guarding condition `G_SILENT_COM_MODE_REQUESTED` of the `CanSM_BSM` state machine (ref. to

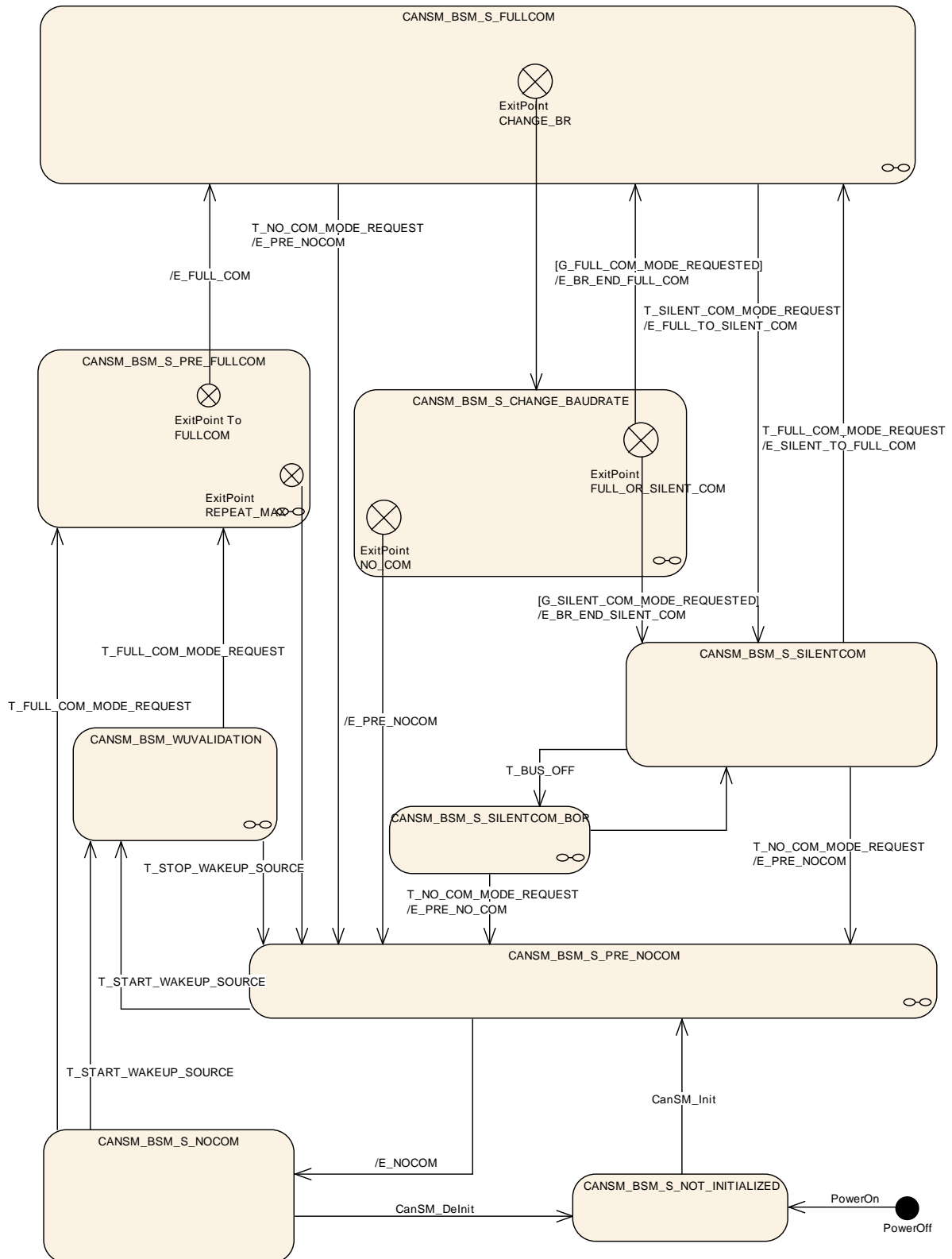


Figure 7-1) shall evaluate, if the latest accepted communication mode request with CanSM\_RequestComMode (ref. to [SWS CanSM\\_00635](#)) for the respective network handle of the state machine has been with the parameter ComM\_Mode equal to COMM\_SILENT\_COMMUNICATION.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.12 Effect: E\_PRE\_NOCOM**

[SWS\_CanSM\_00431] | The effect E\_PRE\_NOCOM of the CanSM\_BSM state machine (ref. to

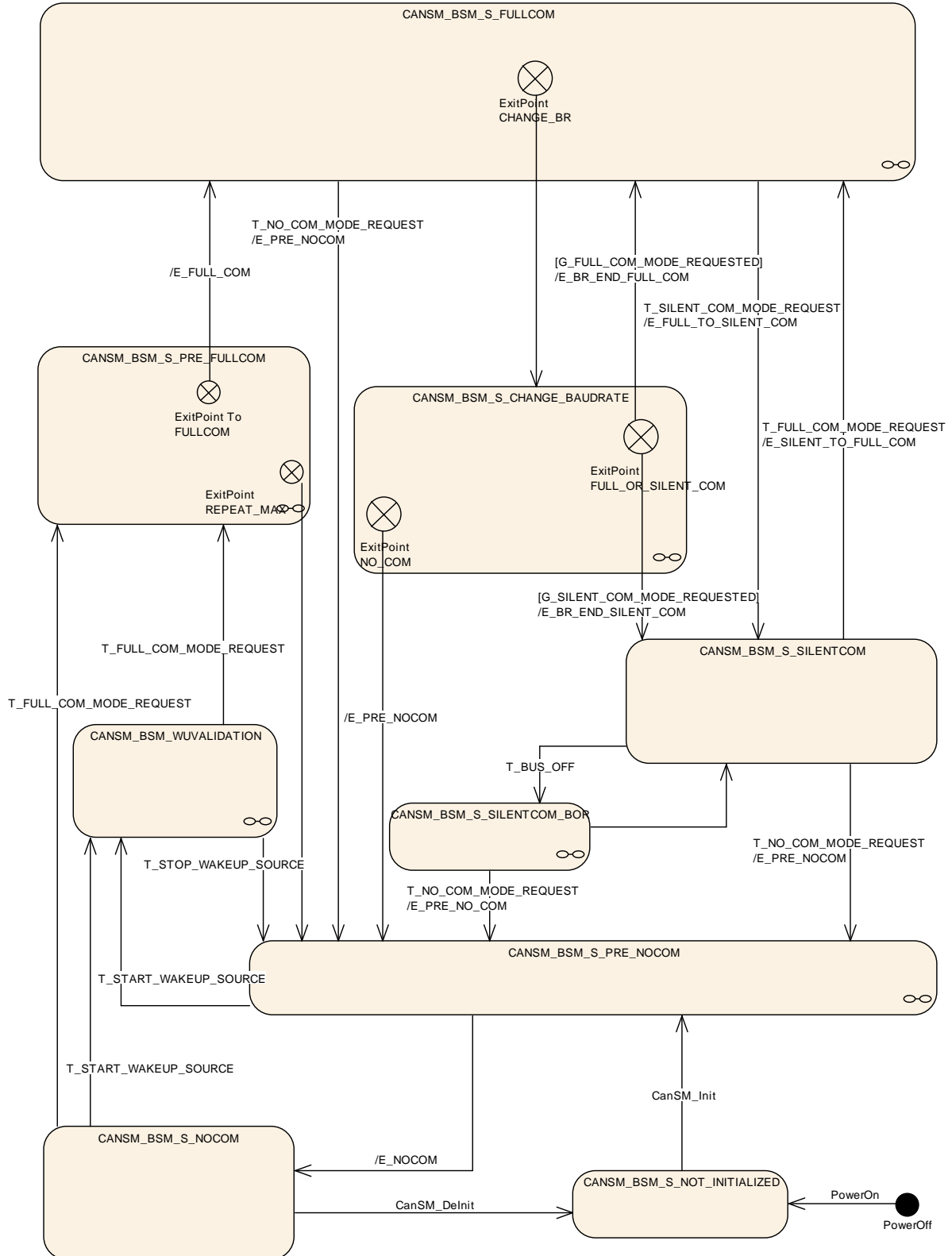


Figure 7-1) shall call for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network :=`

CanSMComMNetworkHandleRef and CurrentState :=  
CANSM\_BSWM\_NO\_COMMUNICATION.] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.13 Effect: E\_NOCOM

[SWS\_CanSM\_00430] | The effect E\_NOCOM of the CanSM\_BSM state machine  
(ref. to

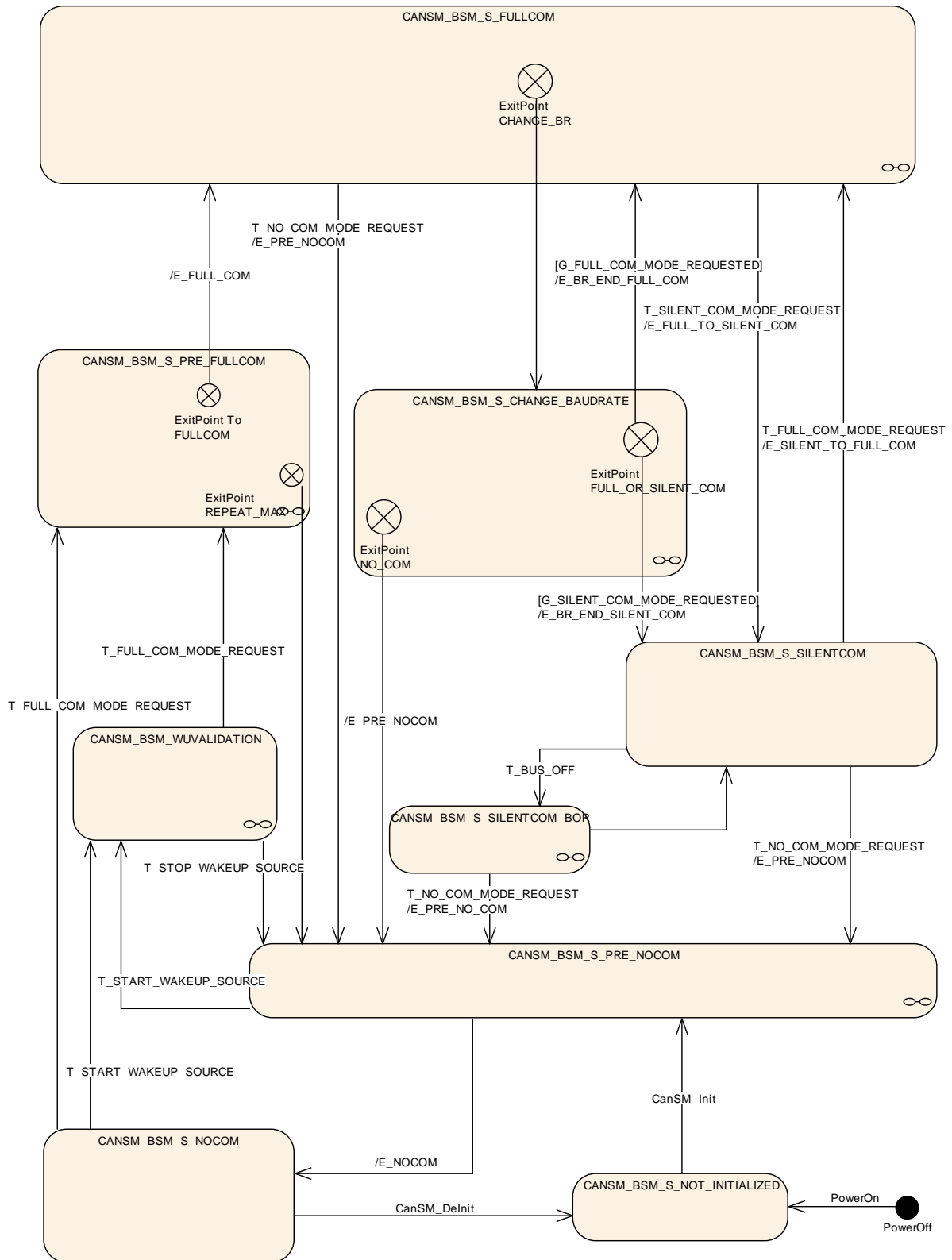


Figure 7-1) shall change the internally stored network mode (ref. to [SWS\\_CanSM\\_00266](#)) of the addressed CAN network to COMM\_NO\_COMMUNICATION. ] (SRS\_Can\_01142, SRS\_Can\_01145)

[SWS\_CanSM\_00651] If a communication mode request for the network is present already (ref. to [SWS\\_CanSM\\_00635](#)) and the stored communication mode request is

COMM\_NO\_COMMUNICATION, then the effect E\_NOCOM of the CanSM\_BSM state machine (ref. to

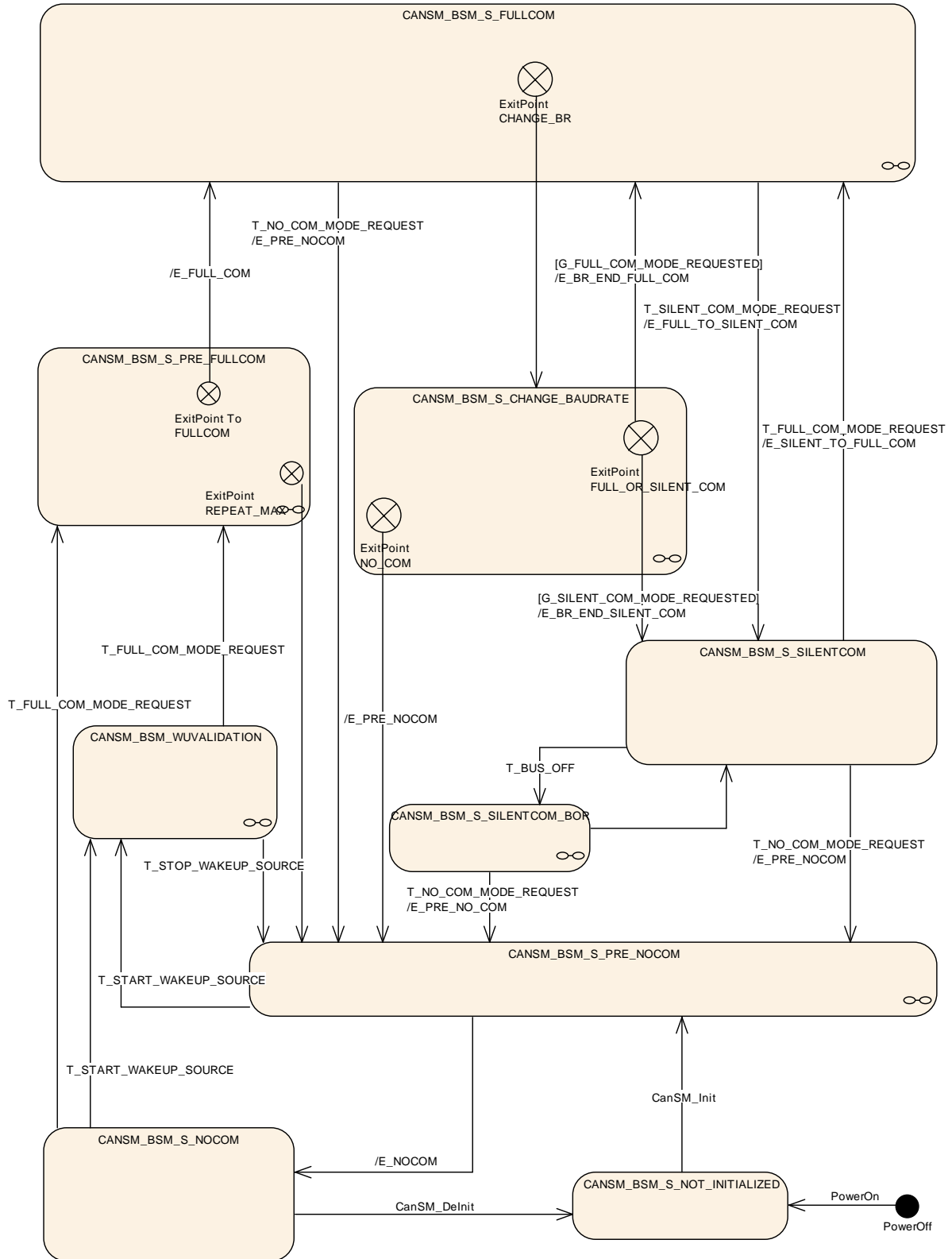


Figure 7-1) shall call the API `ComM_BusSM_ModeIndication` with the parameters `Channel := CanSMComMNetworkHandleRef` (ref. to [ECUC CanSM\\_00161](#)) and `ComMode := COMM_NO_COMMUNICATION.`] (SRS\_Can\_01142, SRS\_Can\_01145)



**7.2.14 Effect: E\_FULL\_COM**

[SWS\_CanSM\_00539] [ If ECU passive is FALSE (ref. to [SWS\\_CanSM\\_00646](#)), then the effect E\_FULL\_COM of the CanSM\_BSM state machine (ref. to

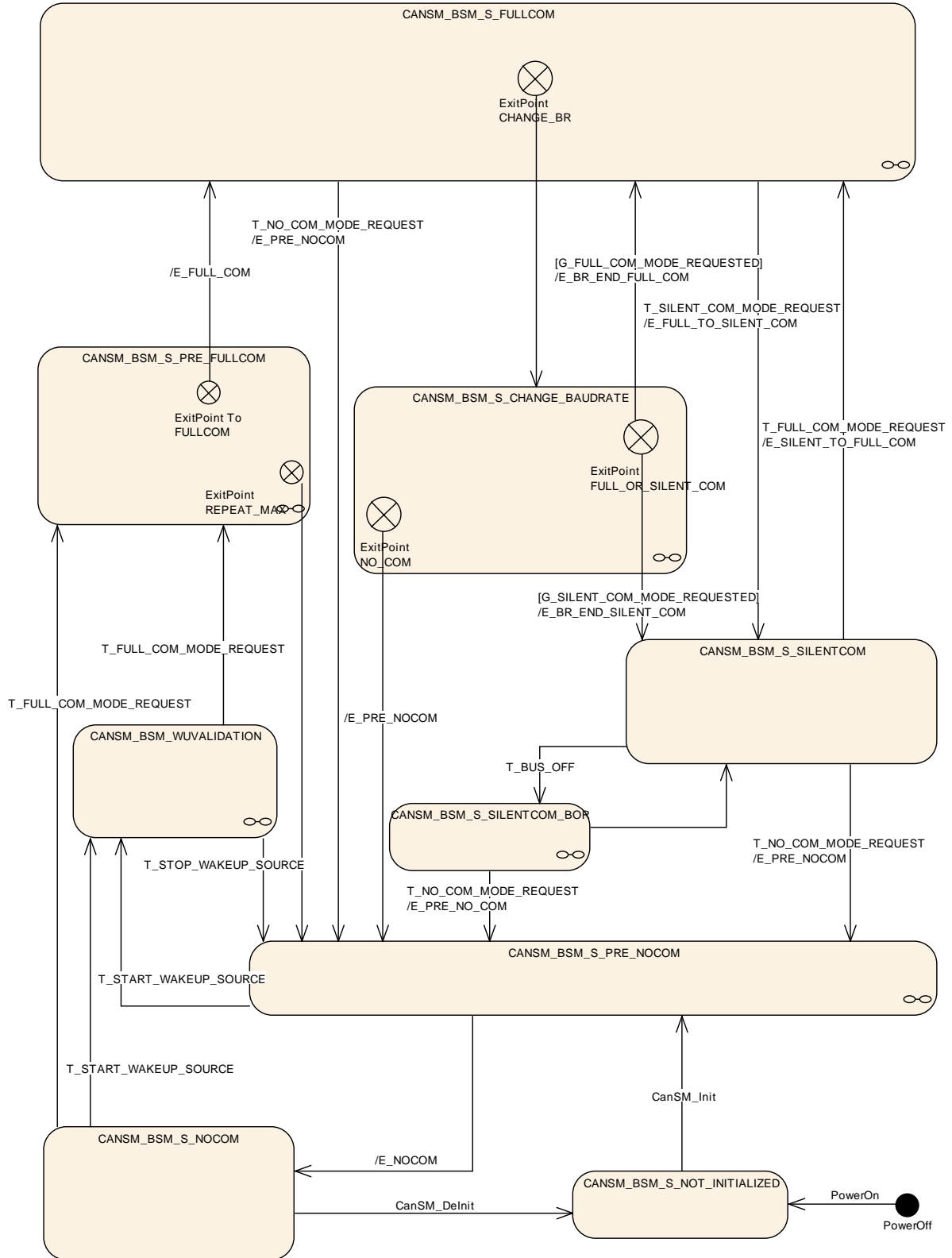


Figure 7-1) shall call at 1<sup>st</sup> place for each configured CAN controller of the CAN network the API CanIf\_SetPduMode with the parameters ControllerId :=

CanSMControllerId (ref. to [ECUC CanSM\\_00141](#)) and PduModeRequest := CANIF\_ONLINE.] (SRS\_Can\_01158)

[SWS\_CanSM\_00647] [ If ECU passive is TRUE (ref. to [SWS CanSM\\_00646](#)), then the effect E\_FULL\_COM of the CanSM\_BSM state machine (ref. to

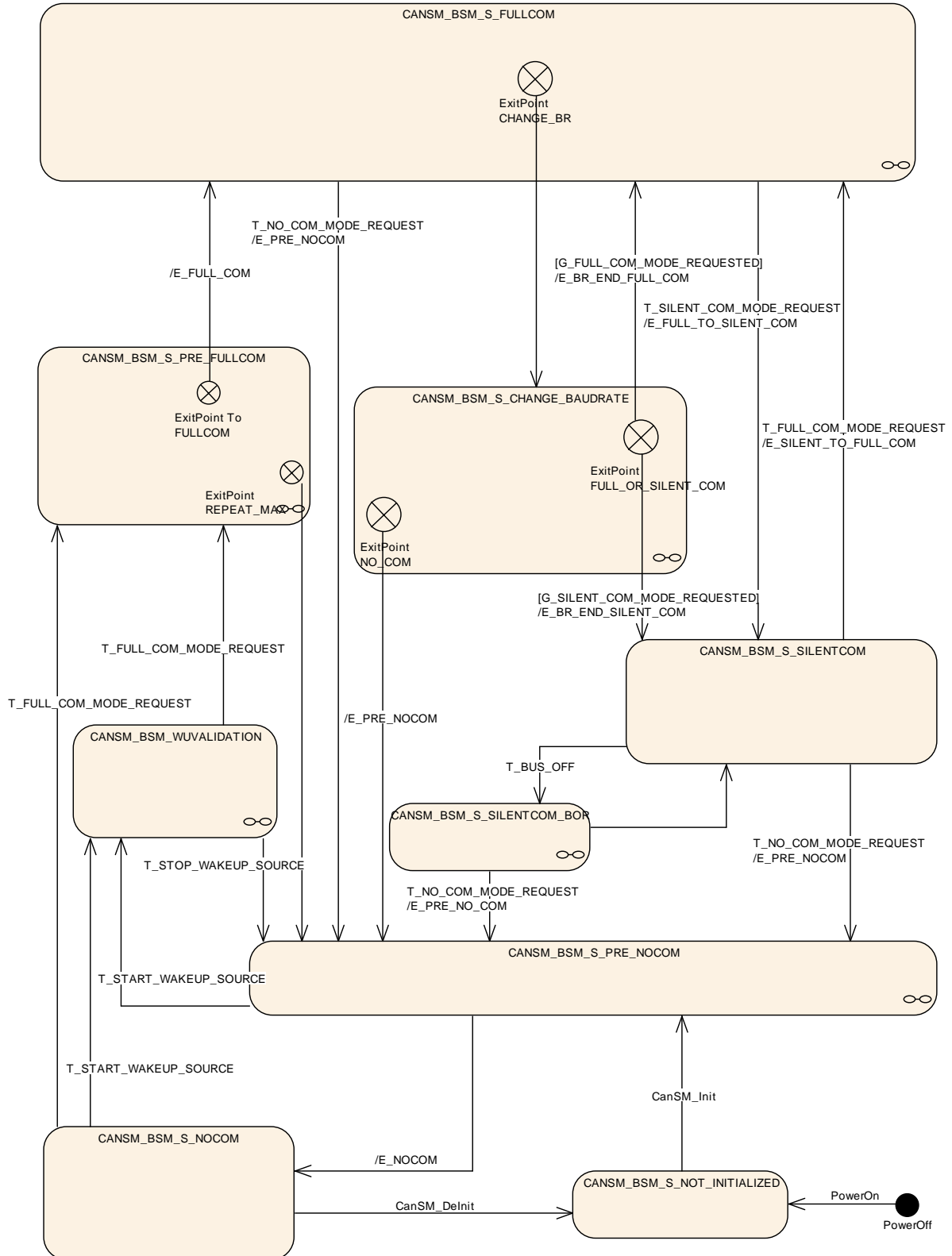


Figure 7-1) shall call at 1<sup>st</sup> place for each configured CAN controller of the CAN

network the API `CanIf_SetPduMode` with the parameters `ControllerId := CanSMControllerId` (ref. to [ECUC\\_CanSM\\_00141](#)) and `PduModeRequest := CANIF_TX_OFFLINE_ACTIVE.`] (SRS\_Can\_01158)

**[SWS\_CanSM\_00435]** [ After considering [SWS\\_CANSM\\_00539](#) and [SWS\\_CanSM\\_00647](#) in context of the effect `E_FULL_COM` of the `CanSM_BSM` state machine (ref. to

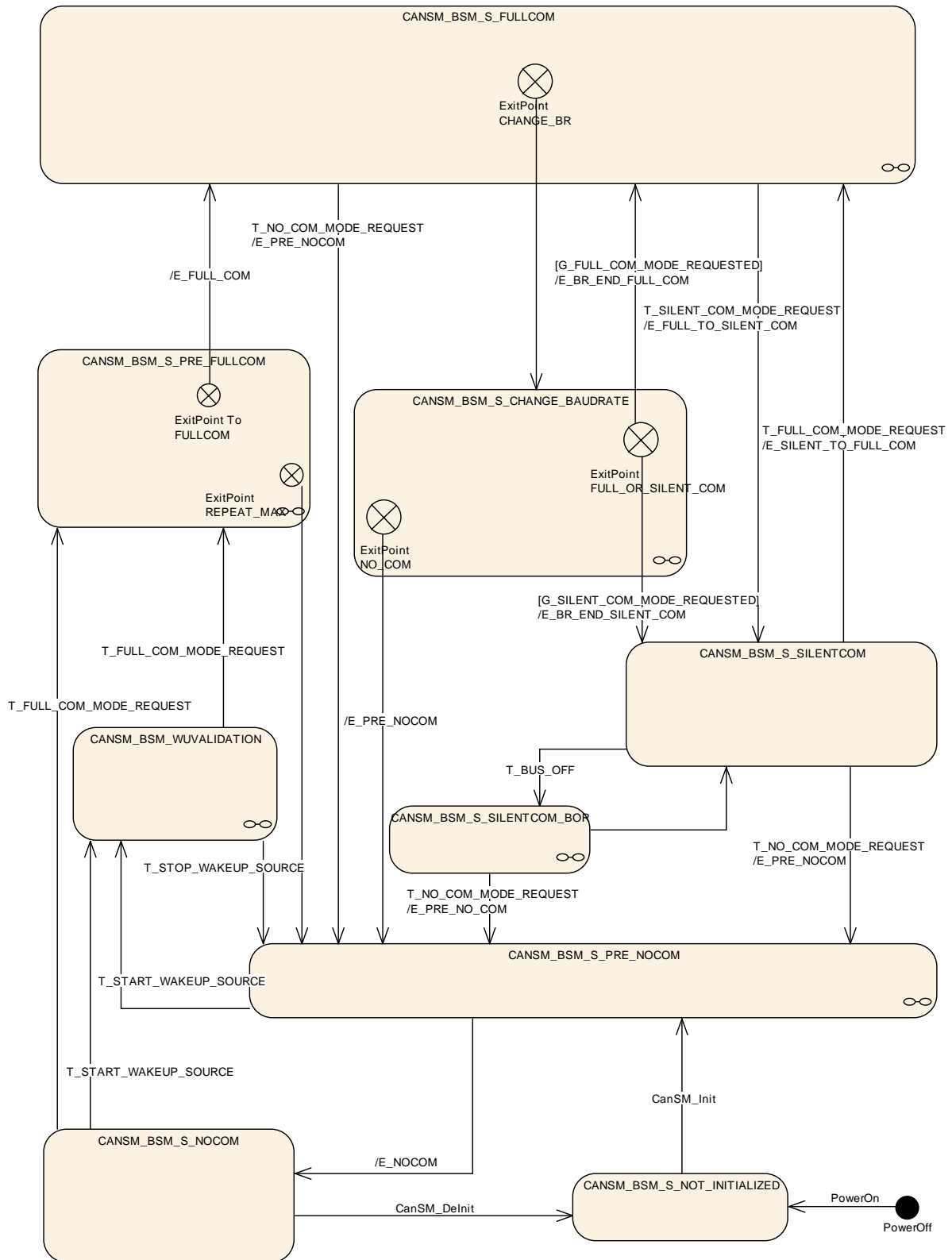


Figure 7-1), the CanSM module shall call the API ComM\_BusSM\_ModeIndication for the corresponding CAN network with the parameters Channel := CanSMComMNetworkHandleRef (ref. to ECUC CanSM\_00161) and ComMode := COMM\_FULL\_COMMUNICATION.

|(SRS\_Can\_01158)

**[SWS\_CanSM\_00540]** [ After considering [SWS\\_CANSM\\_00435](#) in context of the effect `E_FULL_COM` of the `CanSM_BSM` state machine (ref. to Figure 7 1), the `CanSM` module shall call the API `BswM_CanSM_CurrentState` for the corresponding CAN network with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANSM_BSWM_FULL_COMMUNICATION.`] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.15 Effect: `E_FULL_TO_SILENT_COM`

**[SWS\_CanSM\_00434]** [ The effect `E_FULL_TO_SILENT_COM` of the `CanSM_BSM` state machine (ref. to

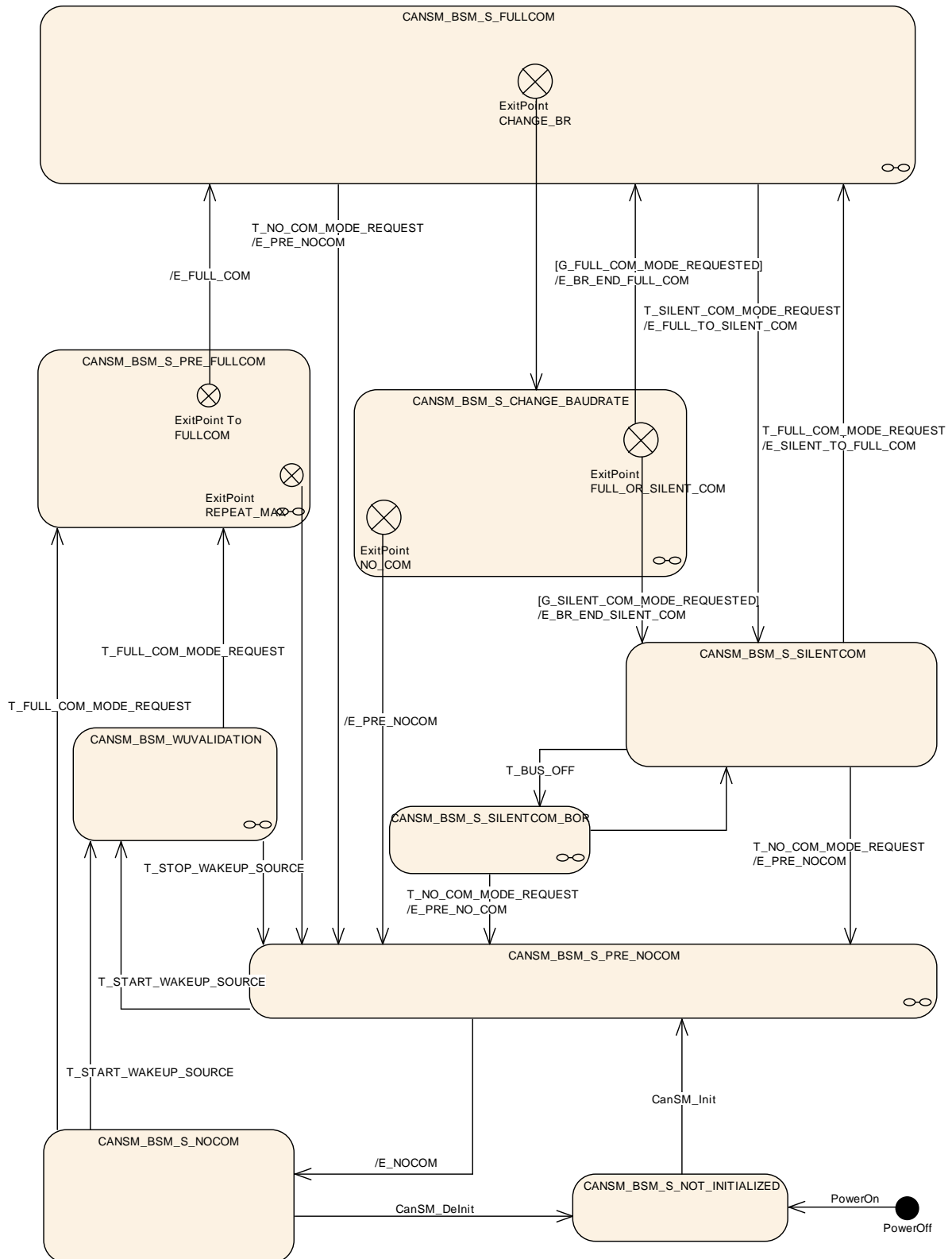


Figure 7-1) shall call at 1<sup>st</sup> place for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANS_M_BSM_S_SILENT_COMMUNICATION.` (SRS\_Can\_01142, SRS\_Can\_01145)

[SWS\_CanSM\_00541] [ The effect E\_FULL\_TO\_SILENT\_COM of the CanSM\_BSM state machine (ref. to

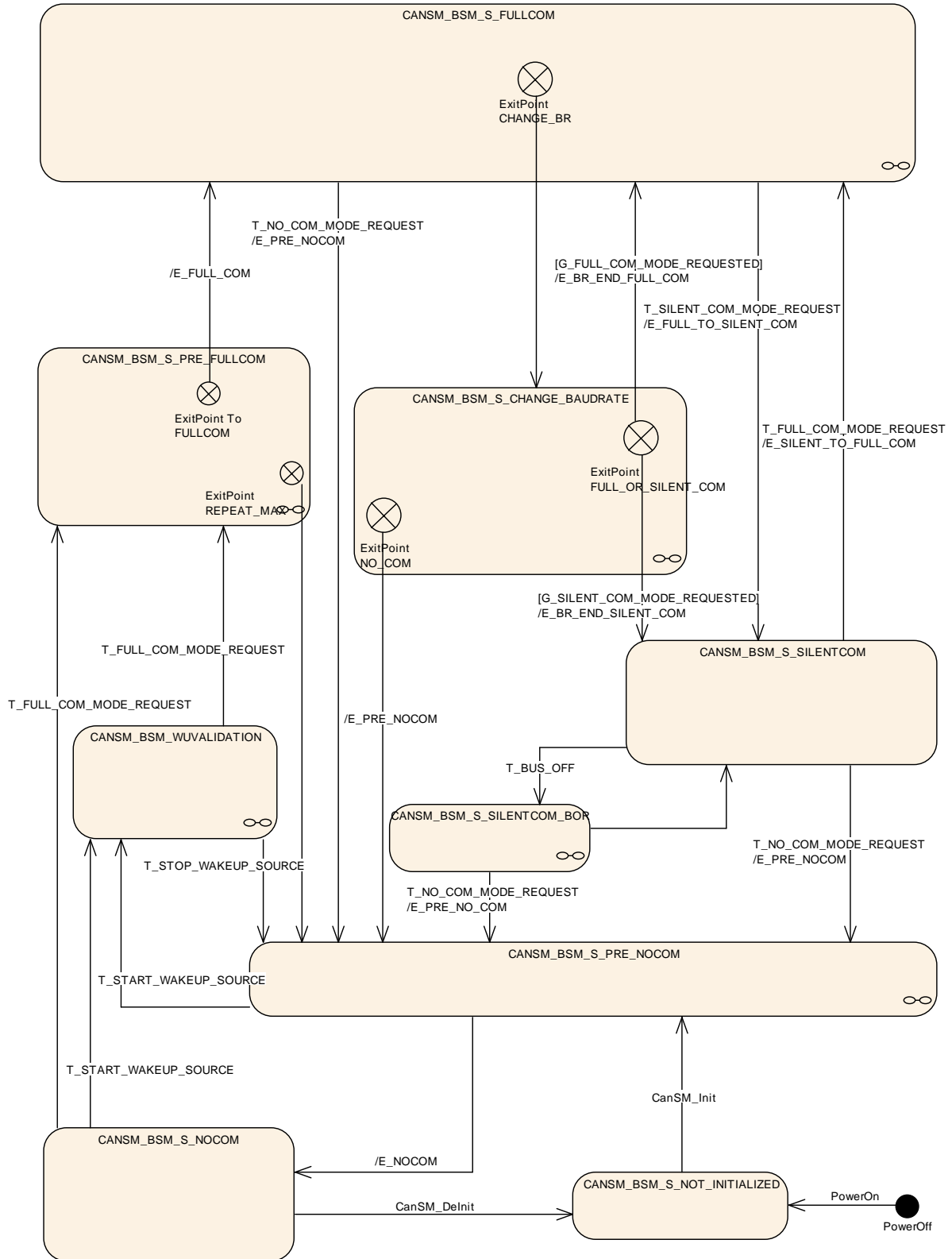


Figure 7-1) shall call at 2<sup>nd</sup> place for each configured CAN controller of the CAN network the API `CanIf_SetPduMode` with the parameters `ControllerId := CanSMControllerId` (ref. to [ECUC CanSM\\_00141](#)) and `PduModeRequest := CANIF_TX_OFFLINE`] (SRS\_Can\_01142, SRS\_Can\_01145)

[SWS\_CanSM\_00538] [ The effect E\_FULL\_TO\_SILENT\_COM of the CanSM\_BSM state machine (ref. to

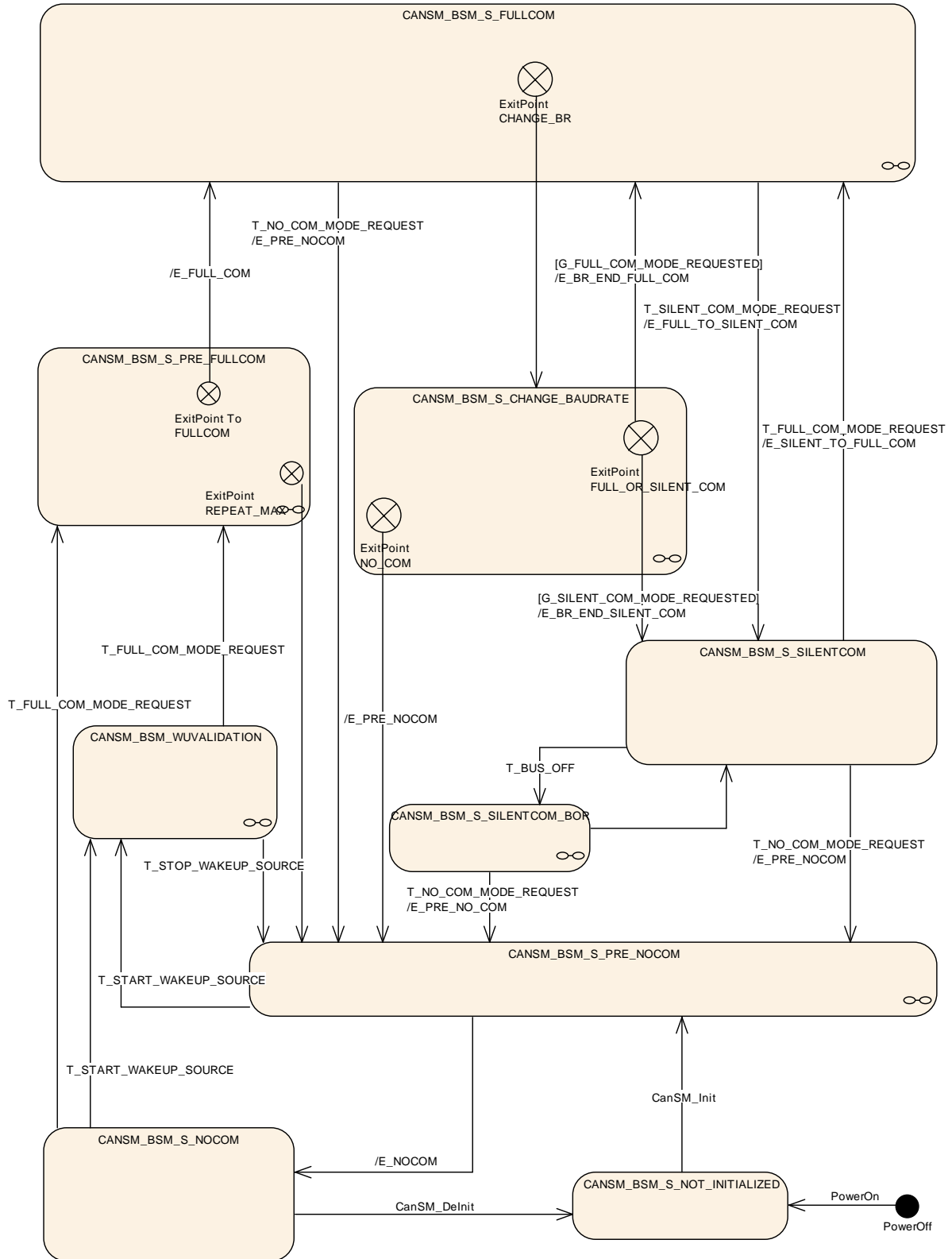


Figure 7-1) shall call at 3<sup>th</sup> place for the corresponding CAN network the API `ComM_BusSM_ModeIndication` with the parameters `Channel := CanSMComMNetworkHandleRef` (ref. to [ECUC\\_CanSM\\_00161](#)) and `ComMode := COMM_SILENT_COMMUNICATION`.] (SRS\_Can\_01142, SRS\_Can\_01145)



**7.2.16 Effect: E\_BR\_END\_FULL\_COM**

[SWS\_CanSM\_00432] | The effect E\_BR\_END\_FULL\_COM of the CanSM\_BSM state machine (ref. to

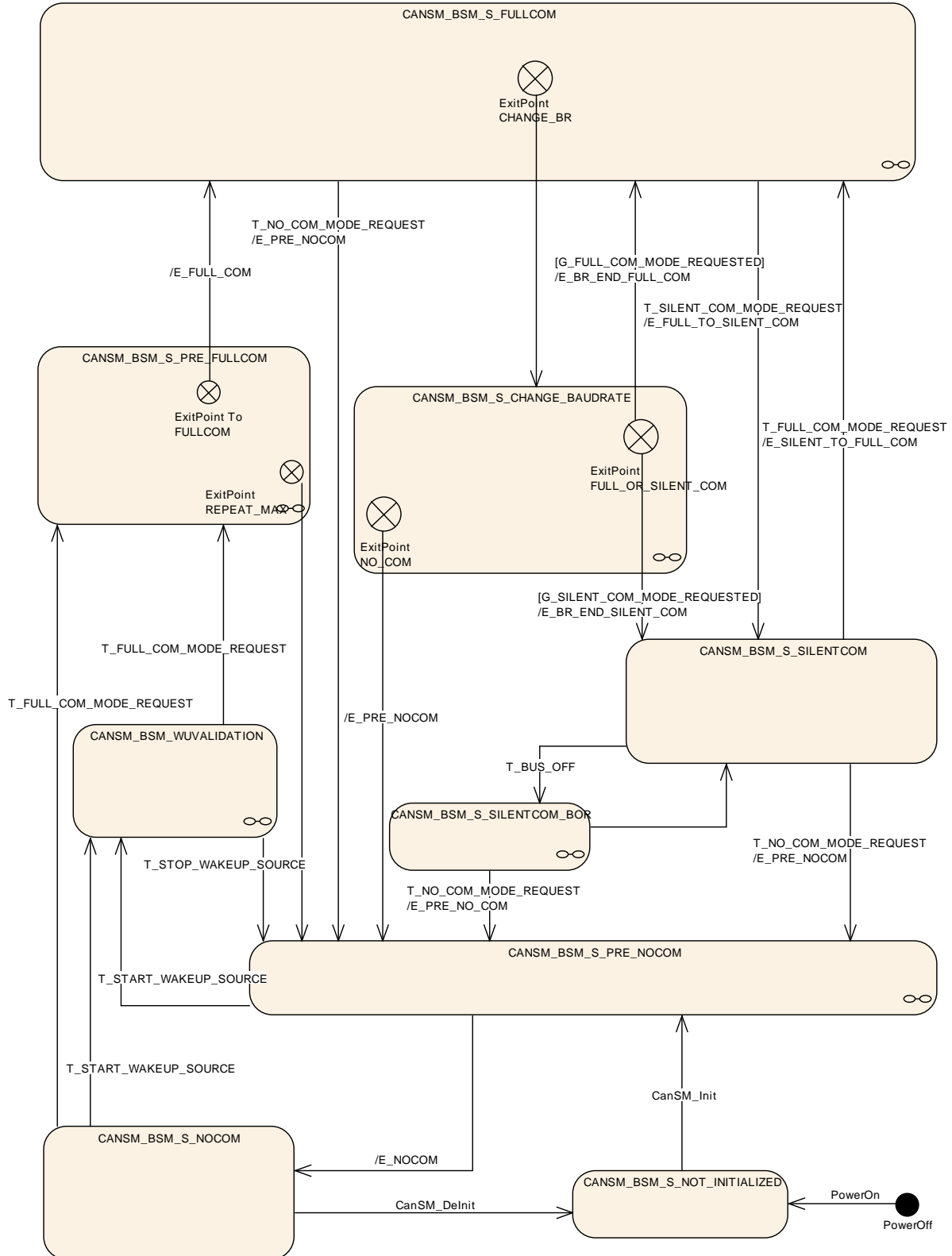


Figure 7-1) shall be the same as E\_FULL\_COM (ref. to chapter 7.2.14).] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.17 Effect: E\_BR\_END\_SILENT\_COM**

[SWS\_CanSM\_00433] [ The effect E\_BR\_END\_SILENT\_COM of the CanSM\_BSM state machine (ref. to

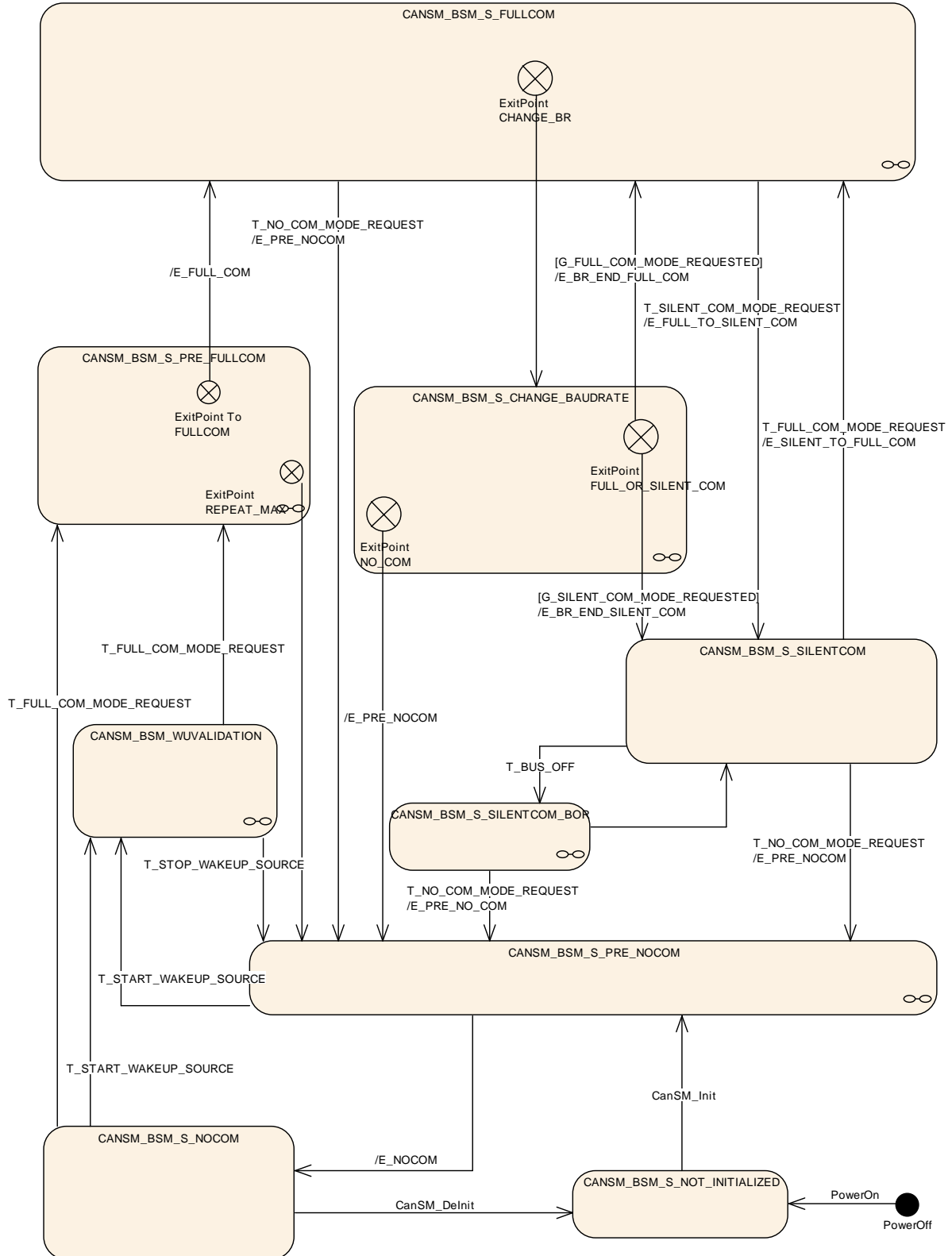


Figure 7-1) shall be the same as E\_FULL\_TO\_SILENT\_COM (ref. to chapter 7.2.15).] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.18 Effect: E\_SILENT\_TO\_FULL\_COM**

[SWS\_CanSM\_00550] [ The effect E\_SILENT\_TO\_FULL\_COM of the CanSM\_BSM state machine (ref. to

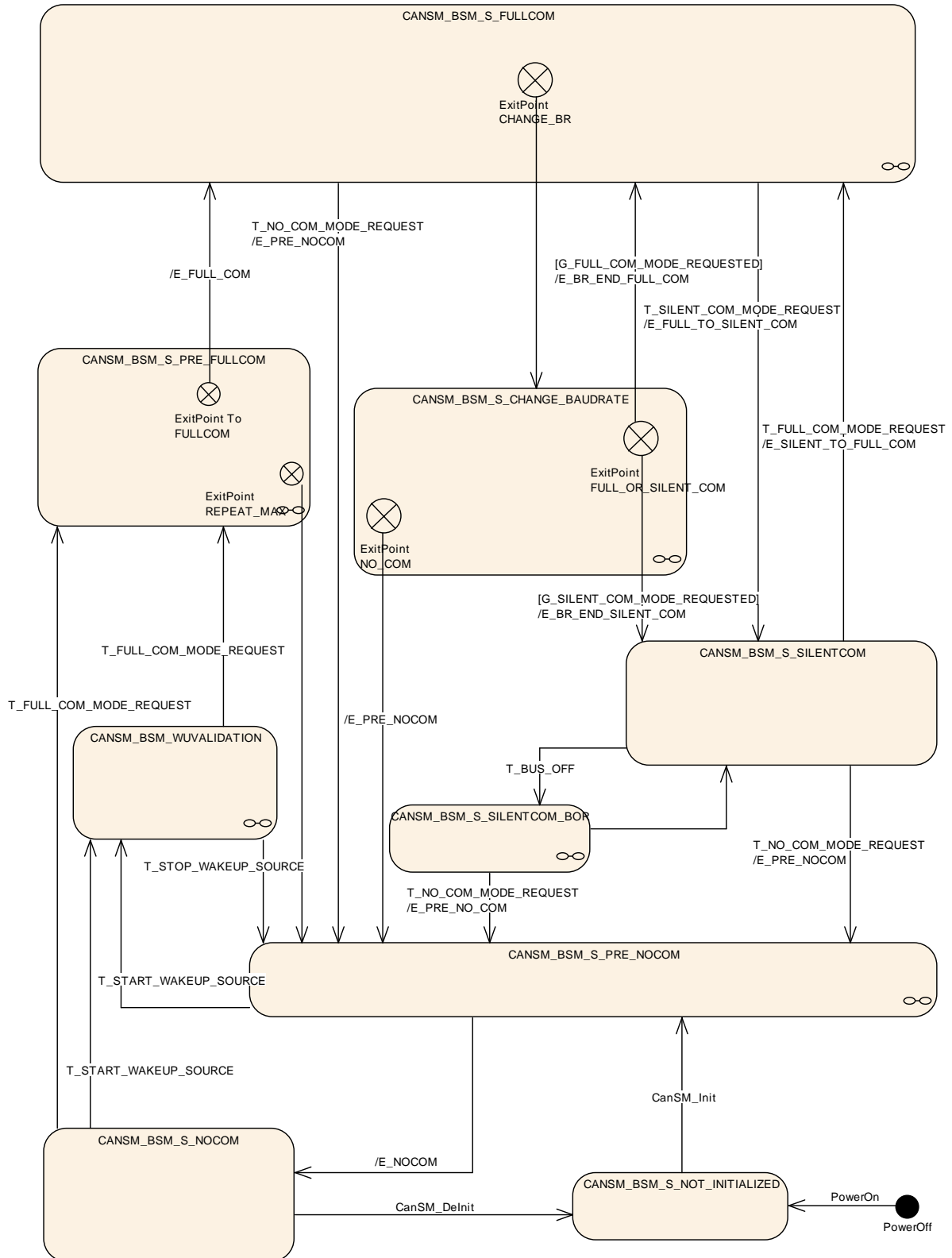
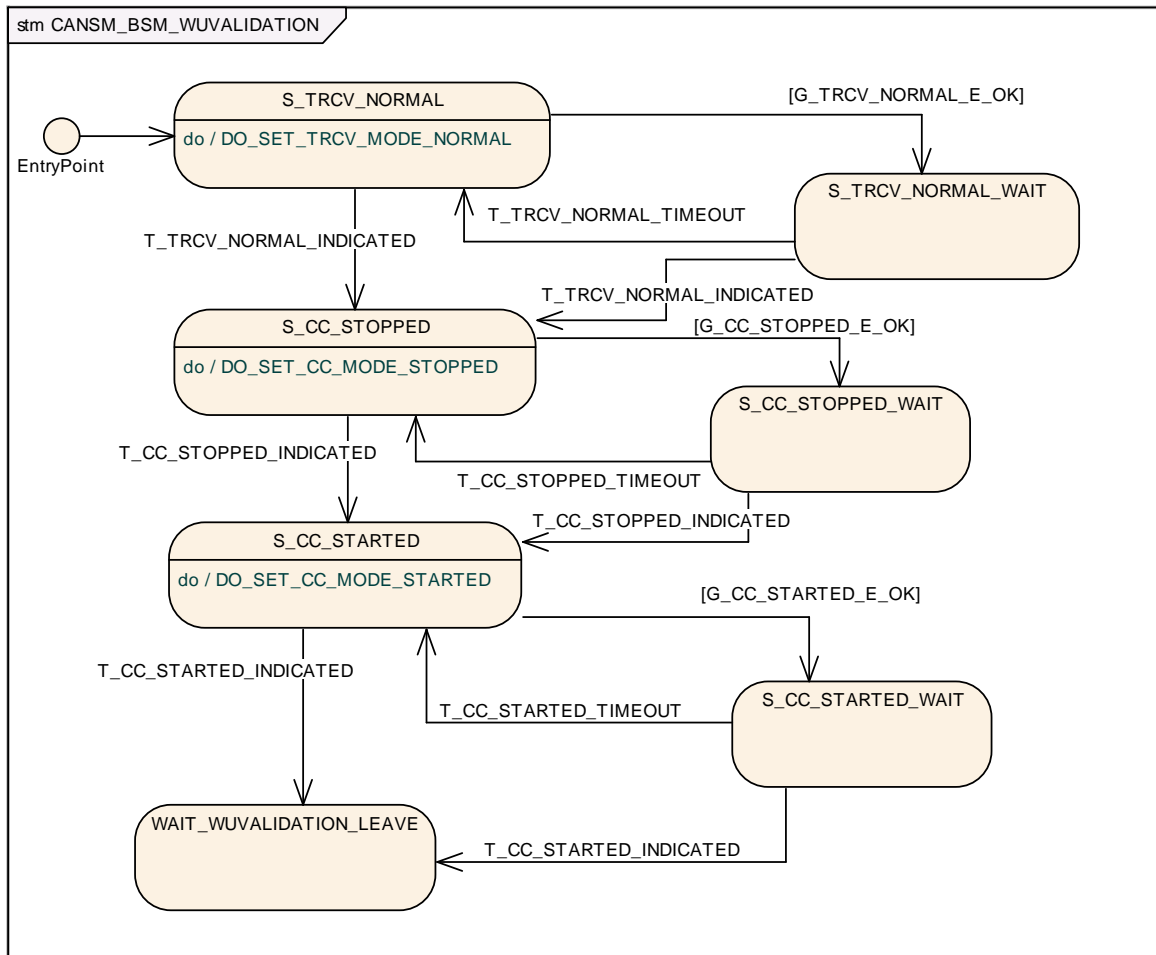


Figure 7-1) shall be the same as E\_FULL\_COM (ref. to chapter 7.2.14).] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19 Sub state machine CANSM\_BSM\_WUVALIDATION**



**Figure 7-2: CANSM\_BSM\_WUVALIDATION, sub state machine of CANSM\_BSM**

**7.2.19.1 State operation to do in: S\_TRCV\_NORMAL**

[SWS\_CanSM\_00623] If for the CAN network a CAN Transceiver is configured (ref. to [ECUC\\_CanSM\\_00137](#)), then as long the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) is in the state S\_TRCV\_NORMAL, the CanSM module shall operate the do action DO\_SET\_TRCV\_MODE\_NORMAL and therefore repeat for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) the API request CanIf\_SetTrcvMode (ref. to chapter 8.5.1) with TransceiverMode equal to CANTRCV\_TRCVMODE\_NORMAL.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.2 Guarding condition G\_TRCV\_NORMAL\_E\_OK**

[SWS\_CanSM\_00624] The guarding condition G\_TRCV\_NORMAL\_E\_OK of the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) shall be passed, if the API call of [SWS\\_CanSM\\_00483](#) has returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.3 Trigger: T\_TRCV\_NORMAL\_INDICATED**

[SWS\_CanSM\_00625] If CanSM module has got the CANTRCV\_TRCVMODE\_NORMAL mode indication (ref. to [SWS\\_CanSM\\_00399](#)) for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) after the respective request (ref. to [SWS\\_CanSM\\_00623](#)), this shall trigger the sub state machine machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) of the CAN network with T\_TRCV\_NORMAL\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.4 Trigger: T\_TRCV\_NORMAL\_TIMEOUT**

[SWS\_CanSM\_00626] After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for the supposed transceiver normal indication (ref. to [SWS\\_CanSM\\_00625](#)), this condition shall trigger the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) of the respective network with T\_TRCV\_NORMAL\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.5 State operation to do in: S\_CC\_STOPPED**

[SWS\_CanSM\_00627] As long the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) is in the state S\_CC\_STOPPED, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STOPPED and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STOPPED, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.6 Guarding condition: G\_CC\_STOPPED\_OK**

[SWS\_CanSM\_00628] The guarding condition G\_CC\_STOPPED\_OK of the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) shall be passed, if all API calls of [SWS\\_CanSM\\_00627](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.7 Trigger: T\_CC\_STOPPED\_INDICATED**

[SWS\_CanSM\_00629] If the CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to stop the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00627](#)), this shall trigger the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) of the CAN network with T\_CC\_STOPPED\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.8 Trigger: T\_CC\_STOPPED\_TIMEOUT**

[SWS\_CanSM\_00630] After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller stopped mode indications (ref. to [SWS\\_CanSM\\_00629](#)), this condition shall trigger the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) of the respective network with T\_CC\_STOPPED\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.9 State operation to do in: S\_CC\_STARTED**

[SWS\_CanSM\_00631] As long the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) is in the state S\_CC\_STARTED, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STARTED and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STARTED, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.10 Guarding condition: G\_CC\_STARTED\_E\_OK**

[SWS\_CanSM\_00632] The guarding condition G\_CC\_STARTED\_OK of the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) shall be passed, if all API calls of [SWS\\_CanSM\\_00631](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

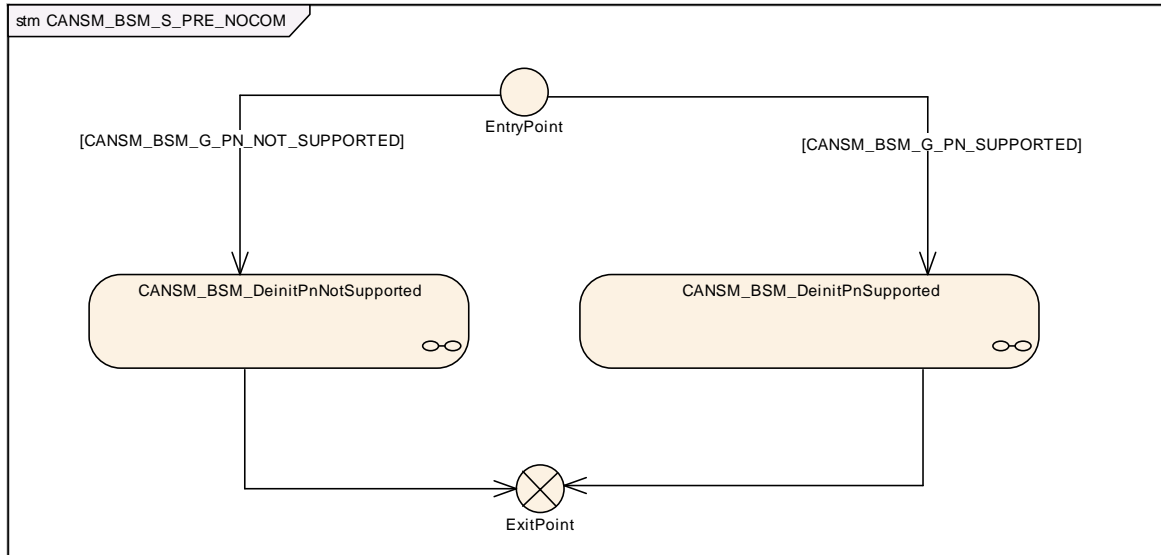
**7.2.19.11 Trigger: T\_CC\_STARTED\_INDICATED**

[SWS\_CanSM\_00633] If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to start the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00631](#)), this shall trigger the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) of the CAN network with T\_CC\_STARTED\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.19.12 Trigger: T\_CC\_STARTED\_TIMEOUT**

[SWS\_CanSM\_00634] After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller started mode indications (ref. to [SWS\\_CanSM\\_00633](#)), this condition shall trigger the sub state machine CANSM\_BSM\_WUVALIDATION (ref. to Figure 7-2) of the respective network with T\_CC\_STARTED\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20 Sub state machine: CANSM\_BSM\_S\_PRE\_NOCOM**



**Figure 7-3: CANSM\_BSM\_S\_PRE\_NOCOM, sub state machine of CANSM\_BSM**

**7.2.20.1 Guarding condition: CANSM\_BSM\_G\_PN\_NOT\_SUPPORTED**

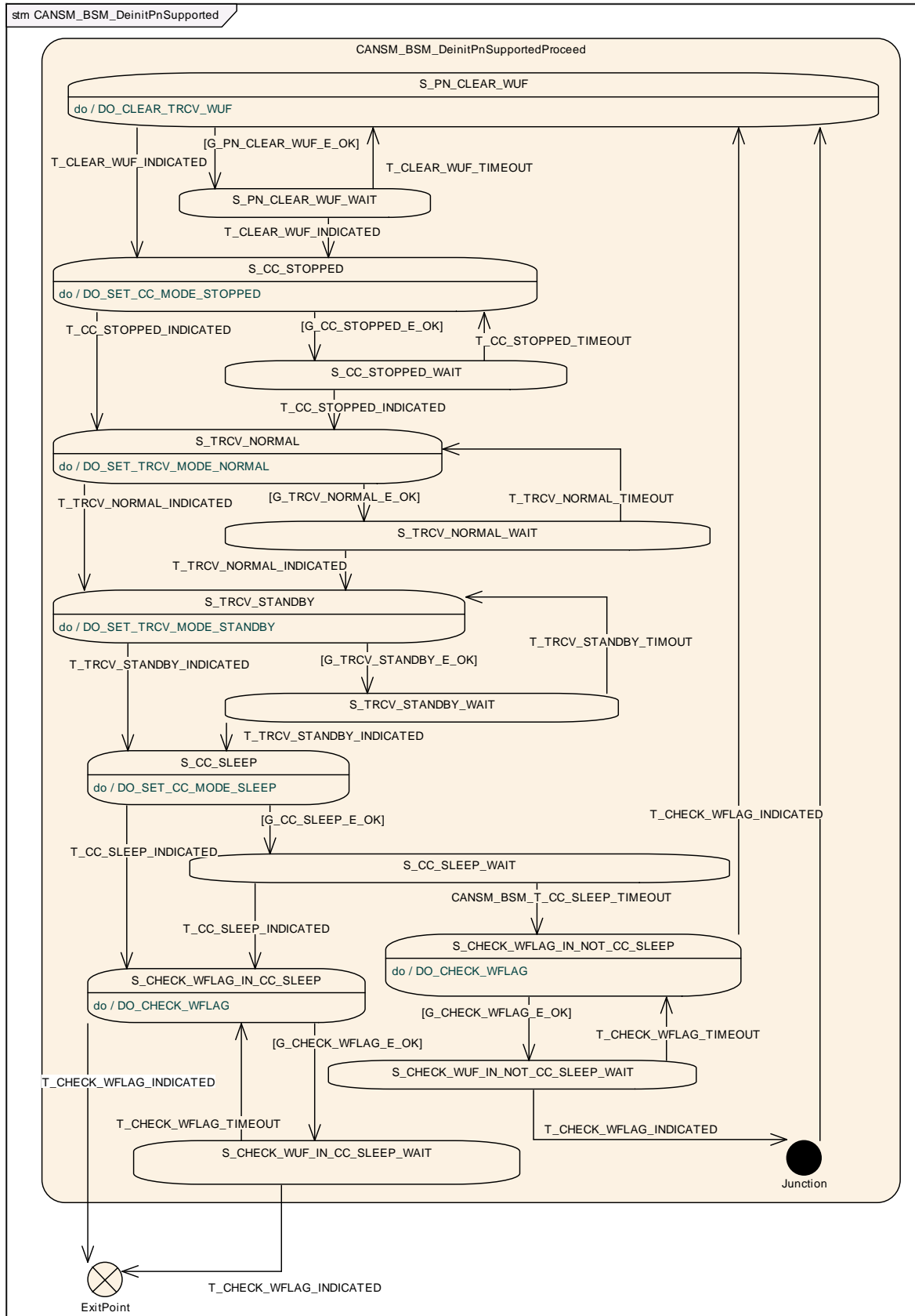
**[SWS\_CanSM\_00436]** [ The guarding condition CANSM\_BSM\_G\_PN\_NOT\_SUPPORTED of the sub state machine CANSM\_BSM\_S\_PRE\_NO\_COM (ref. to Figure 7-3) shall evaluate, if the configuration parameter CanTrcvPnEnabled (ref. to [9], ECUC\_CanTrcv\_00172) is FALSE, which is available via the reference CanSMTransceiverId (ref. to [ECUC\\_CanSM\\_00137](#)) or if no CanSMTransceiverId is configured at all.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.2 Guarding condition: CANSM\_BSM\_G\_PN\_SUPPORTED**

**[SWS\_CanSM\_00437]** [ The guarding condition CANSM\_BSM\_G\_PN\_SUPPORTED of the sub state machine CANSM\_BSM\_S\_PRE\_NO\_COM (ref. to Figure 7-3) shall evaluate, if a CanSMTransceiverId (ref. to [ECUC\\_CanSM\\_00137](#)) is configured and if the configuration parameter CanTrcvPnEnabled (ref. to [9], ECUC\_CanTrcv\_00172) is TRUE, which is available via the reference CanSMTransceiverId (ref. to [ECUC\\_CanSM\\_00137](#)).] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.20.3 Sub state machine: CANSM\_BSM\_DeInitPnSupported





**Figure 7-4: CANSM\_BSM\_DeinitPnSupported, sub state machine of CANSM\_BSM\_S\_PRE\_NOCOM**

#### 7.2.20.3.1 State operation to do in: S\_PN\_CLEAR\_WUF

**[SWS\_CanSM\_00438]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

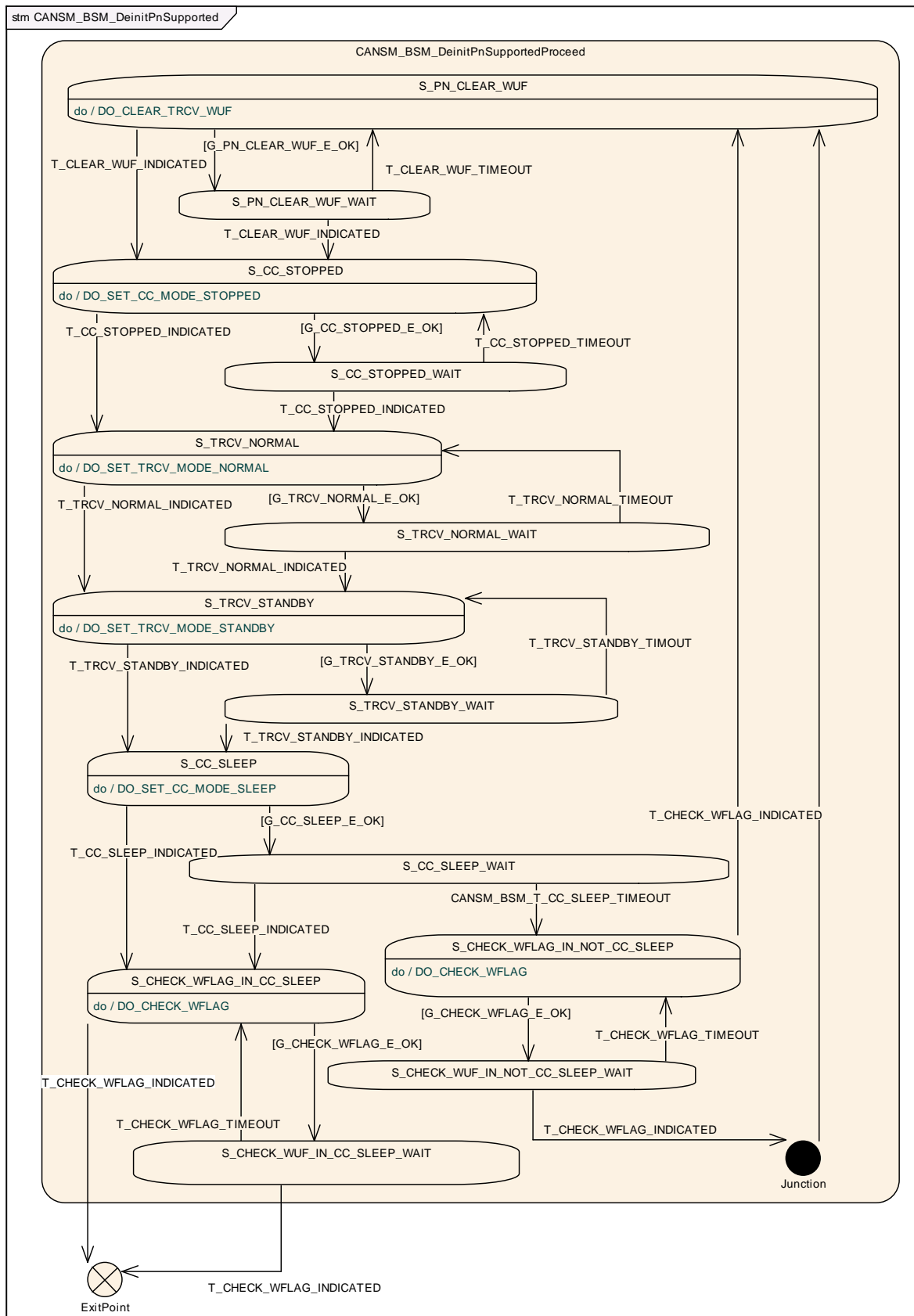


Figure 7-4) is in the state `S_PN_CLEAR_WUF`, the CanSM module operate the do action `DO_CLEAR_TRCV_WUF` and therefore repeat the API request `CanIf_ClrTrcvWufFlag` (ref. to chapter 8.5.1) and use the configured Transceiver

(ref. to [ECUC\\_CanSM\\_00137](#)) as API function parameter.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.3.2 Guarding condition: G\_PN\_CLEAR\_WUF\_E\_OK**

**[SWS\_CanSM\_00439]** [ The guarding condition G\_PN\_CLEAR\_WUF\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

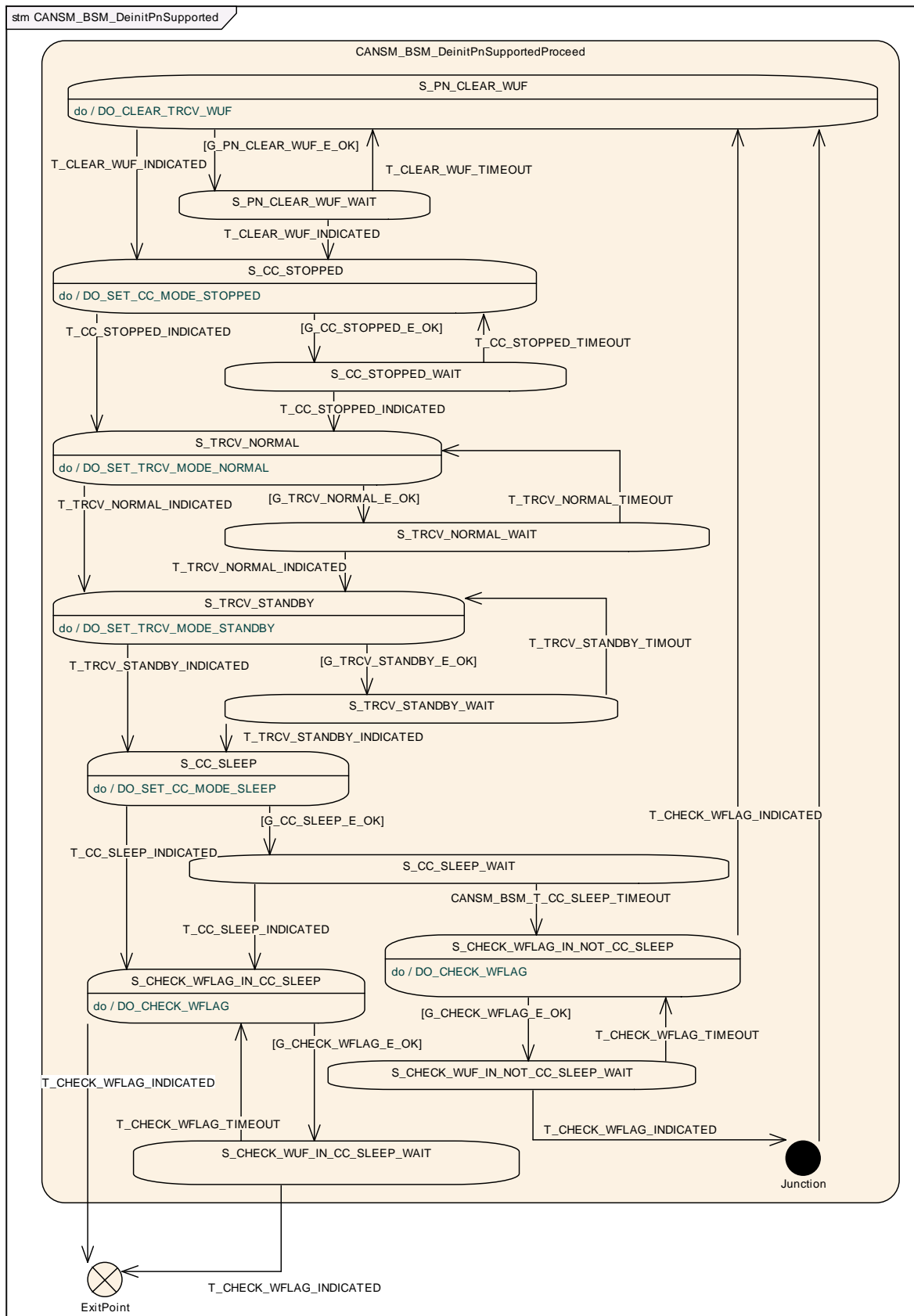


Figure 7-4) shall be passed, if the API call of [SWS CanSM 00438](#) has returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

### 7.2.20.3.3 Trigger: T\_CLEAR\_WUF\_INDICATED

**[SWS\_CanSM\_00440]** [ The callback function `CanSM_ClearTrcvWufFlagIndication` (ref. to [SWS\\_CanSM\\_00413](#)) shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to

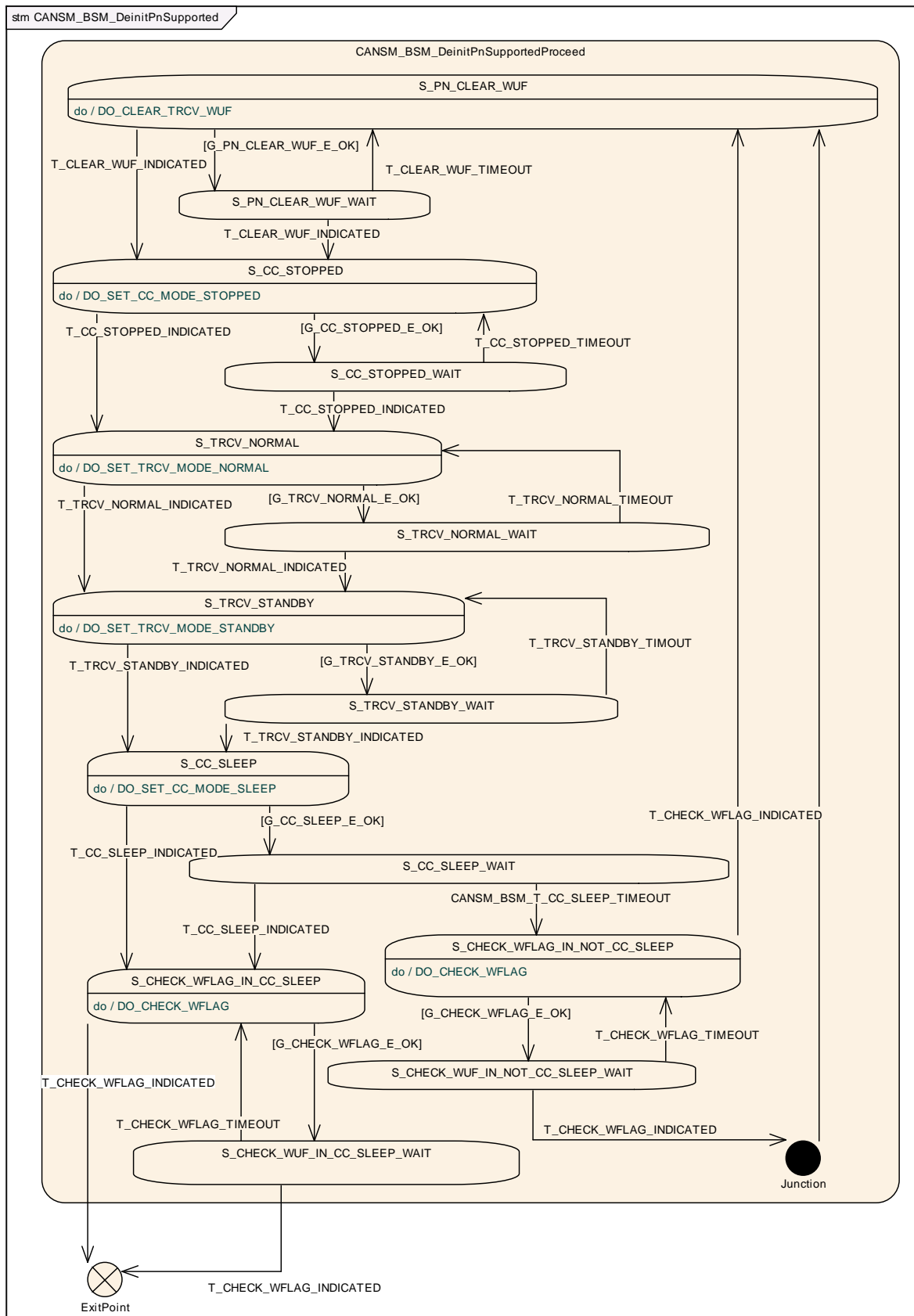


Figure 7-4) of the CAN network with T\_CLEAR\_WUF\_INDICATED, if the function parameter Transceiver of CanSM\_ClearTrcvWuflFlagIndication matches to

the configured CAN Transceiver (ref. to [ECUC\\_CanSM\\_00137](#)) of the CAN network.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.3.4 Trigger: T\_CLEAR\_WUF\_TIMEOUT

**[SWS\_CanSM\_00443]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for the callback function `CanSM_ClearTrcvWufFlagIndication` (ref. to [SWS\\_CanSM\\_00440](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to



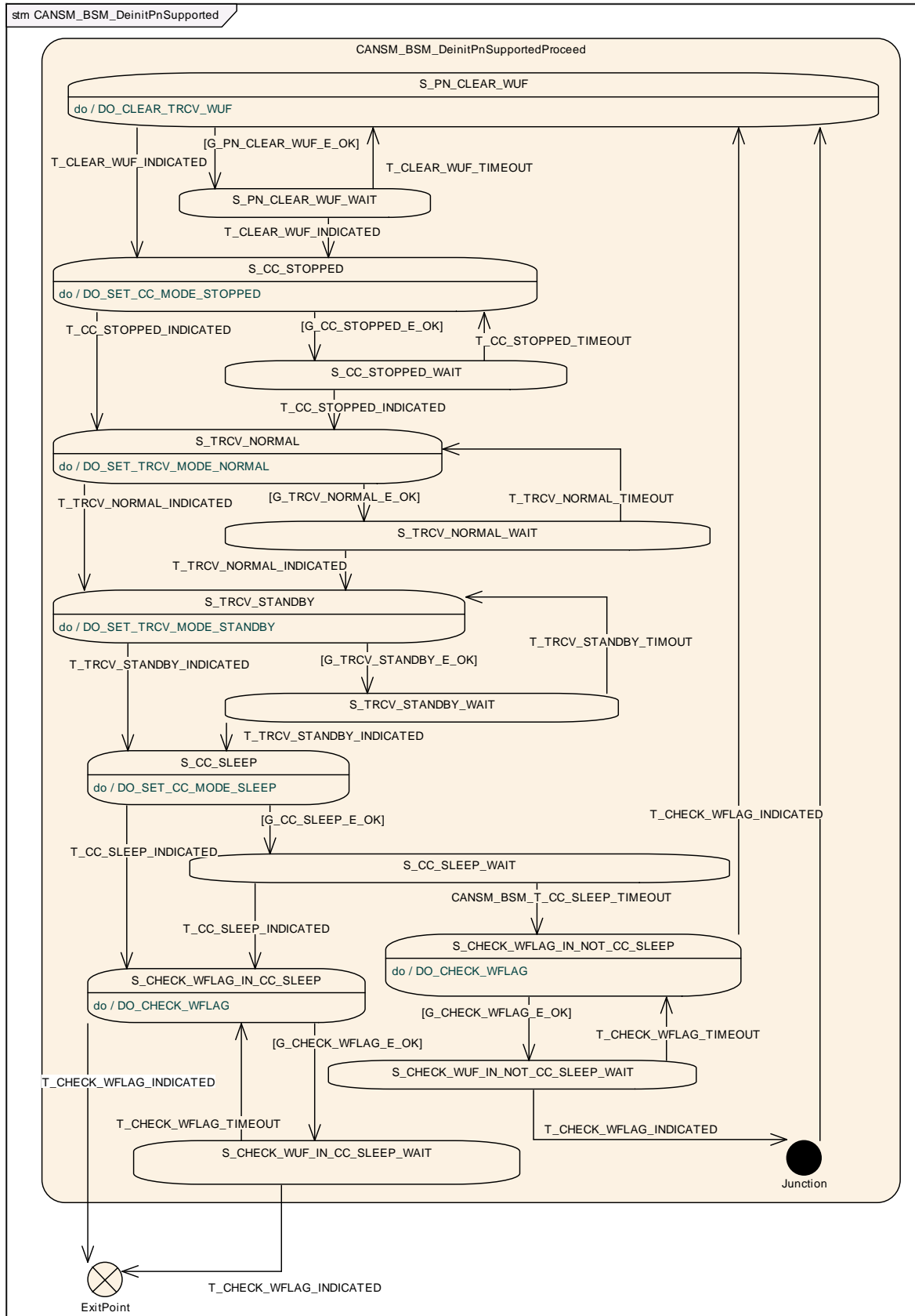


Figure 7-4) of the respective network with  
T\_CLEAR\_WUF\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.3.5 State operation to do in: S\_CC\_STOPPED

**[SWS\_CanSM\_00441]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

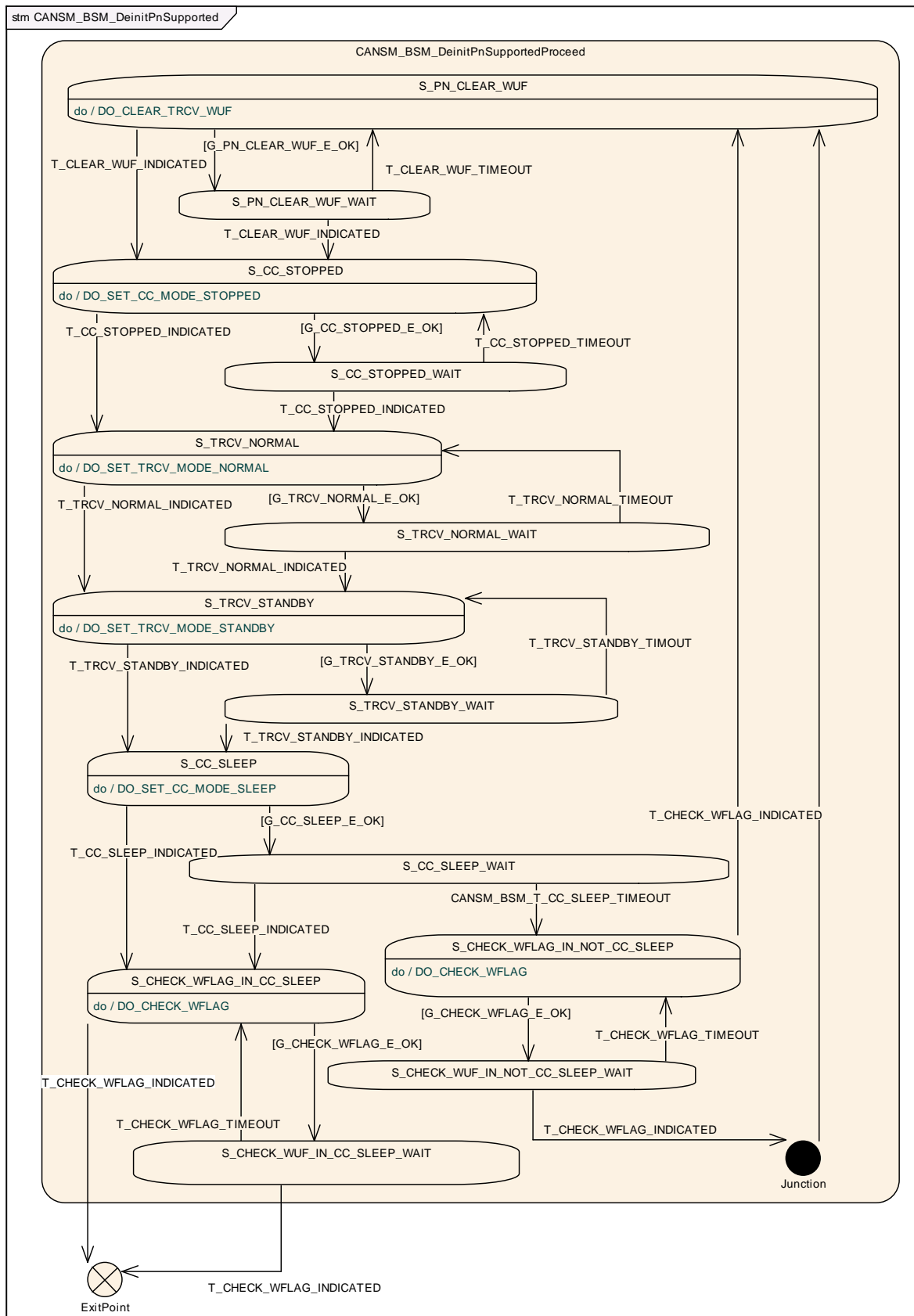


Figure 7-4) is in the state `S_CC_STOPPED`, the CanSM module shall operate the do action `DO_SET_CC_MODE_STOPPED` and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC CanSM 00141](#)) the API request

CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STOPPED, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.3.6 Guarding condition: G\_CC\_STOPPED\_E\_OK**

**[SWS\_CanSM\_00442]** | The guarding condition G\_CC\_STOPPED\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

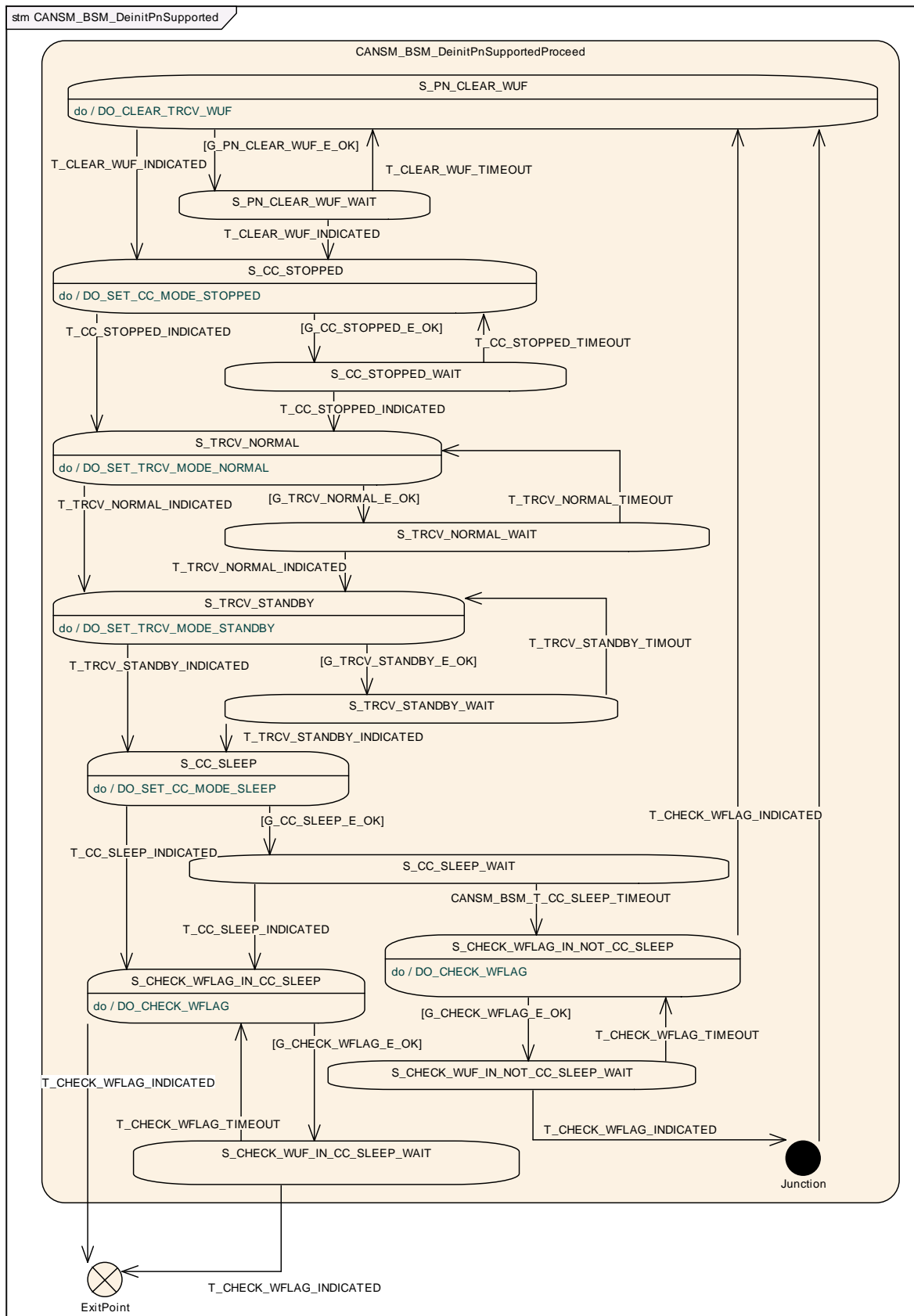


Figure 7-4) shall be passed, if all API calls of [SWS CanSM 00441](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.7 Trigger: T\_CC\_STOPPED\_INDICATED**

**[SWS\_CanSM\_00444]** [ If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to stop the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00442](#)), this shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to

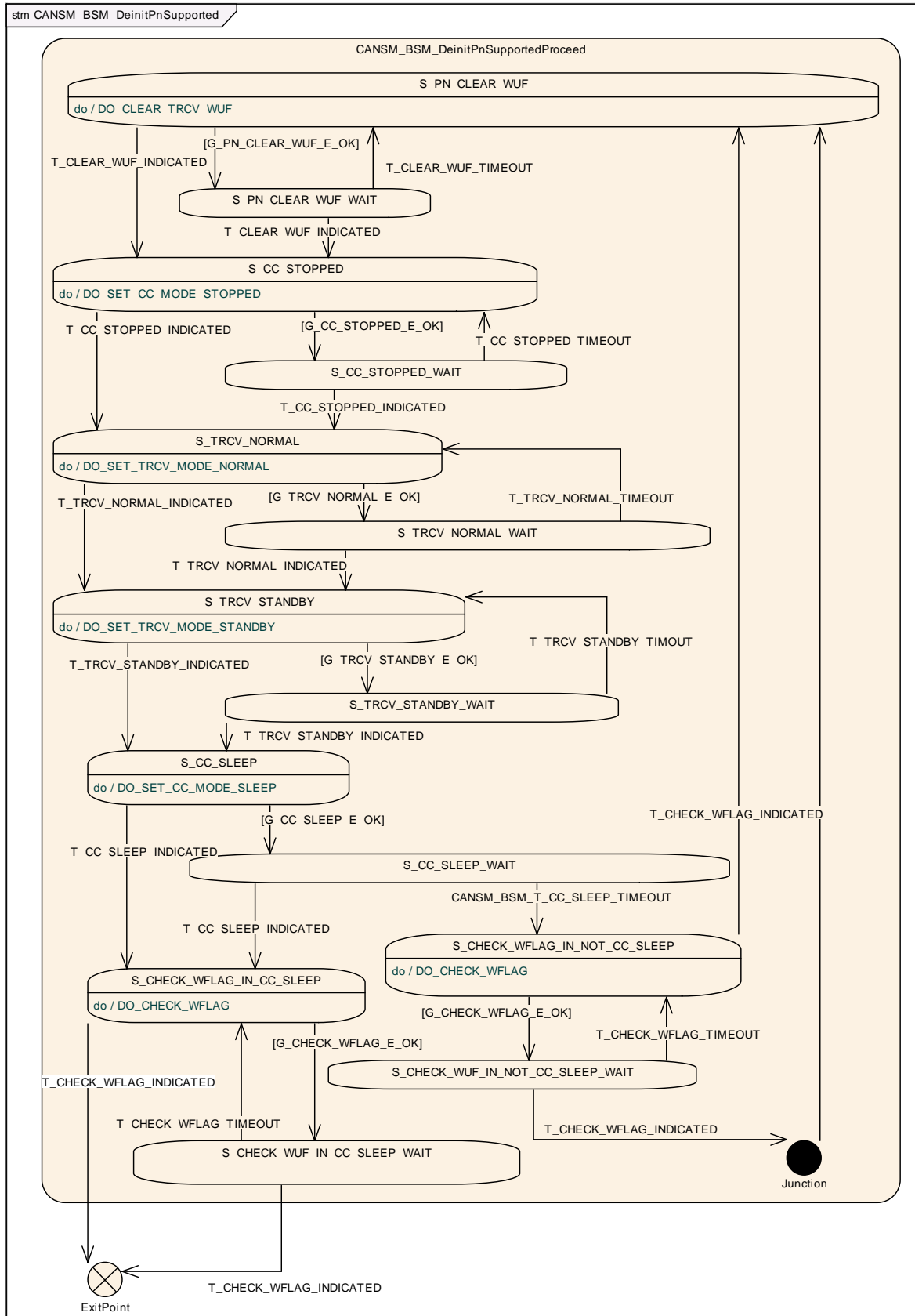


Figure 7-4) of the CAN network with  
T\_CC\_STOPPED\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.8 Trigger: T\_CC\_STOPPED\_TIMEOUT**

**[SWS\_CanSM\_00445]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller stopped mode indications (ref. to [SWS\\_CanSM\\_00444](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to



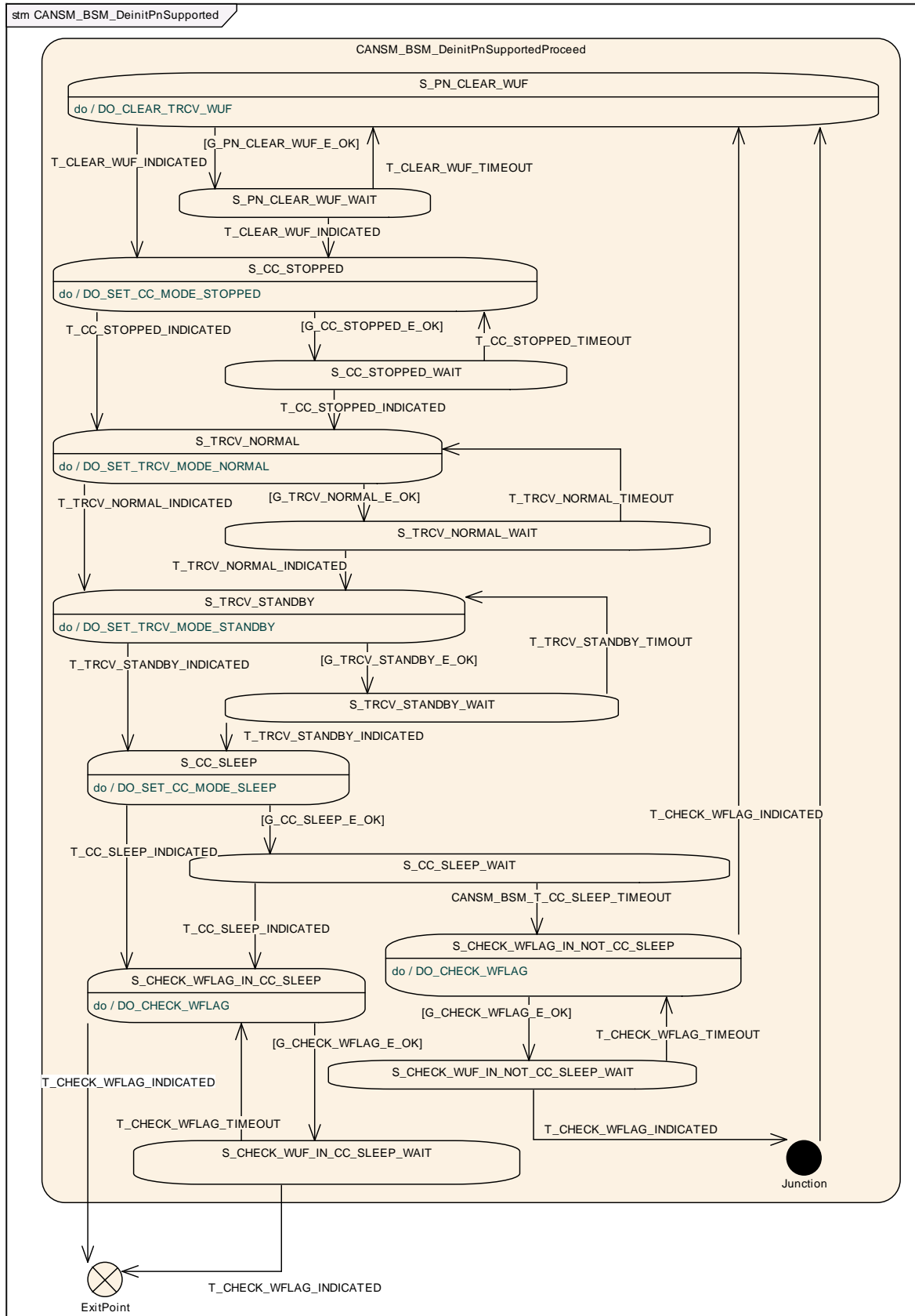


Figure 7-4) of the respective network with T\_CC\_STOPPED\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.3.9 State operation to do in: S\_TRCV\_NORMAL

[SWS\_CanSM\_00446] [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

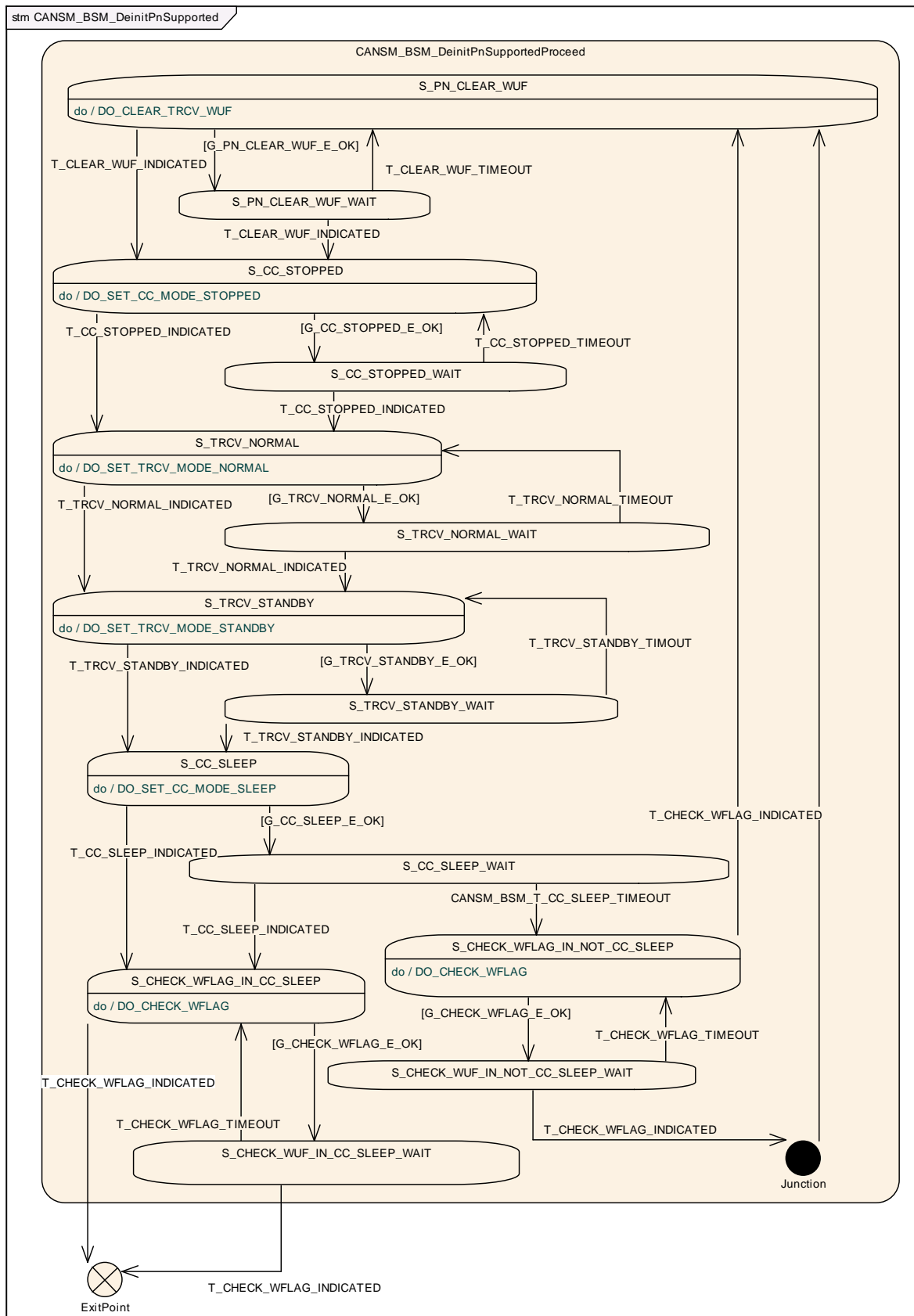


Figure 7-4) is in the state S\_TRCV\_NORMAL, the CanSM module shall operate the do action DO\_SET\_TRCV\_MODE\_NORMAL and therefore repeat for the configured CAN Transceiver of the CAN network (ref. to [ECUC CanSM\\_00137](#)) the API request

CanIf\_SetTrcvMode (ref. to chapter 8.5.1) with TransceiverMode equal to CANTRCV\_TRCVMODE\_NORMAL.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.3.10 Guarding condition: G\_TRCV\_NORMAL\_E\_OK**

**[SWS\_CanSM\_00447]** [ The guarding condition G\_TRCV\_NORMAL\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

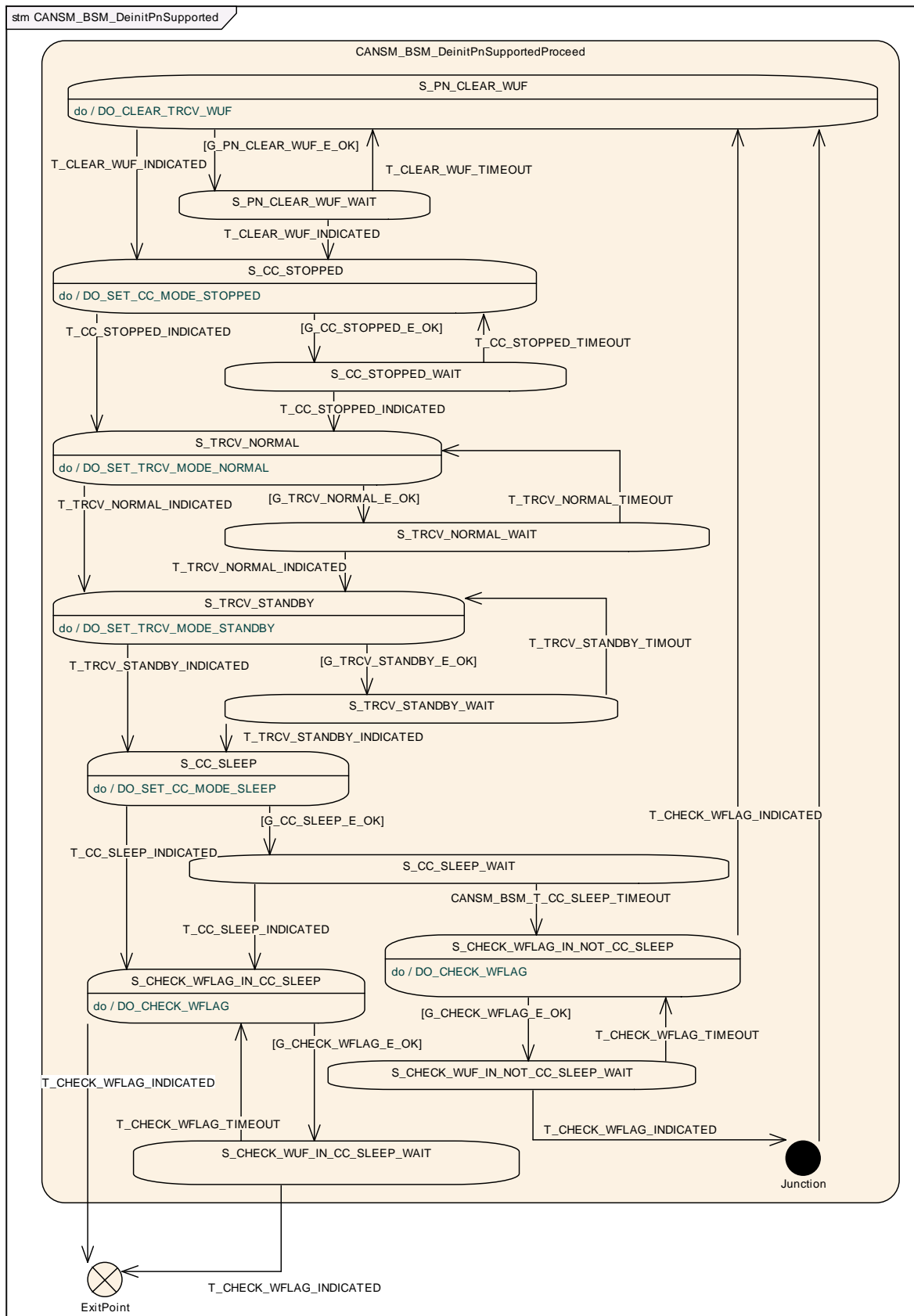


Figure 7-4) shall be passed, if the API call of [SWS CanSM 00446](#) has returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.11 Trigger: T\_TRCV\_NORMAL\_INDICATED**

**[SWS\_CanSM\_00448]** [ If CanSM module has got the CANTRCV\_TRCVMODE\_NORMAL mode indication (ref. to [SWS\\_CanSM\\_00399](#)) for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) after the respective request (ref. to [SWS\\_CanSM\\_00446](#)), this shall trigger the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

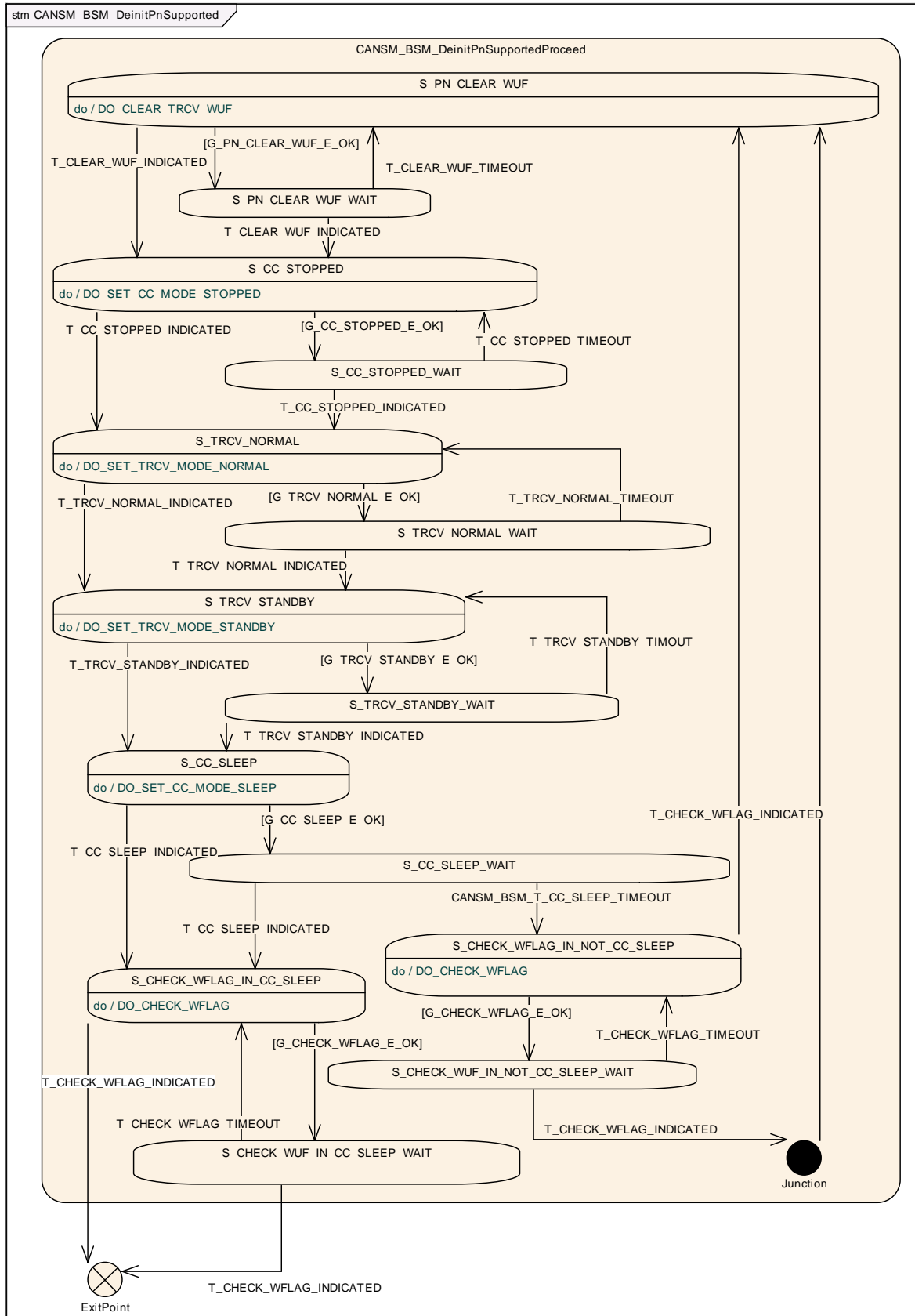


Figure 7-4) of the CAN network with  
T\_TRCV\_NORMAL\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.12 Trigger: T\_TRCV\_NORMAL\_TIMEOUT**

**[SWS\_CanSM\_00449]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for the supposed transceiver normal indication (ref. to [SWS\\_CanSM\\_00448](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to



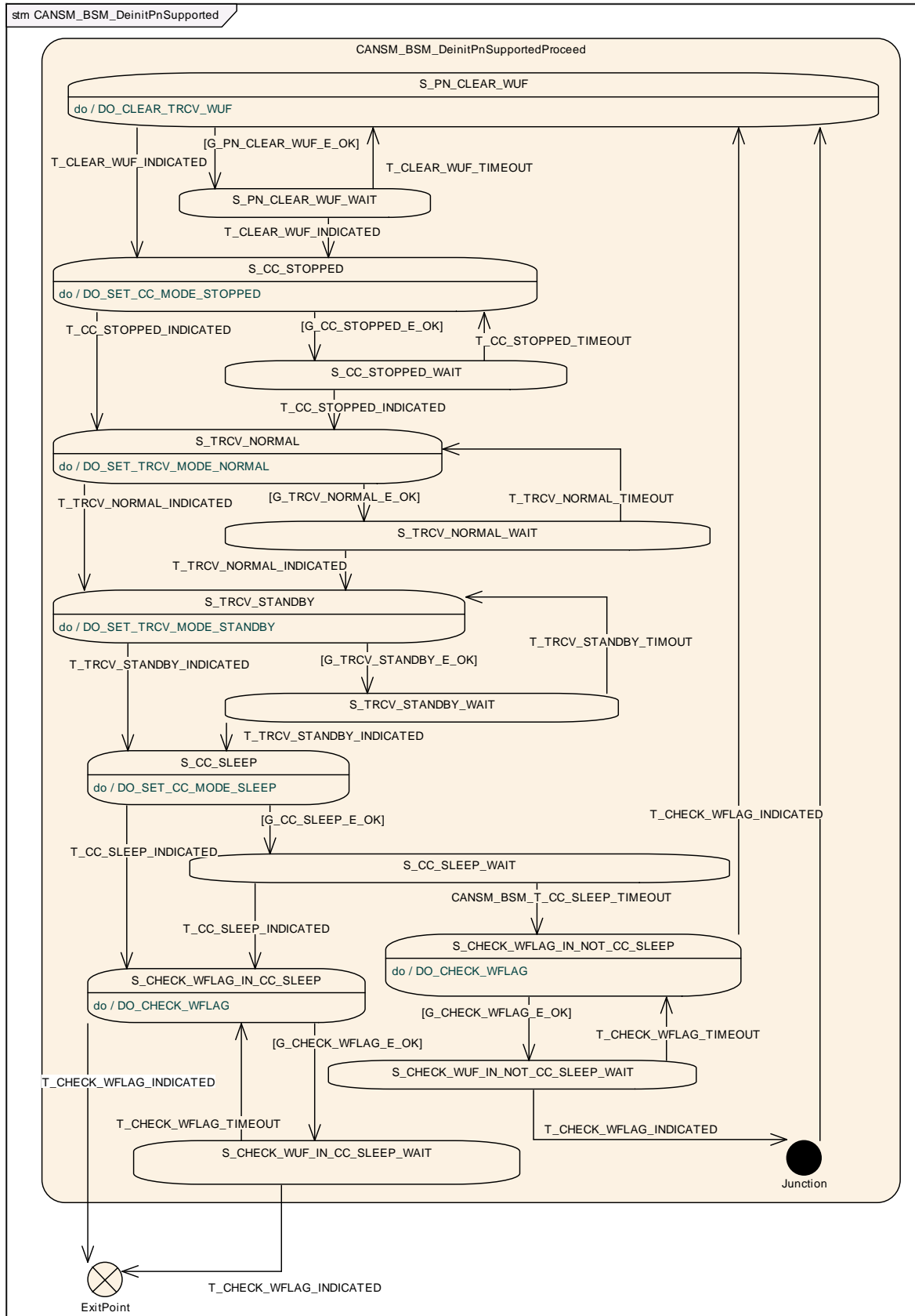


Figure 7-4) of the respective network with  
T\_TRCV\_NORMAL\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.13 State operation to do in: S\_TRCV\_STANDBY**

**[SWS\_CanSM\_00450]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

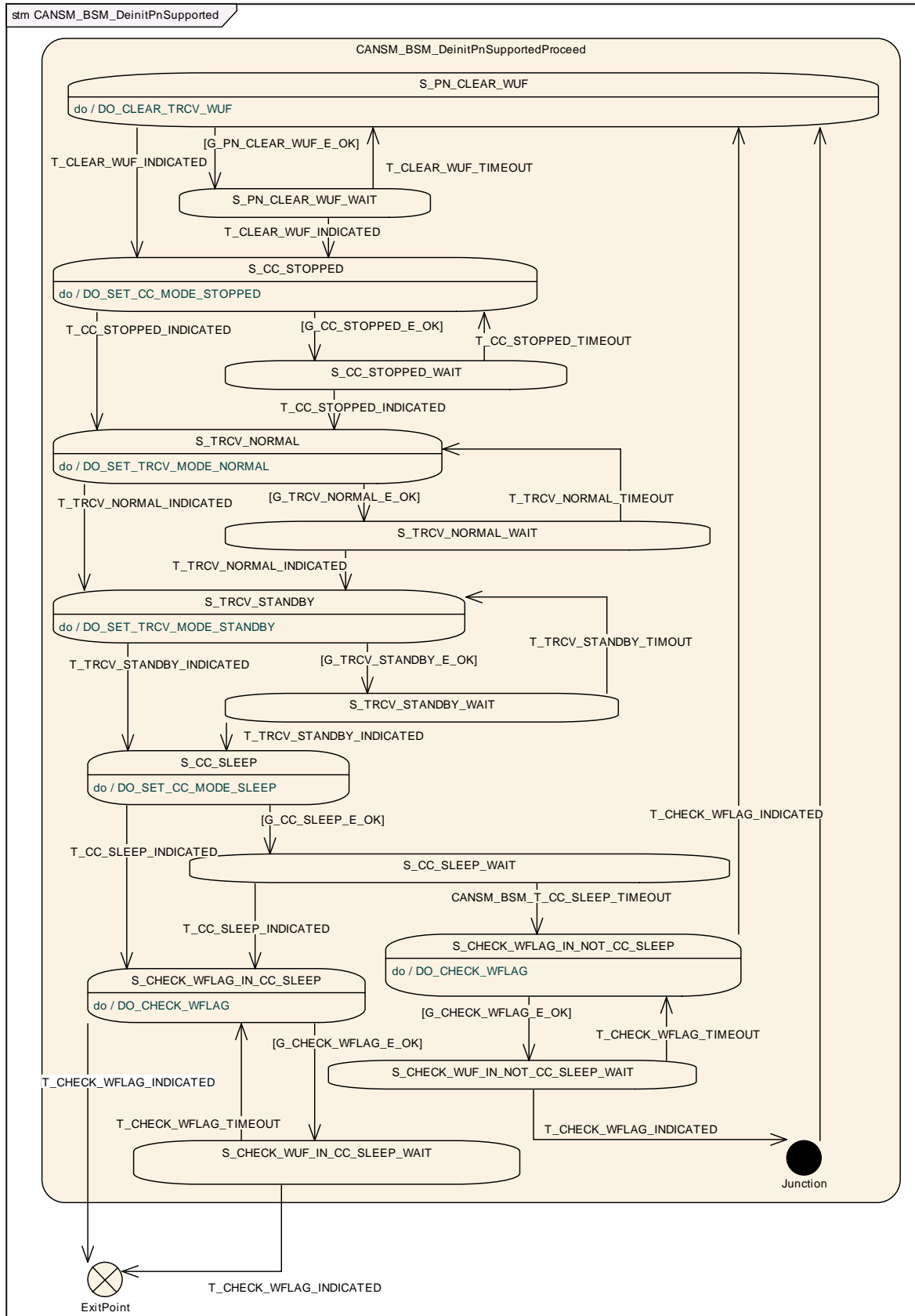


Figure 7-4) is in the state S\_TRCV\_STANDBY, the CanSM module shall operate the do action DO\_SET\_TRCV\_STANDBY and therefore repeat for the configured CAN Transceiver of the CAN network (ref. to [ECUC CanSM 00137](#)) the API request

CanIf\_SetTrcvMode (ref. to chapter 8.5.1) with TransceiverMode equal to CANTRCV\_TRCVMODE\_STANDBY.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.3.14 Guarding condition: G\_TRCV\_STANDBY\_E\_OK**

**[SWS\_CanSM\_00451]** [ The guarding condition G\_TRCV\_STANDBY\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

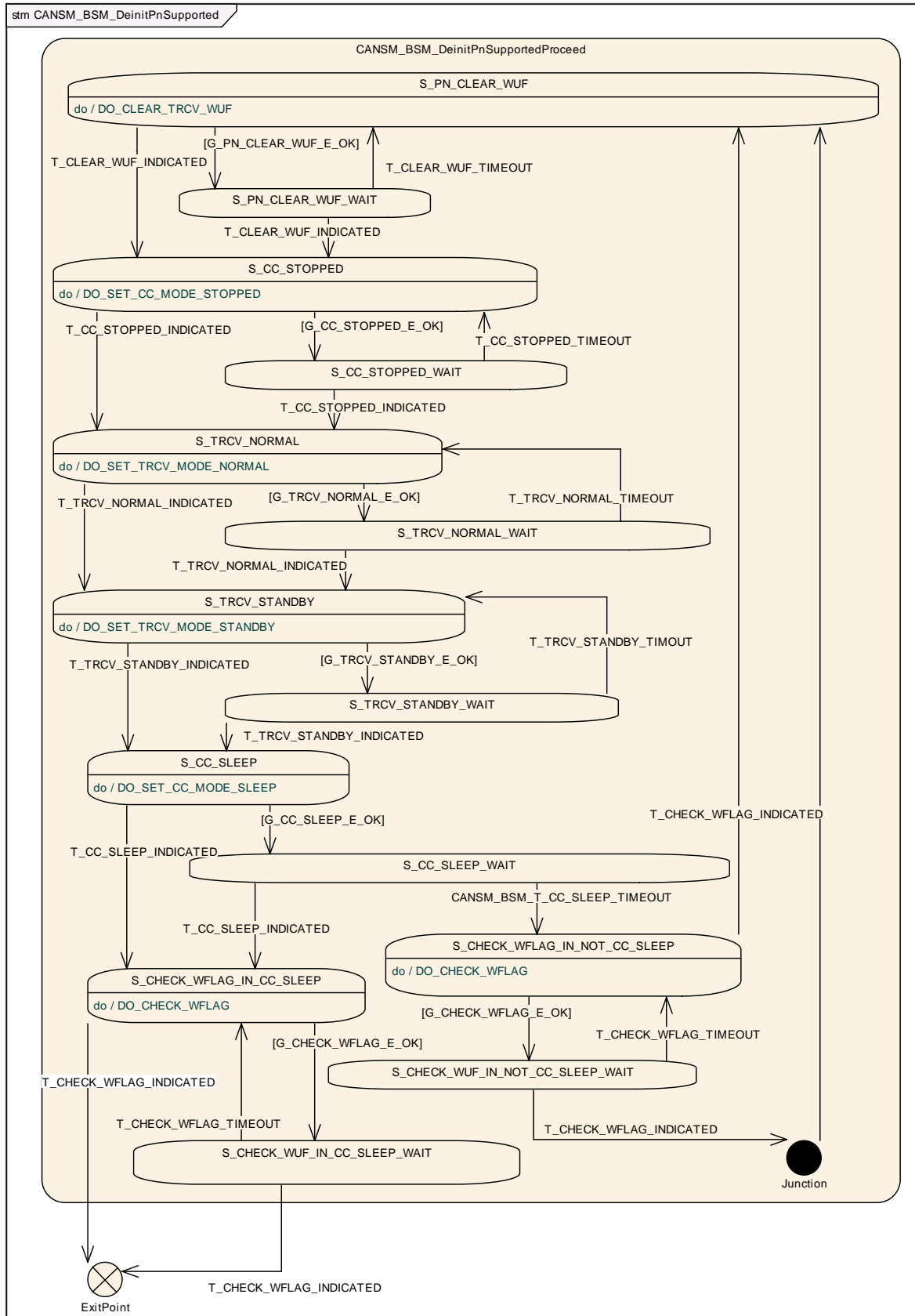


Figure 7-4) shall be passed, if the API call of [SWS CanSM 00450](#) has returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.15 Trigger: T\_TRCV\_STANDBY\_INDICATED**

**[SWS\_CanSM\_00452]** [ If the CanSM module has got the CANTRCV\_TRCVMODE\_STANDBY mode indication (ref. to [SWS\\_CanSM\\_00399](#)) for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) after the respective request (ref. to [SWS\\_CanSM\\_00450](#)), this shall trigger the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

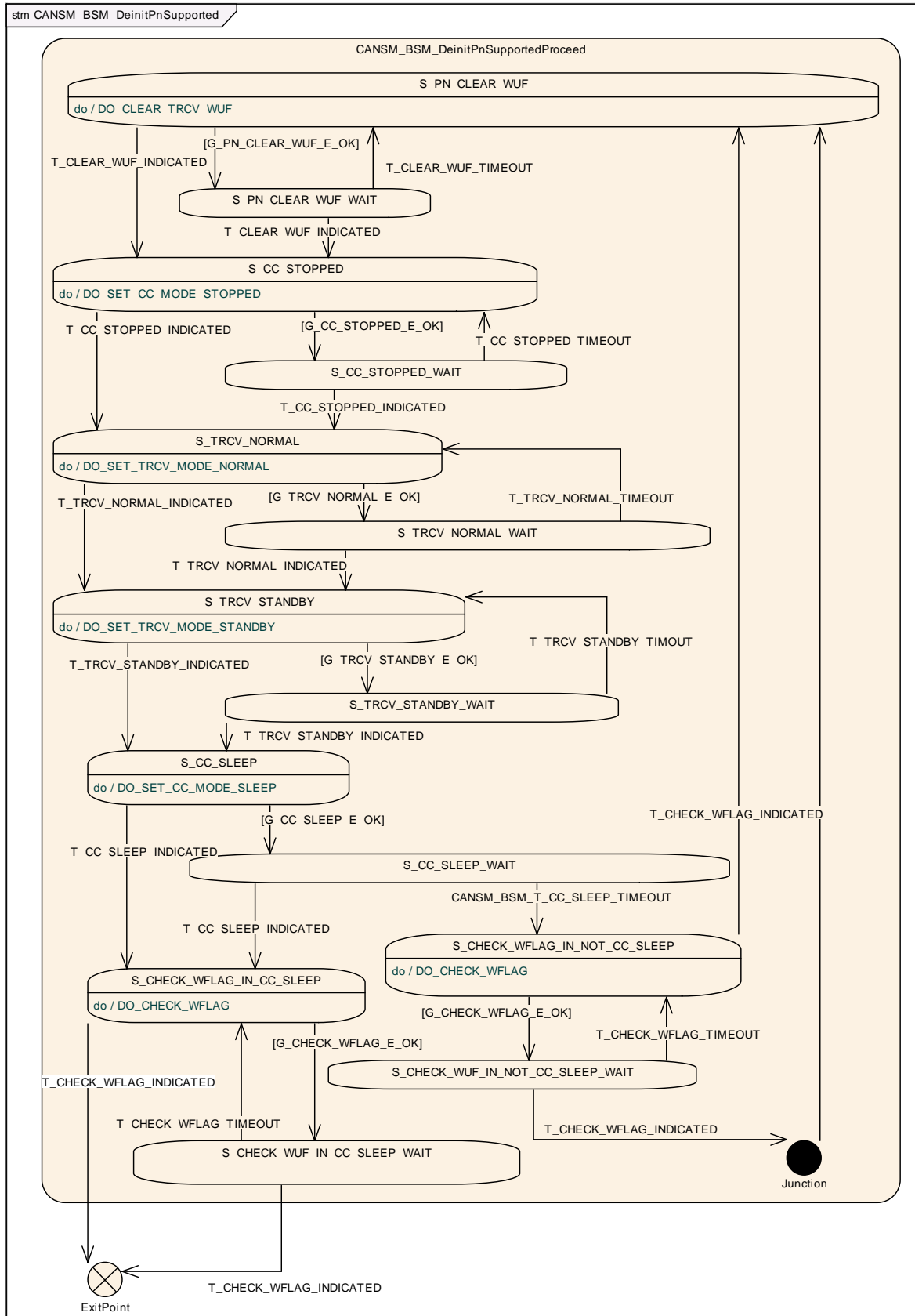


Figure 7-4) of the CAN network with T\_TRCV\_STANDBY\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.16 Trigger: T\_TRCV\_STANDBY\_TIMEOUT**

**[SWS\_CanSM\_00454]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for the supposed transceiver standby indication (ref. to [SWS\\_CanSM\\_00452](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to



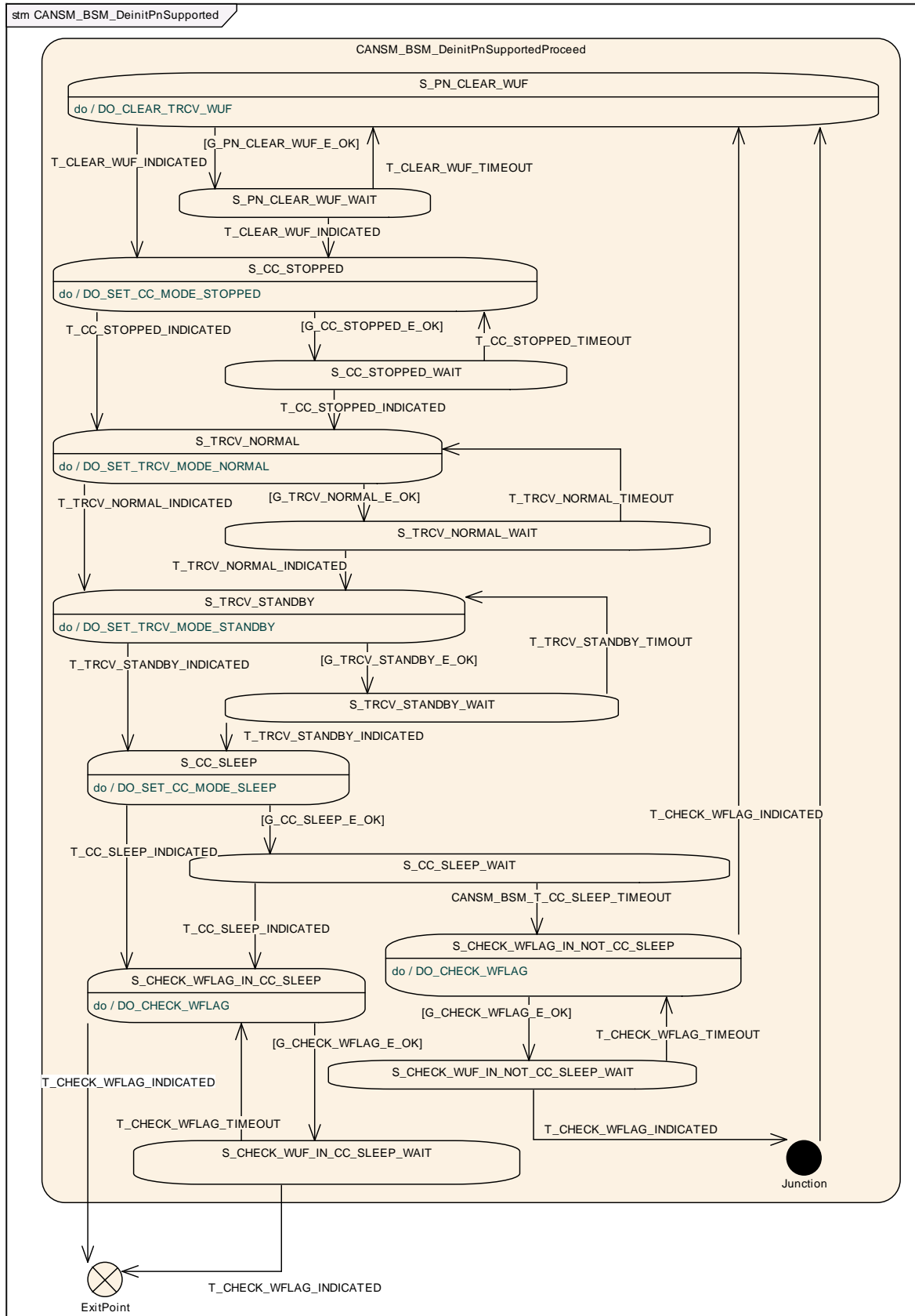


Figure 7-4) of the respective network with  
T\_TRCV\_STANDBY\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.17 State operation to do in: S\_CC\_SLEEP**

**[SWS\_CanSM\_00453]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

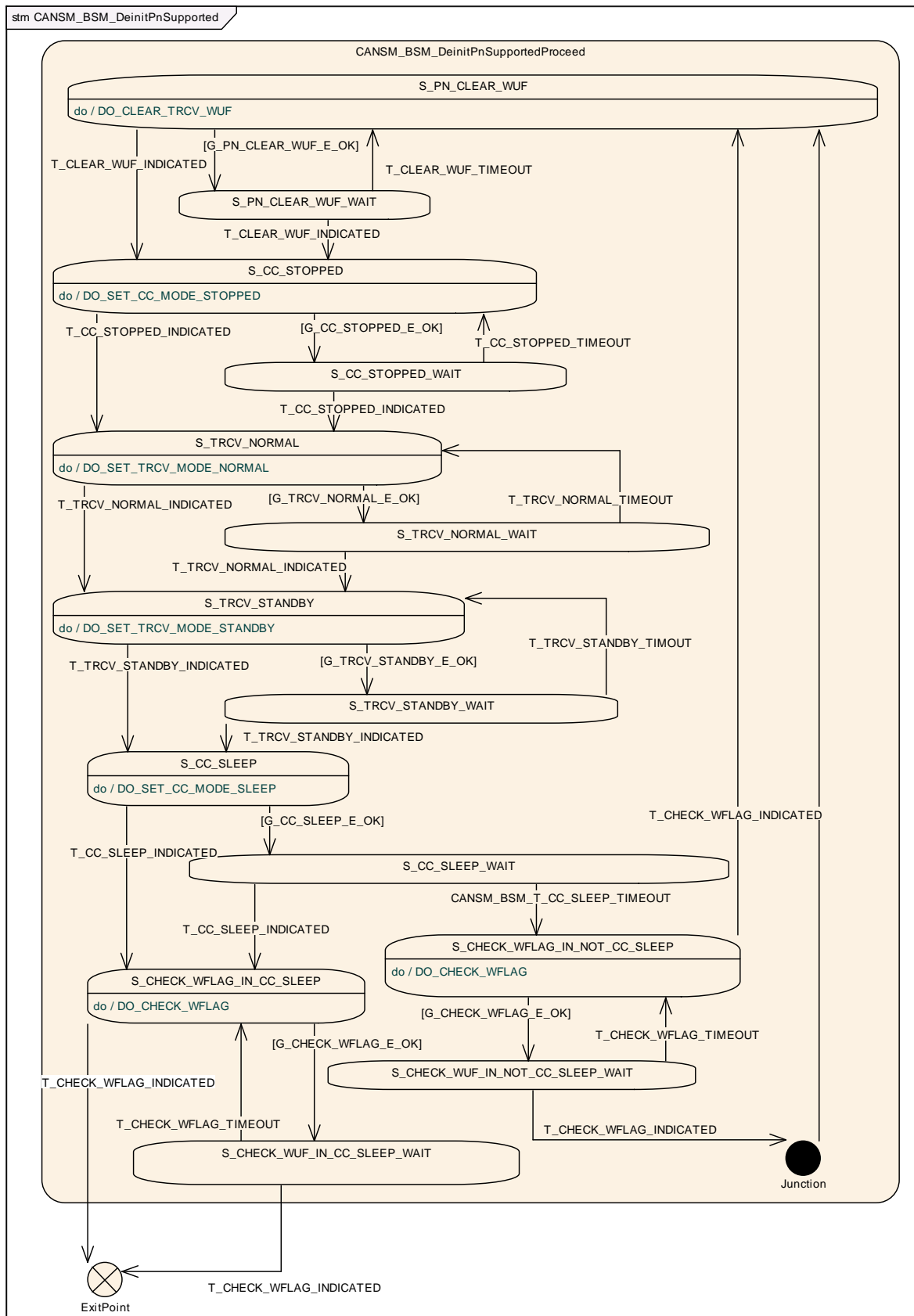


Figure 7-4) is in the state S\_CC\_SLEEP, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_SLEEP and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC CanSM 00141](#)) the API request

CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_SLEEP, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.3.18 Guarding condition: G\_CC\_SLEEP\_E\_OK

**[SWS\_CanSM\_00455]** | The guarding condition G\_CC\_SLEEP\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

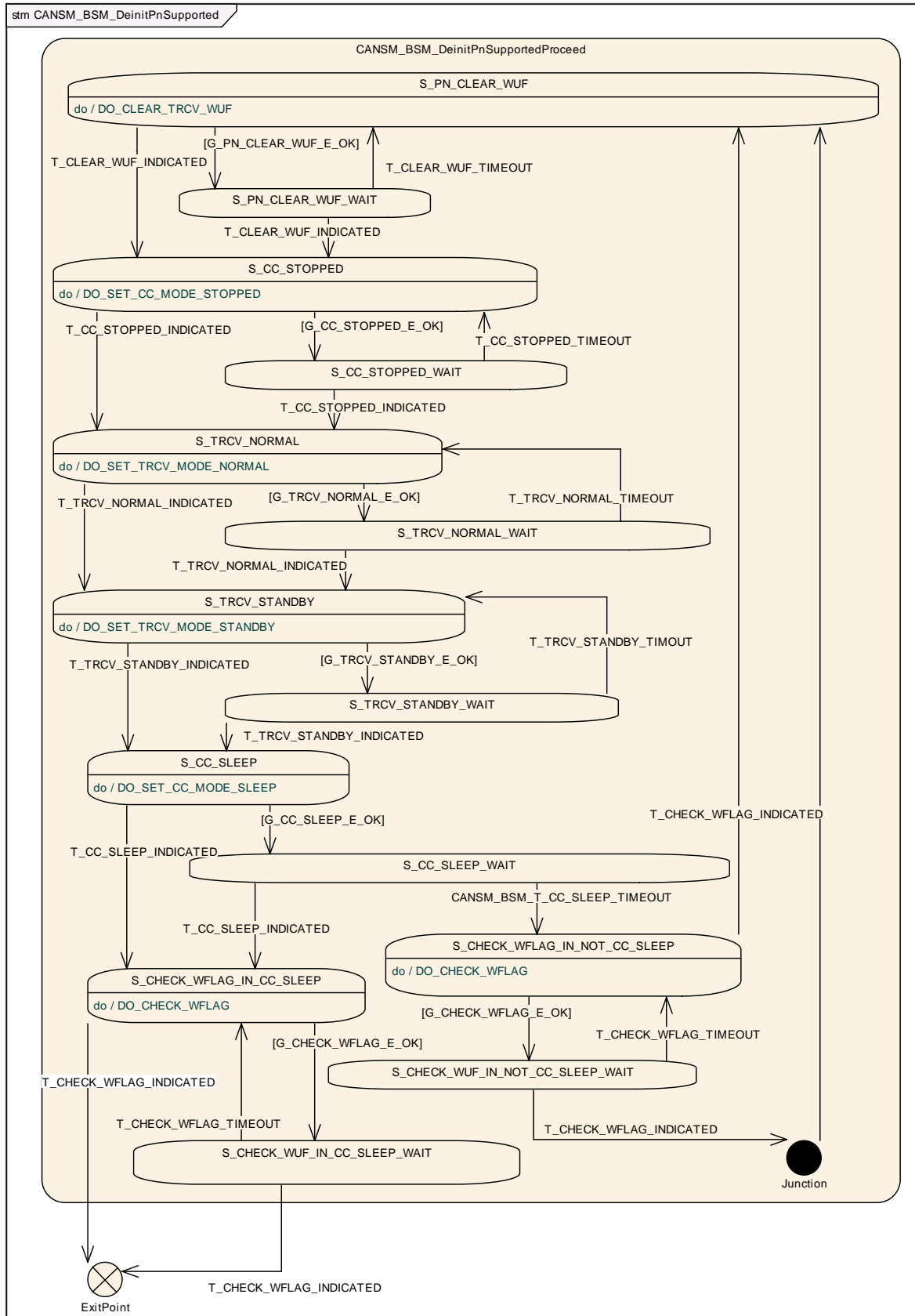


Figure 7-4) shall be passed, if all API calls of [SWS CanSM 00453](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.19 Trigger: T\_CC\_SLEEP\_INDICATED**

**[SWS\_CanSM\_00456]** [ If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to set the CAN controllers of the CAN network to sleep mode (ref. to [SWS\\_CanSM\\_00453](#)), this shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to

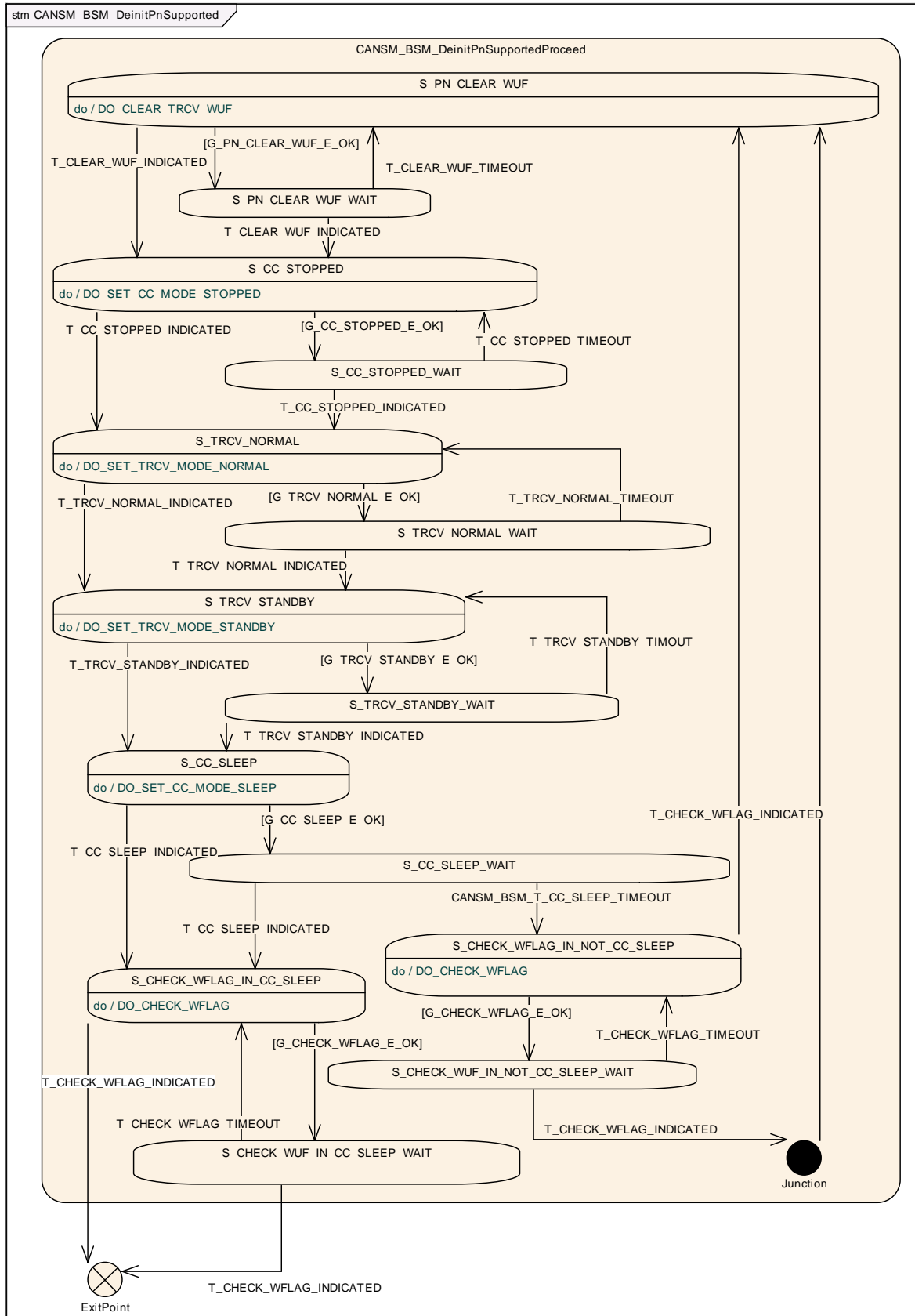


Figure 7-4) of the CAN network with T\_CC\_SLEEP\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.20 Trigger: CANSM\_BSM\_T\_CC\_SLEEP\_TIMEOUT**

**[SWS\_CanSM\_00457]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller sleep mode indications (ref. to [SWS\\_CanSM\\_00456](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to



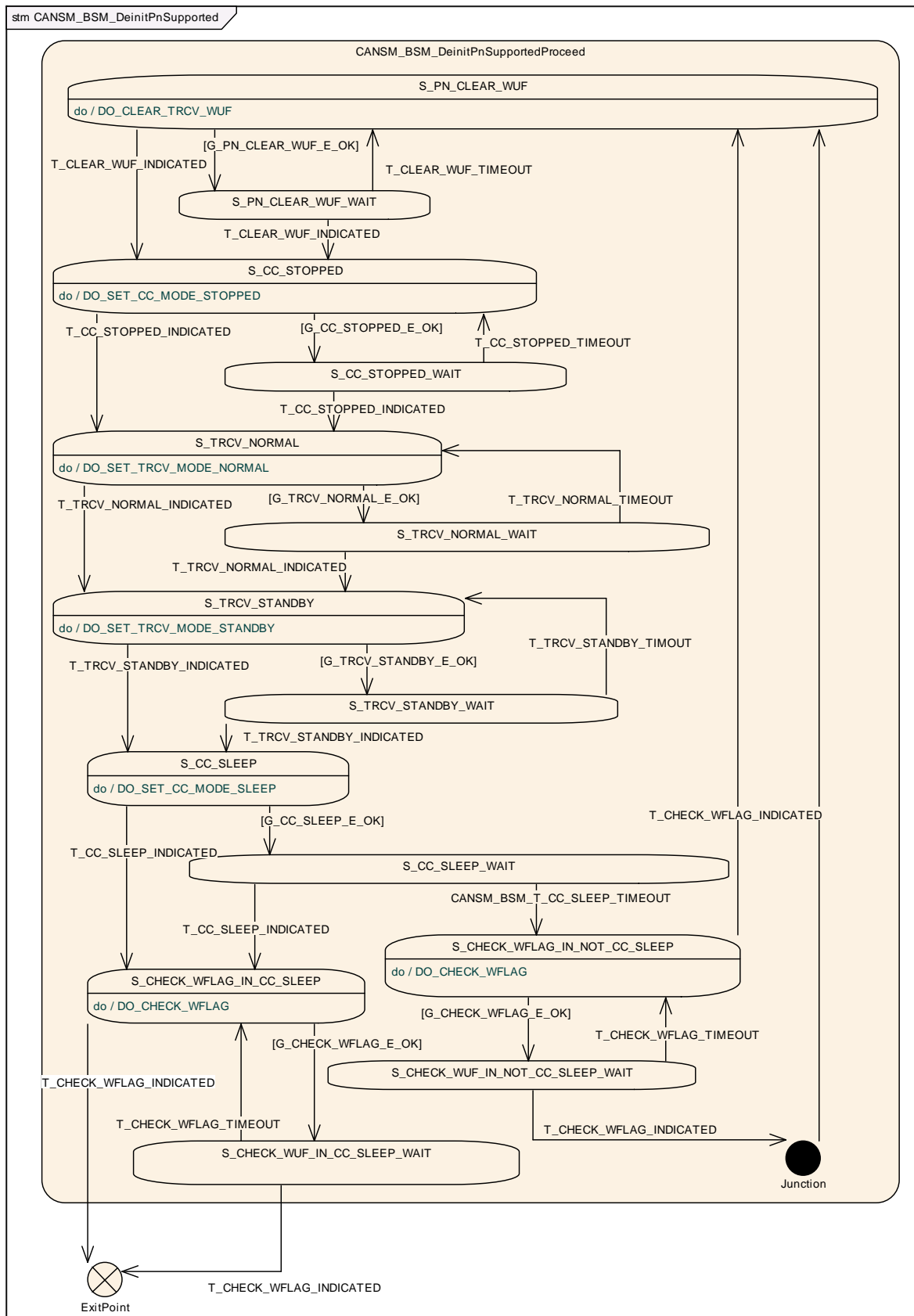


Figure 7-4) of the respective network with  
CANSM\_BSM\_T\_CC\_SLEEP\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.21 State operation to do in: S\_CHECK\_WFLAG\_IN\_CC\_SLEEP**

**[SWS\_CanSM\_00458]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

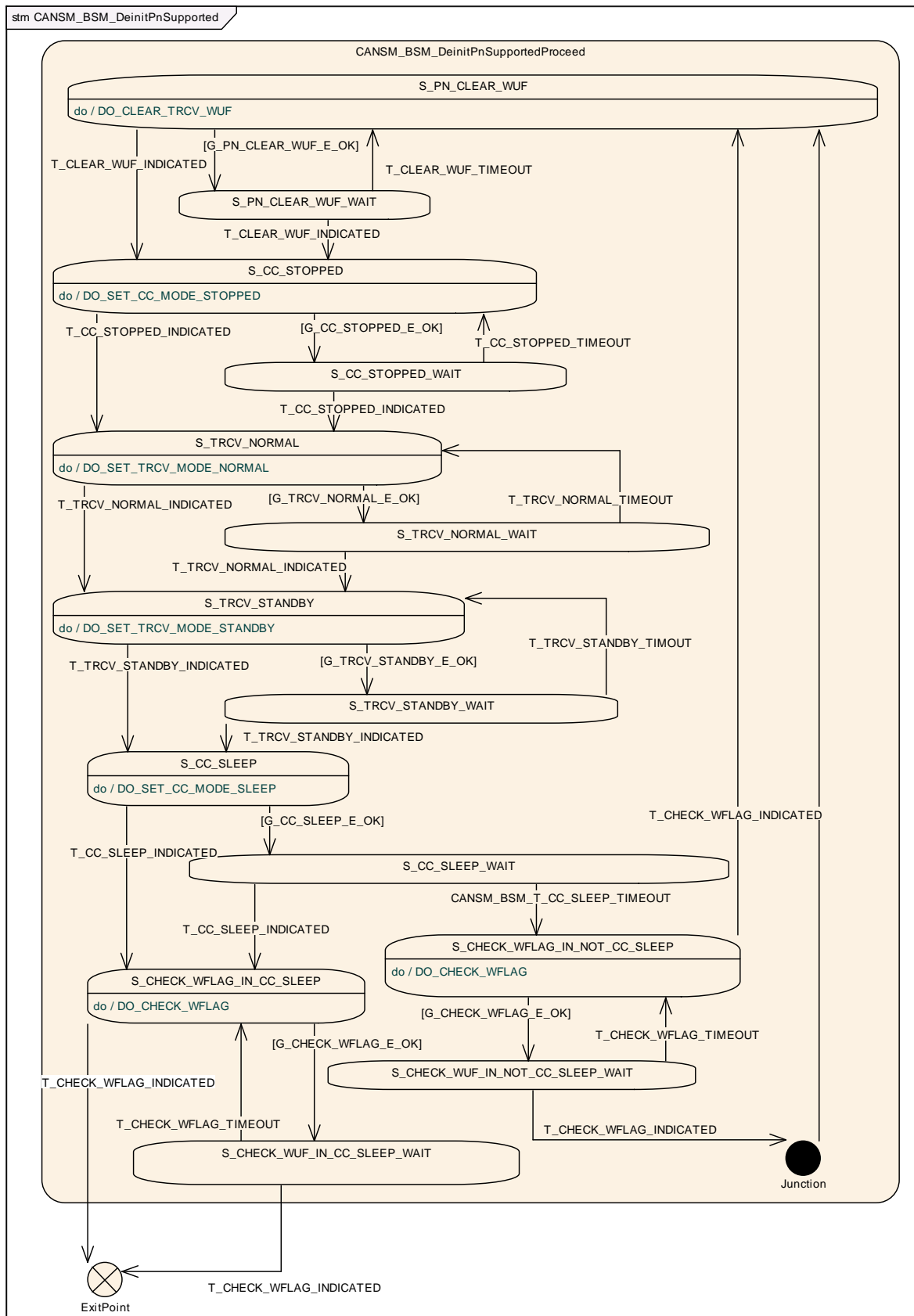


Figure 7-4) is in the state S\_CHECK\_WFLAG\_IN\_CC\_SLEEP, the CanSM module operate the do action DO\_CHECK\_WFLAG and therefore repeat the API request CanIf\_CheckTrcvWakeFlag (ref. to chapter 8.5.1) and use the configured CAN

Transceiver of the related Network (ref. to [ECUC\\_CanSM\\_00137](#)) as Transceiver parameter.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.3.22 Guarding condition: G\_CHECK\_WFLAG\_E\_OK**

**[SWS\_CanSM\_00459]** [ The guarding condition G\_CHECK\_WFLAG\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

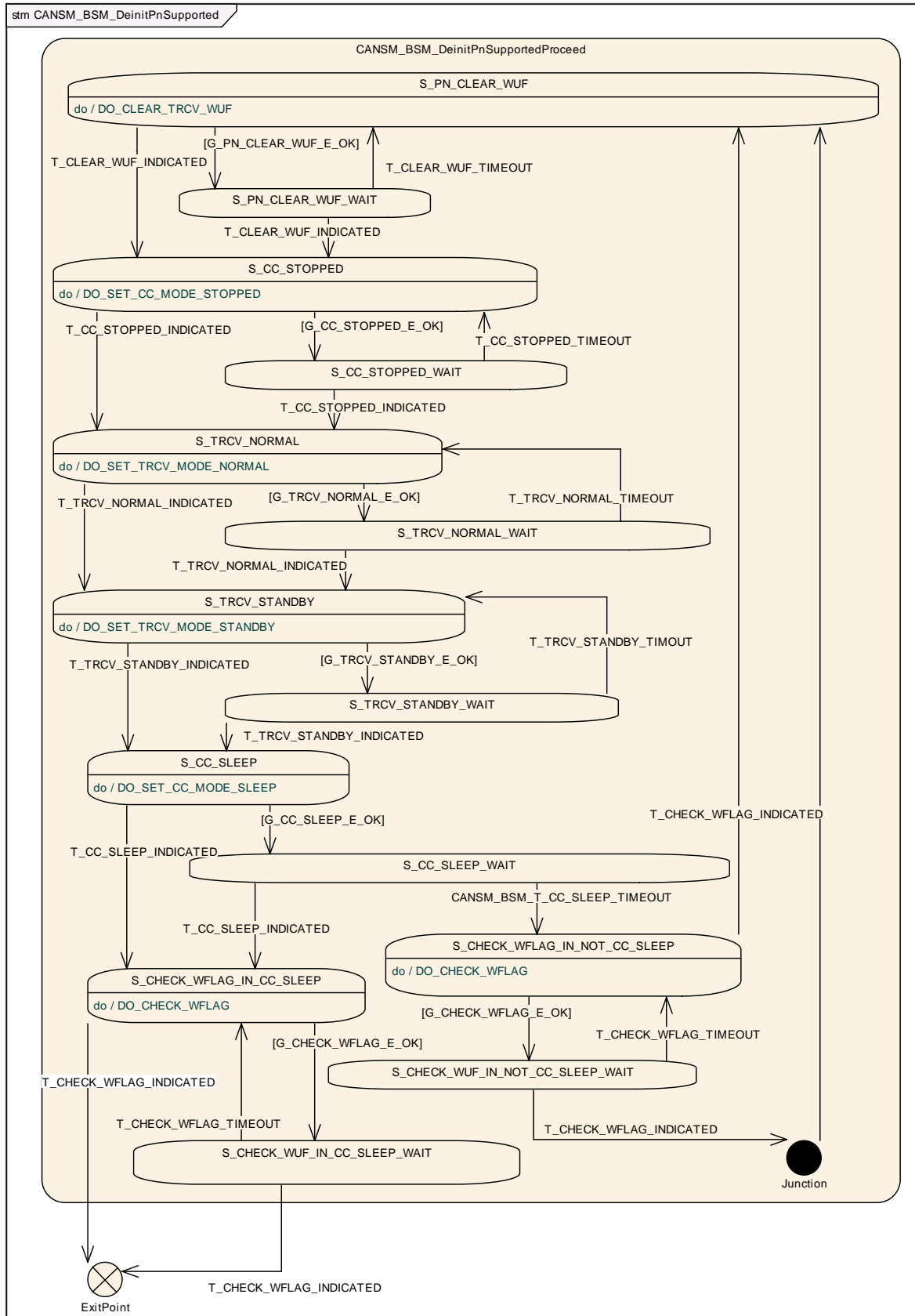


Figure 7-4) shall be passed, if the API call of [SWS CanSM\\_00458](#) or [SWS CanSM\\_00462](#) has returned `E_OK`.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.23 Trigger: T\_CHECK\_WFLAG\_INDICATED**

**[SWS\_CanSM\_00460]** [ The callback function

CanSM\_CheckTransceiverWakeFlagIndication (ref. to [SWS\\_CanSM\\_00416](#))  
shall trigger the sub state machine CANSM\_BSM\_DeinitPnSupported (ref. to

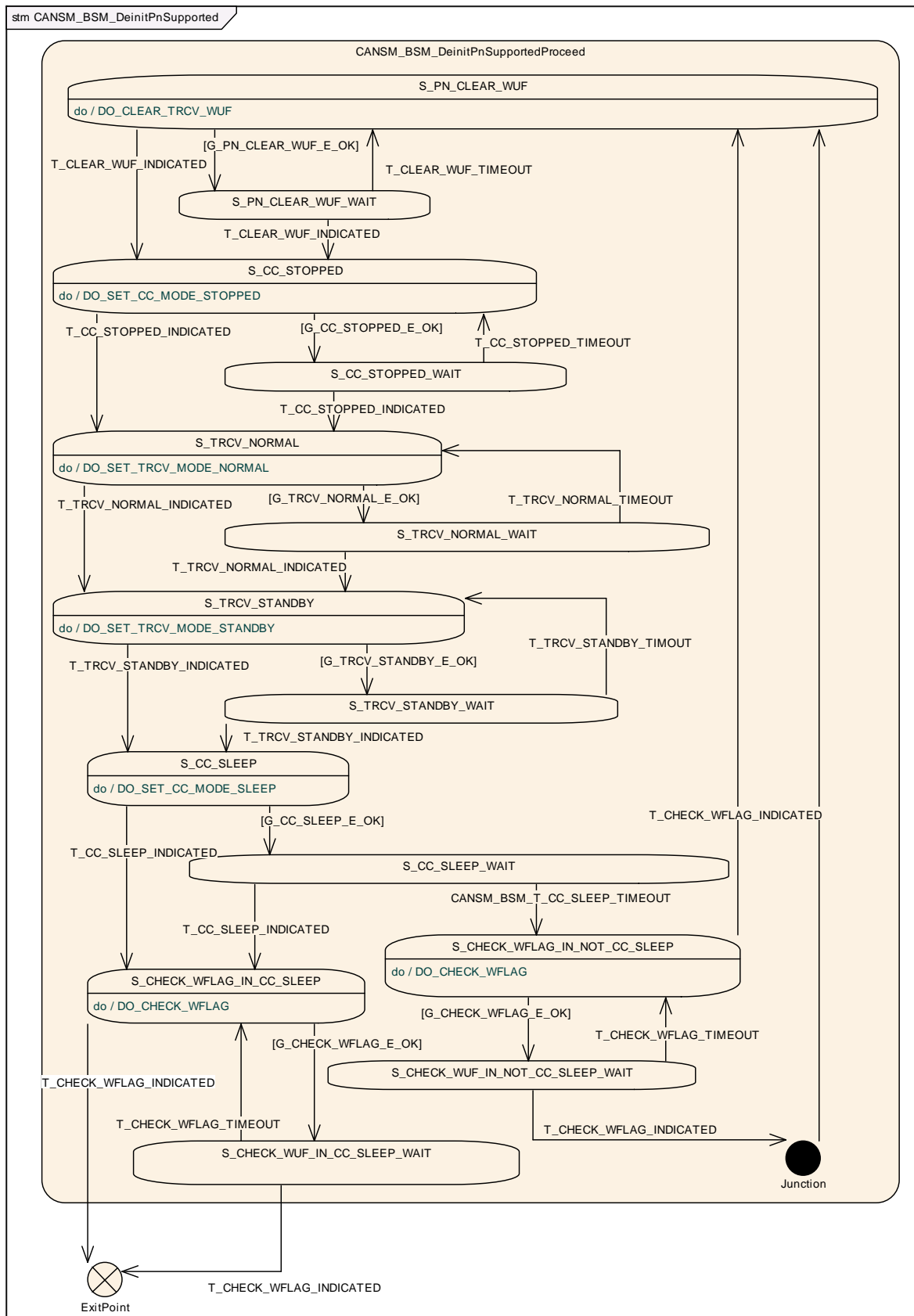


Figure 7-4) of the CAN network with T\_CHECK\_WFLAG\_INDICATED, if the function parameter Transceiver of CanSM\_CheckTransceiverWakeFlagIndication

matches to the configured CAN Transceiver (ref. to [ECUC\\_CanSM\\_00137](#)) of the CAN network.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.3.24 Trigger: T\_CHECK\_WFLAG\_TIMEOUT

**[SWS\_CanSM\_00461]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for the callback function `CanSM_CheckTransceiver-WakeFlagIndication` (ref. to [SWS\\_CanSM\\_00460](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to



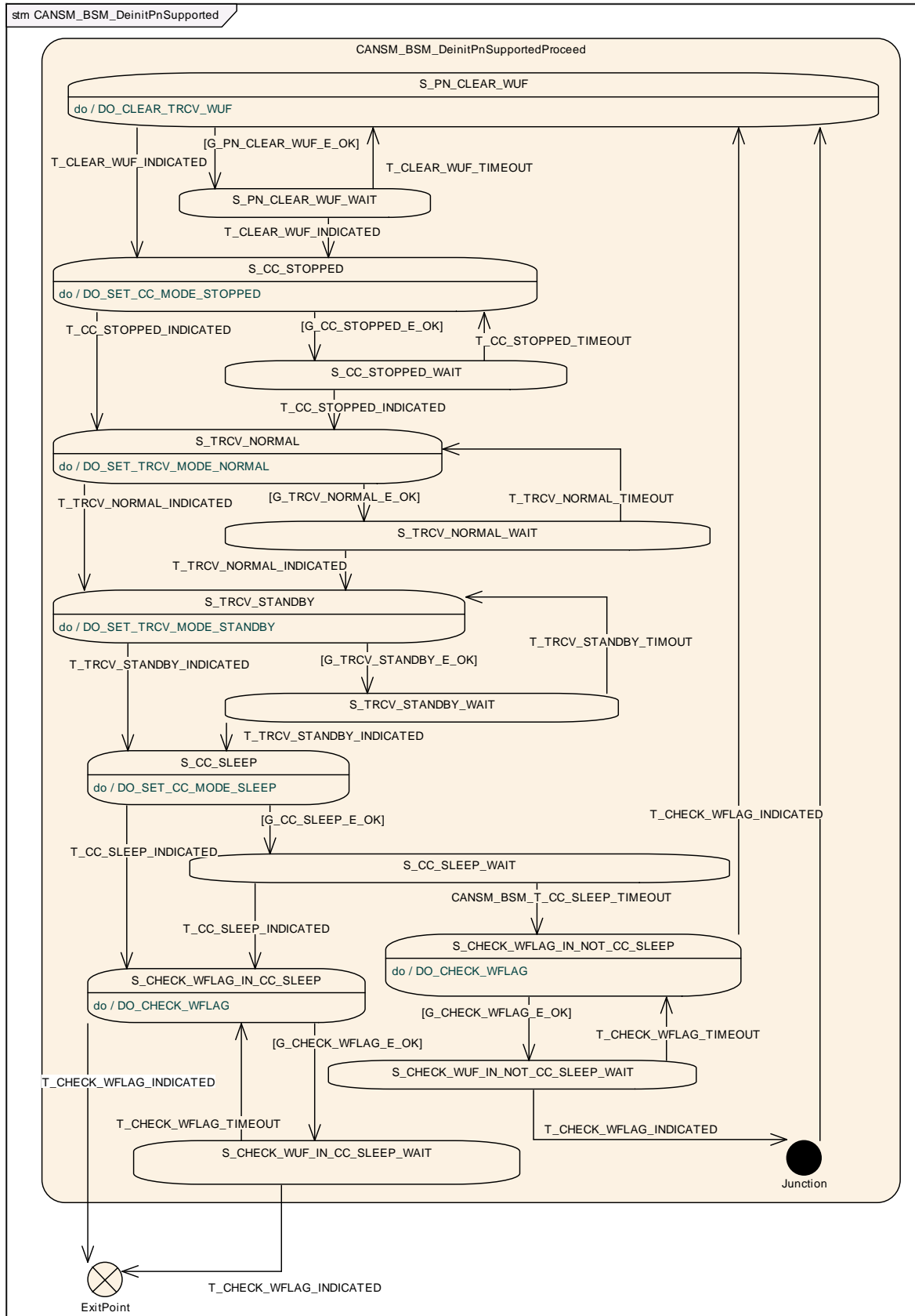


Figure 7-4) of the respective network with `T_CHECK_WFLAG_TIMEOUT`.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.3.25 State operation to do in: S\_CHECK\_WFLAG\_IN\_NOT\_CC\_SLEEP**

**[SWS\_CanSM\_00462]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnSupported (ref. to

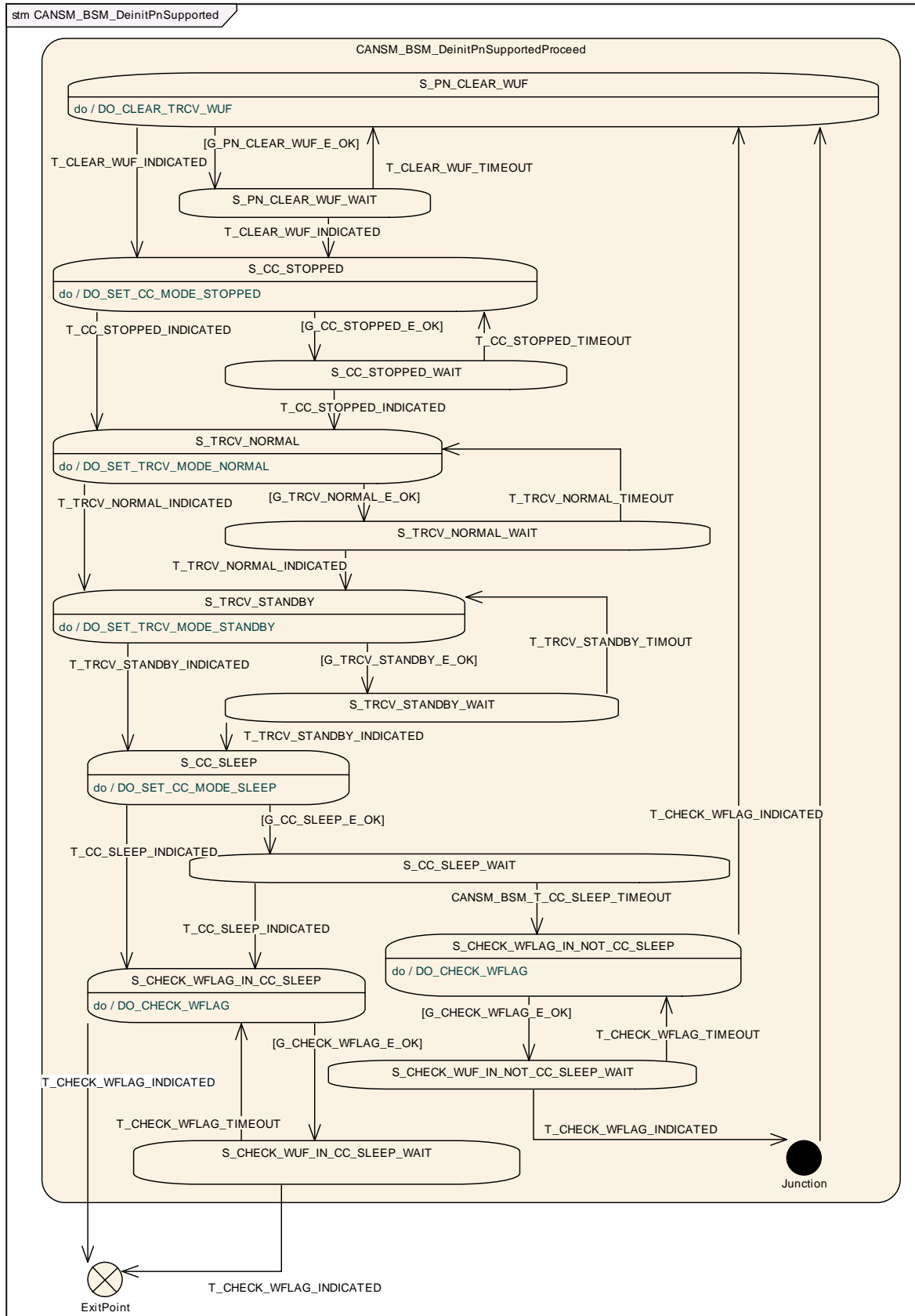
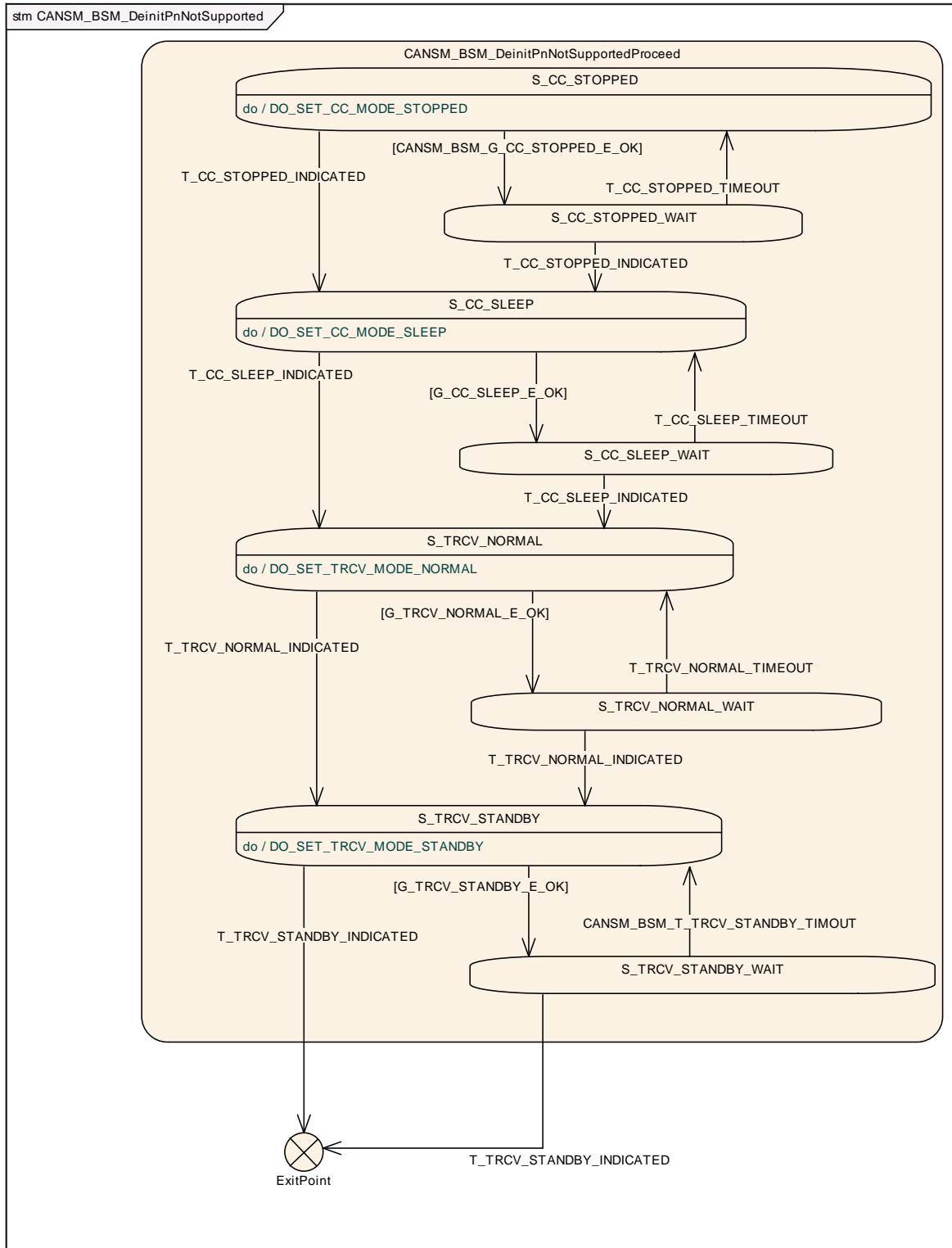


Figure 7-4) is in the state S\_CHECK\_WFLAG\_IN\_NOT\_CC\_SLEEP, the CanSM module operate the do action DO\_CHECK\_WFLAG and therefore repeat the API request CanIf\_CheckTrcvWakeFlag (ref. to chapter 8.5.1) and use the configured

CAN Transceiver of the related Network (ref. to [ECUC CanSM\\_00137](#)) as Transceiver parameter.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.4 Sub state machine: CANSM\_BSM\_DeinitPnNotSupported**



**Figure 7-5: CANSM\_BSM\_DeinitPnNotSupported, sub state machine of CANSM\_BSM\_S\_PRE\_NOCOM**

7.2.20.4.1 State operation to do in: S\_CC\_STOPPED

[SWS\_CanSM\_00464] [ As long the sub state machine  
CANSM\_BSM\_DeinitPnNotSupported (ref. to

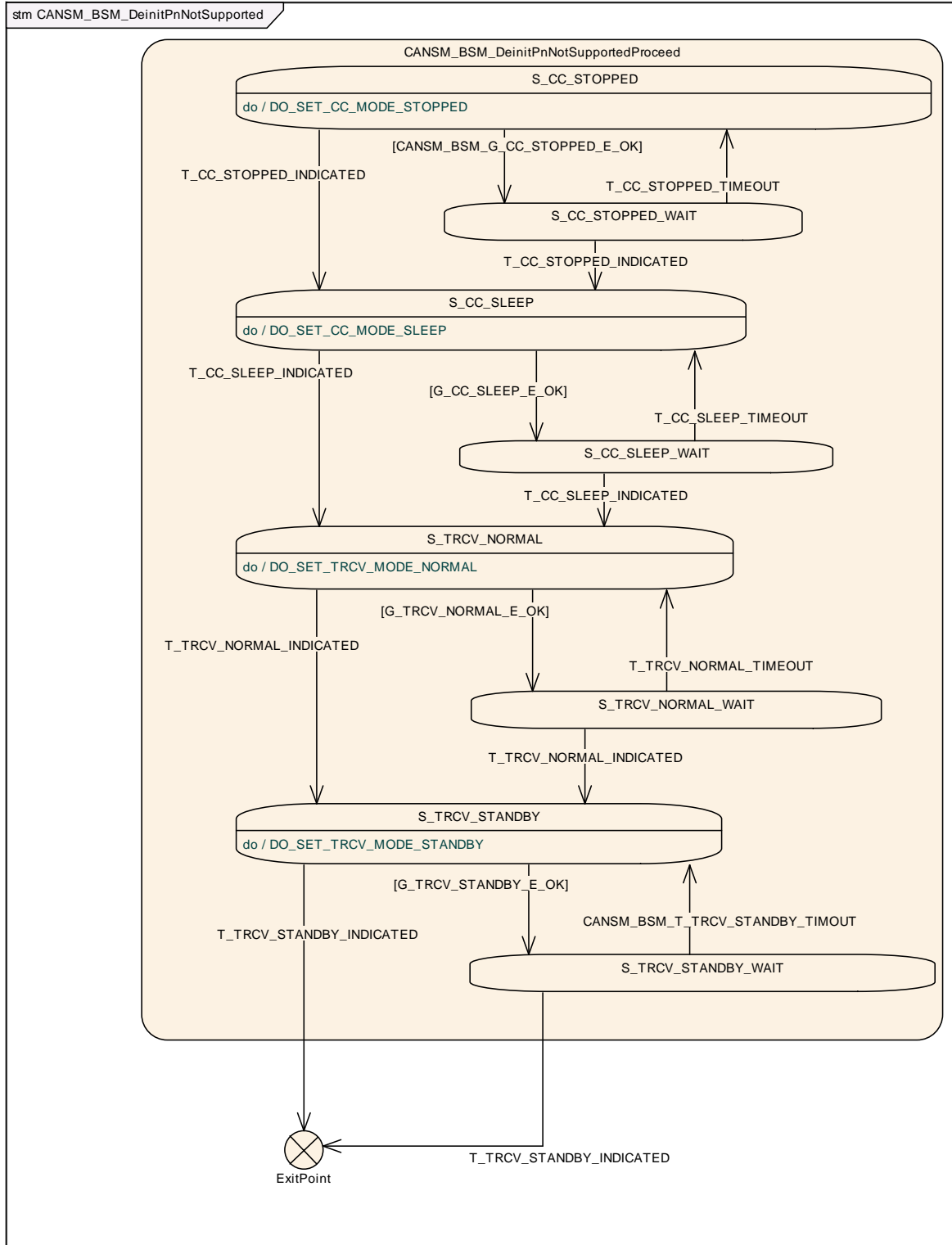


Figure 7-5) is in the state S\_CC\_STOPPED, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STOPPED and therefore repeat for all configured CAN controllers of the CAN network (ref. to ECUC\_CanSM\_00141) the API request

CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STOPPED, if the current CAN controller mode (ref. to [SWS CanSM 00638](#)) is different.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.4.2 Guarding condition: CANSM\_BSM\_G\_CC\_STOPPED\_OK**

**[SWS\_CanSM\_00465]** | The guarding condition CANSM\_BSM\_G\_CC\_STOPPED\_OK of the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

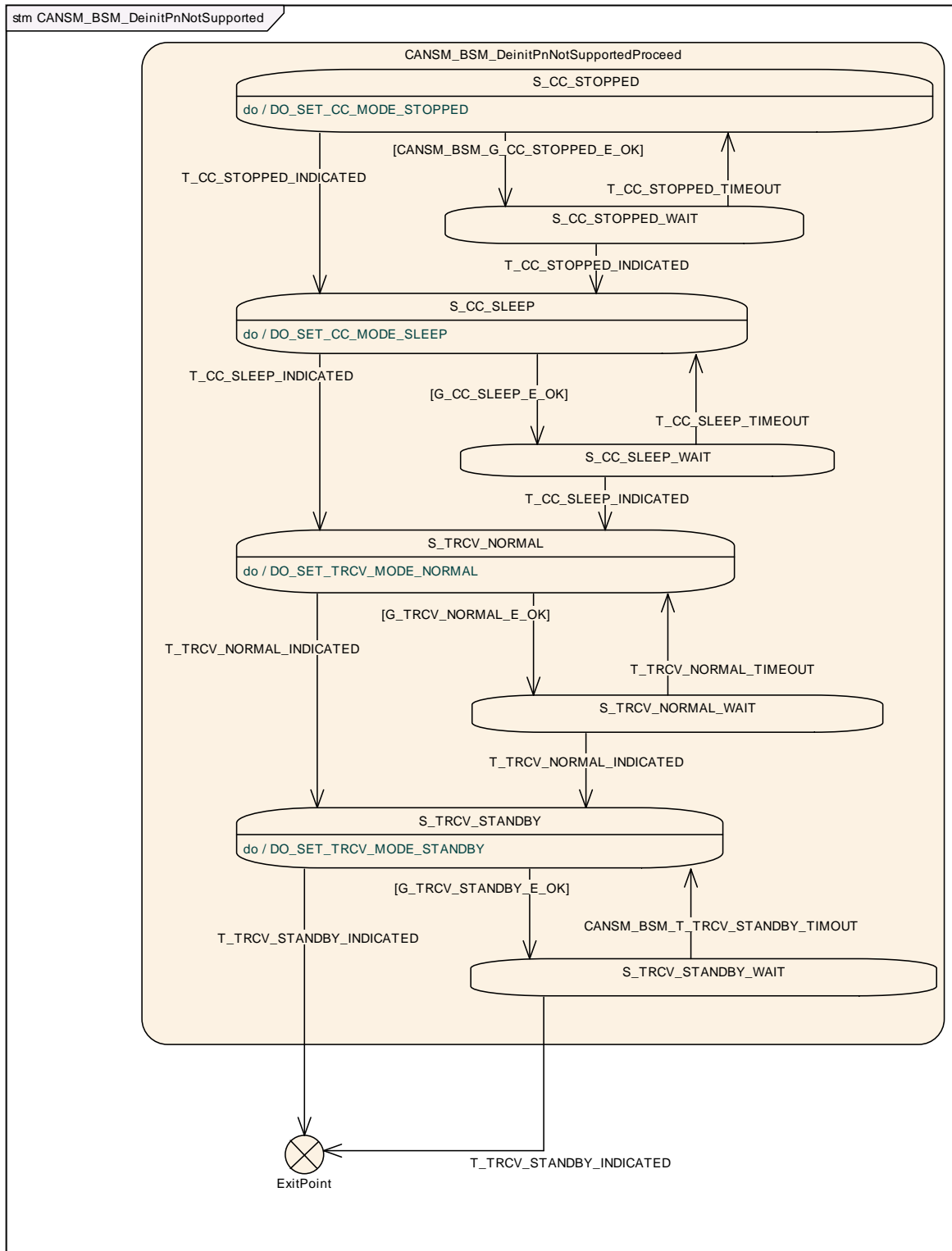


Figure 7-5) shall be passed, if all API calls of [SWS CanSM 00464](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.4.3 Trigger: T\_CC\_STOPPED\_INDICATED**

**[SWS\_CanSM\_00466]** [ If CanSM module has got all mode indications (ref. to [SWS CanSM 00396](#)) for the configured CAN controllers of the CAN network (ref. to

[ECUC CanSM\\_00141](#)) after the respective requests to stop the CAN controllers of the CAN network (ref. to [SWS CanSM\\_00464](#)), this shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

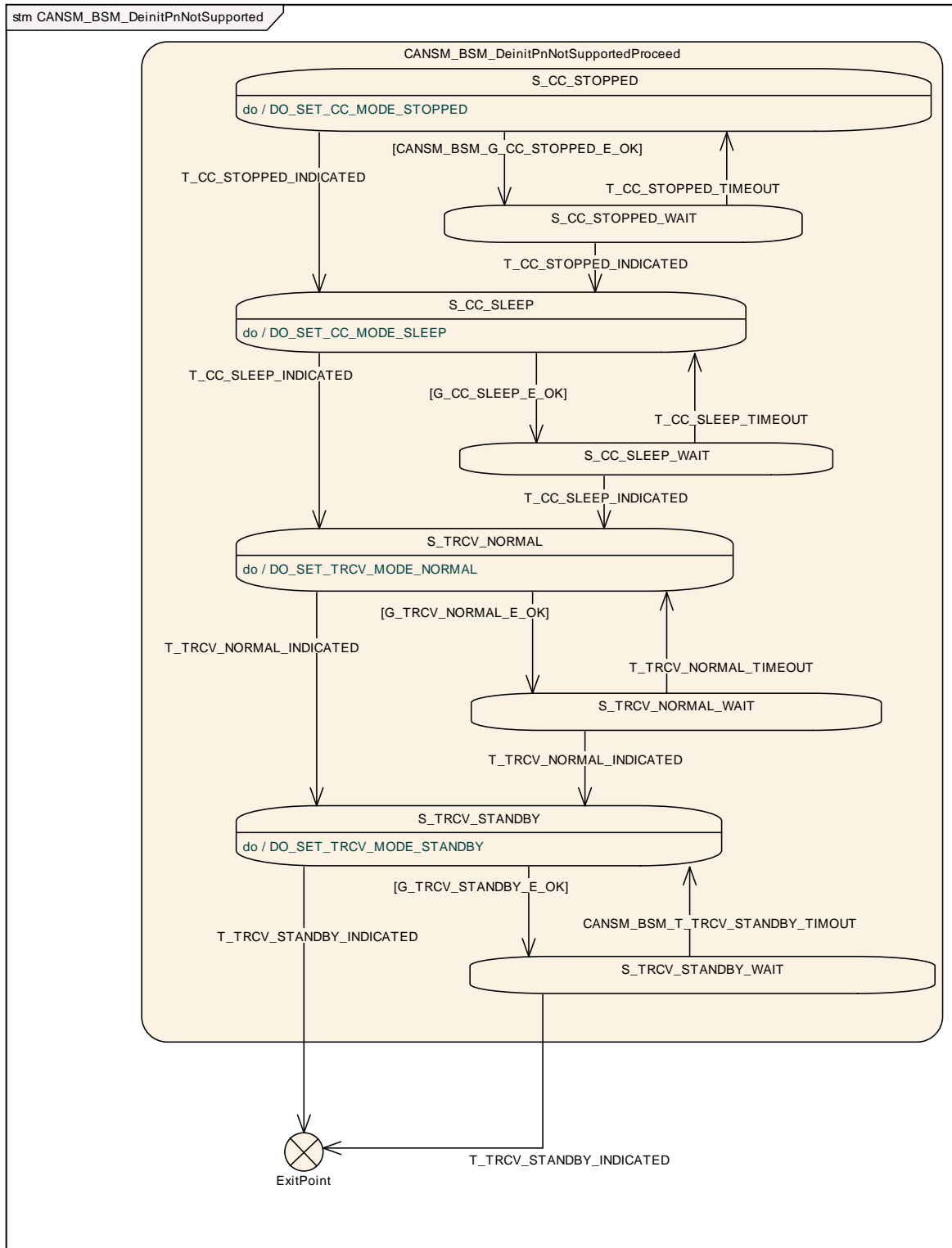


Figure 7-5) of the CAN network with T\_CC\_STOPPED\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)



7.2.20.4.4 Trigger: T\_CC\_STOPPED\_TIMEOUT

**[SWS\_CanSM\_00467]** [ After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller stopped mode indications (ref. to [SWS\\_CanSM\\_00466](#)), this condition shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

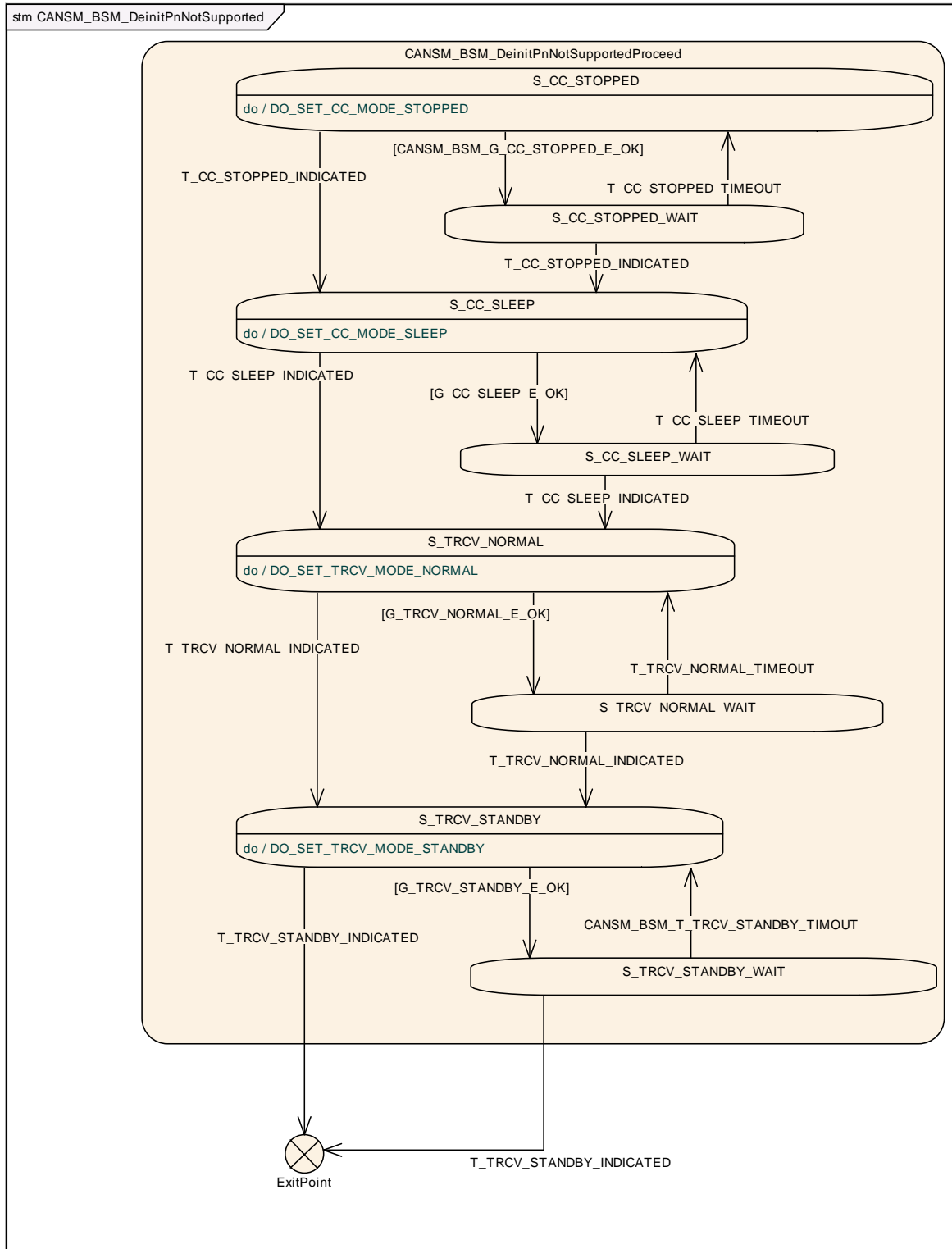


Figure 7-5) of the respective network with  
T\_CC\_STOPPED\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.4.5 State operation to do in: S\_CC\_SLEEP**

**[SWS\_CanSM\_00468]** [ As long the sub state machine  
CANSM\_BSM\_DeinitPnNotSupported (ref. to

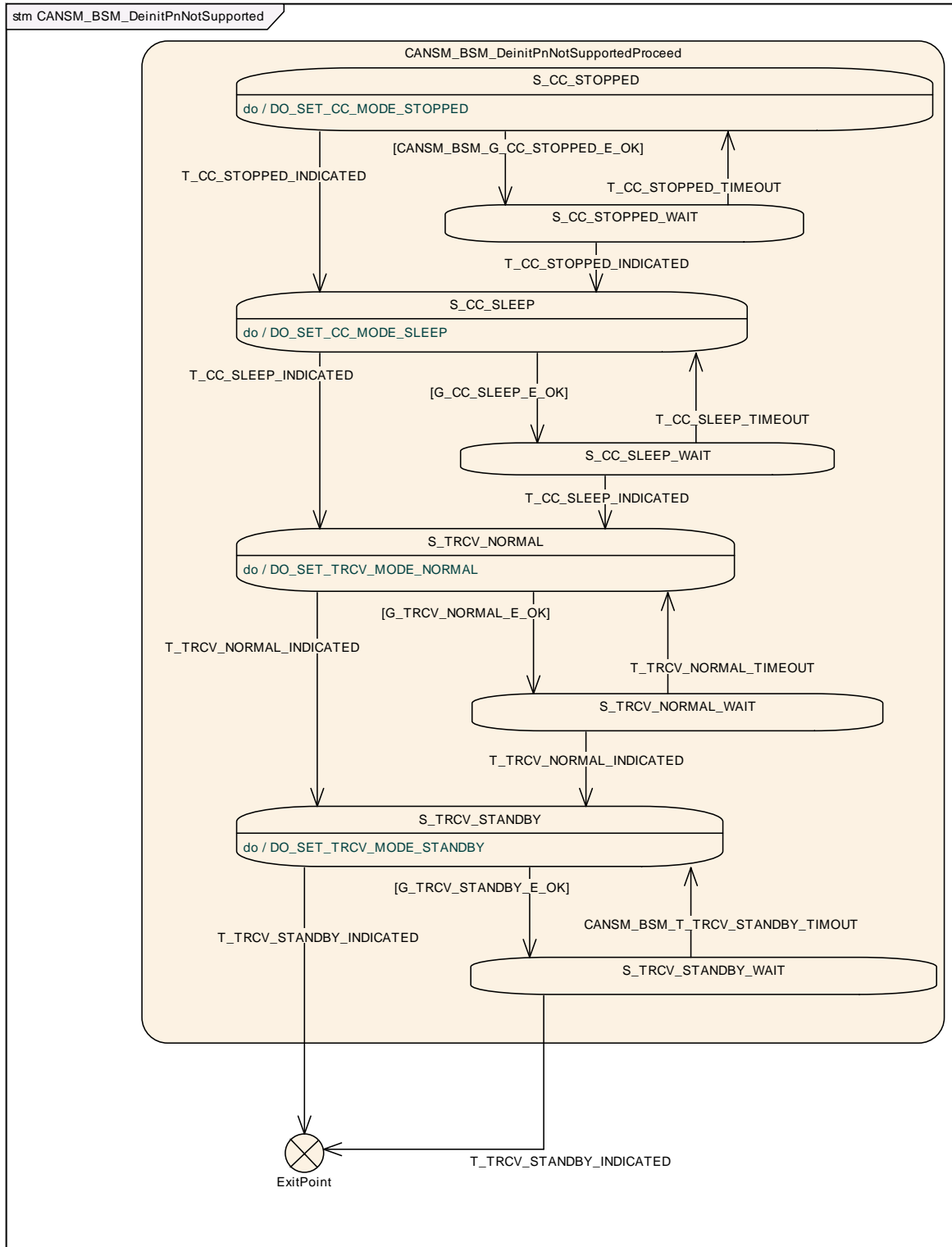


Figure 7-5) is in the state `S_CC_SLEEP`, the CanSM module shall operate the do action `DO_SET_CC_MODE_SLEEP` and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC CanSM 00141](#)) the API request `CanIf_SetControllerMode` (ref. to chapter 8.5.1) with `ControllerMode` equal to `CAN_CS_SLEEP`, if the current CAN controller mode (ref. to [SWS CanSM 00638](#)) is different.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.4.6 Guarding condition: `G_CC_SLEEP_E_OK`

**[SWS\_CanSM\_00469]** [ The guarding condition `G_CC_SLEEP_E_OK` of the sub state machine `CANSM_BSM_DeinitPnNotSupported` (ref. to

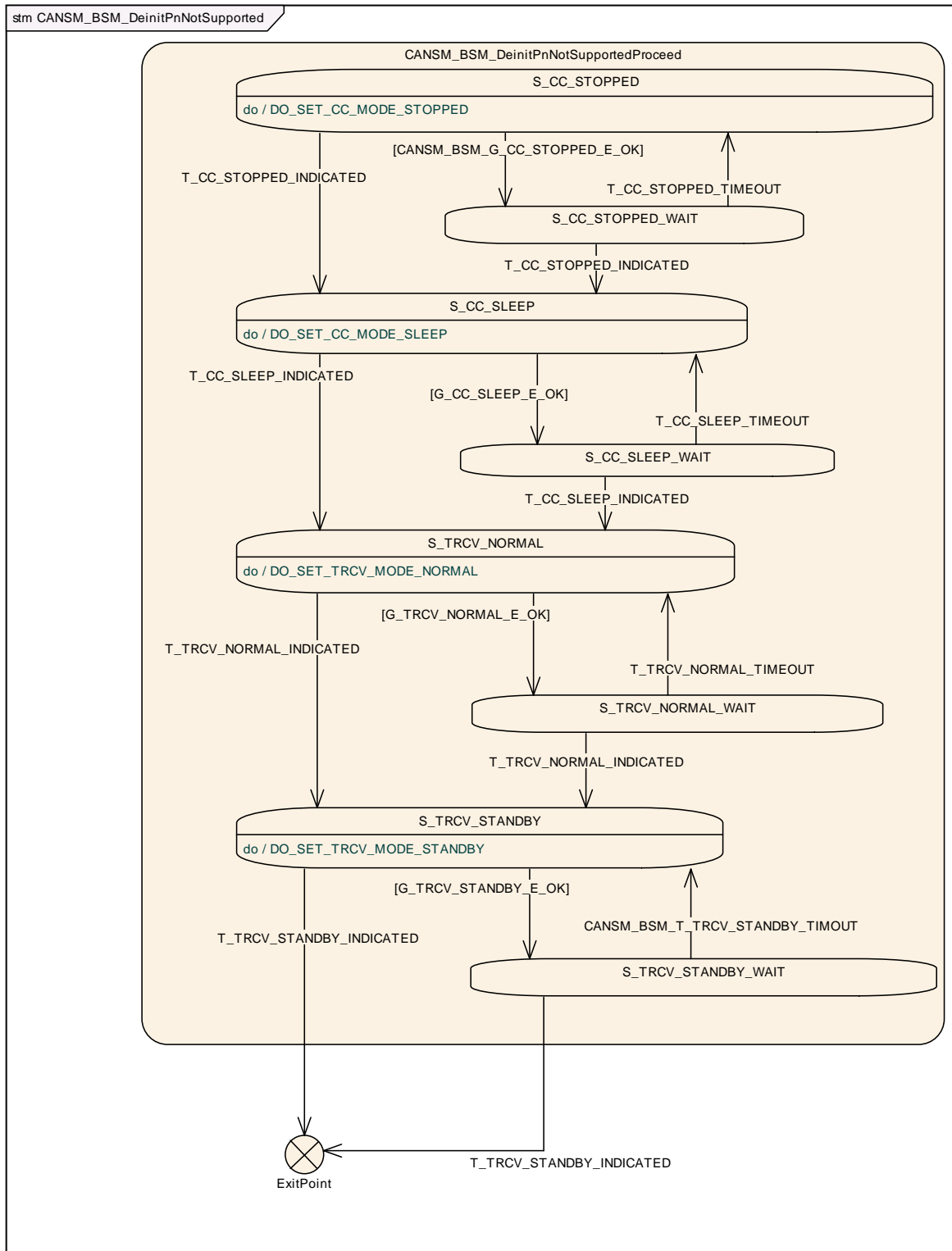


Figure 7-5) shall be passed, if all API calls of [SWS CanSM 00468](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.4.7 Trigger: T\_CC\_SLEEP\_INDICATED**

**[SWS\_CanSM\_00470]** [ If CanSM module has got all mode indications (ref. to [SWS CanSM 00396](#)) for the configured CAN controllers of the CAN network (ref. to

[ECUC CanSM\\_00141](#)) after the respective requests to set the CAN controllers of the CAN network to sleep mode (ref. to [SWS CanSM\\_00468](#)), this shall trigger the sub state machine `CANSM_BSM_DeinitPnNotSupported` (ref. to

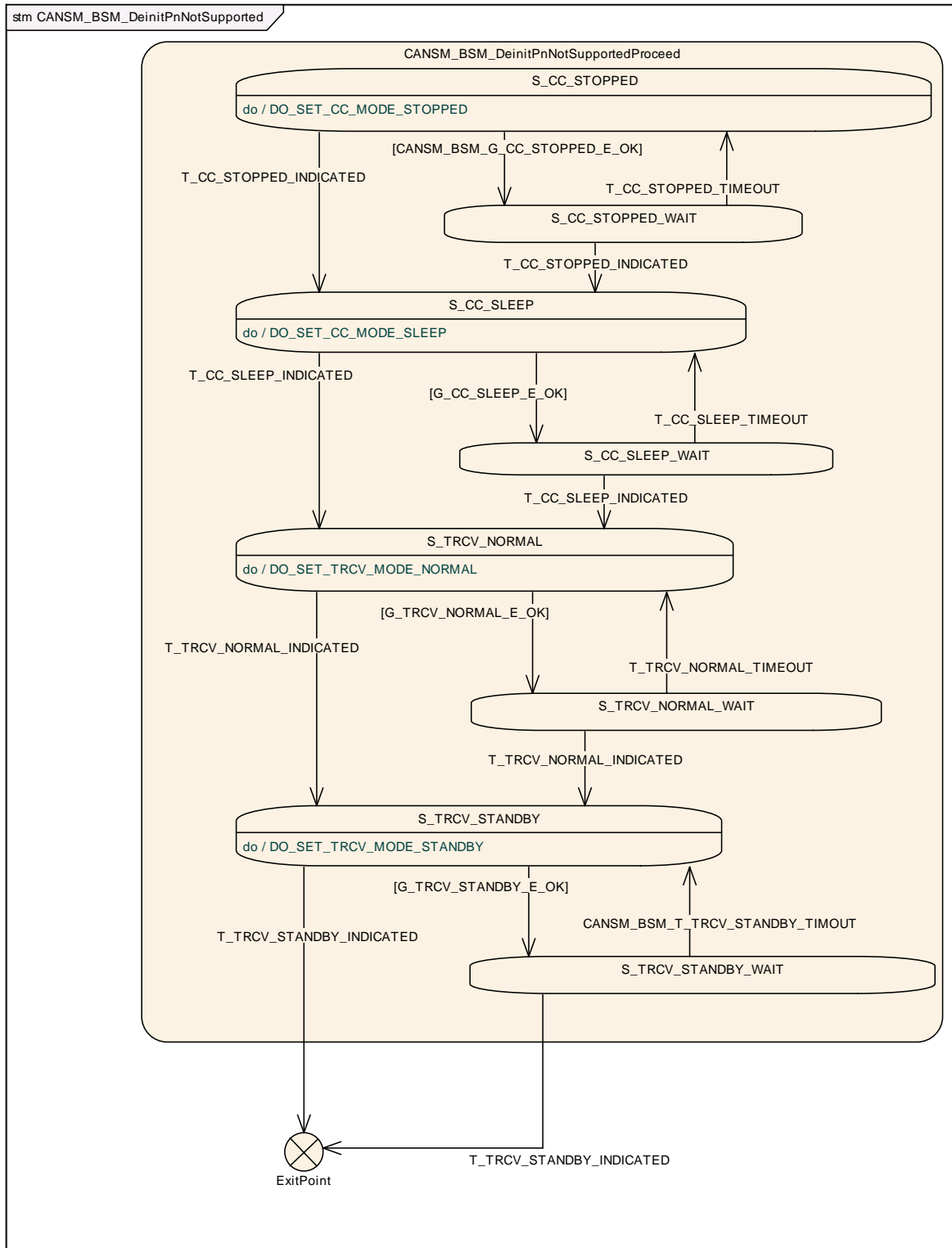


Figure 7-5) of the CAN network with `T_CC_SLEEP_INDICATED`.] (SRS\_Can\_01142, SRS\_Can\_01145)

7.2.20.4.8 Trigger: T\_CC\_SLEEP\_TIMEOUT

[SWS\_CanSM\_00471] [ After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to ECUC\_CanSM\_00336) for all supposed controller sleep mode indications (ref. to SWS\_CanSM\_00470), this condition shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

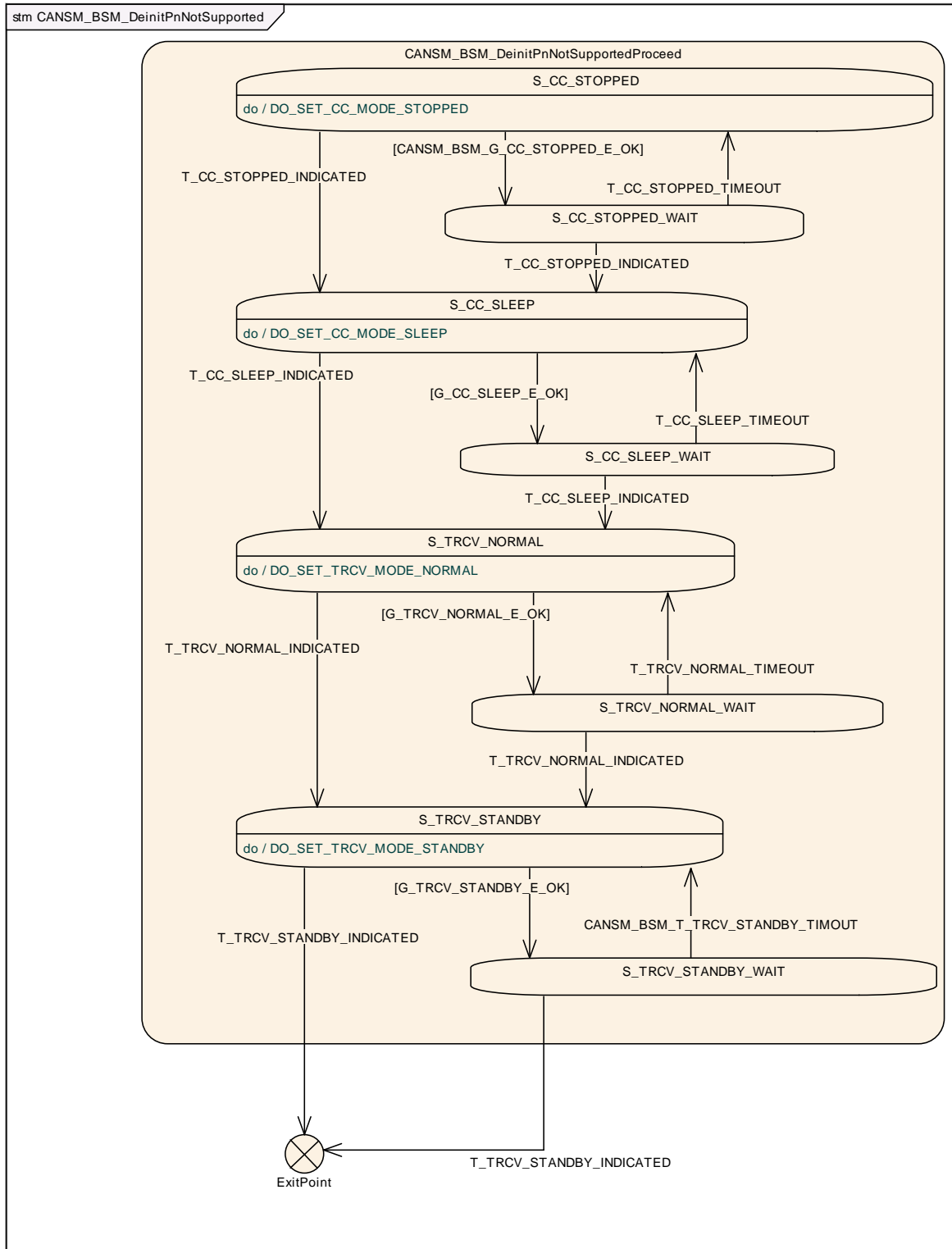


Figure 7-5) of the respective network with  
T\_CC\_SLEEP\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### **7.2.20.4.9 State operation to do in: S\_TRCV\_NORMAL**

**[SWS\_CanSM\_00472]** [ If for the CAN network a CAN Transceiver is configured  
(ref. to [ECUC\\_CanSM\\_00137](#)), then as long the sub state machine  
CANSM\_BSM\_DeinitPnNotSupported (ref. to

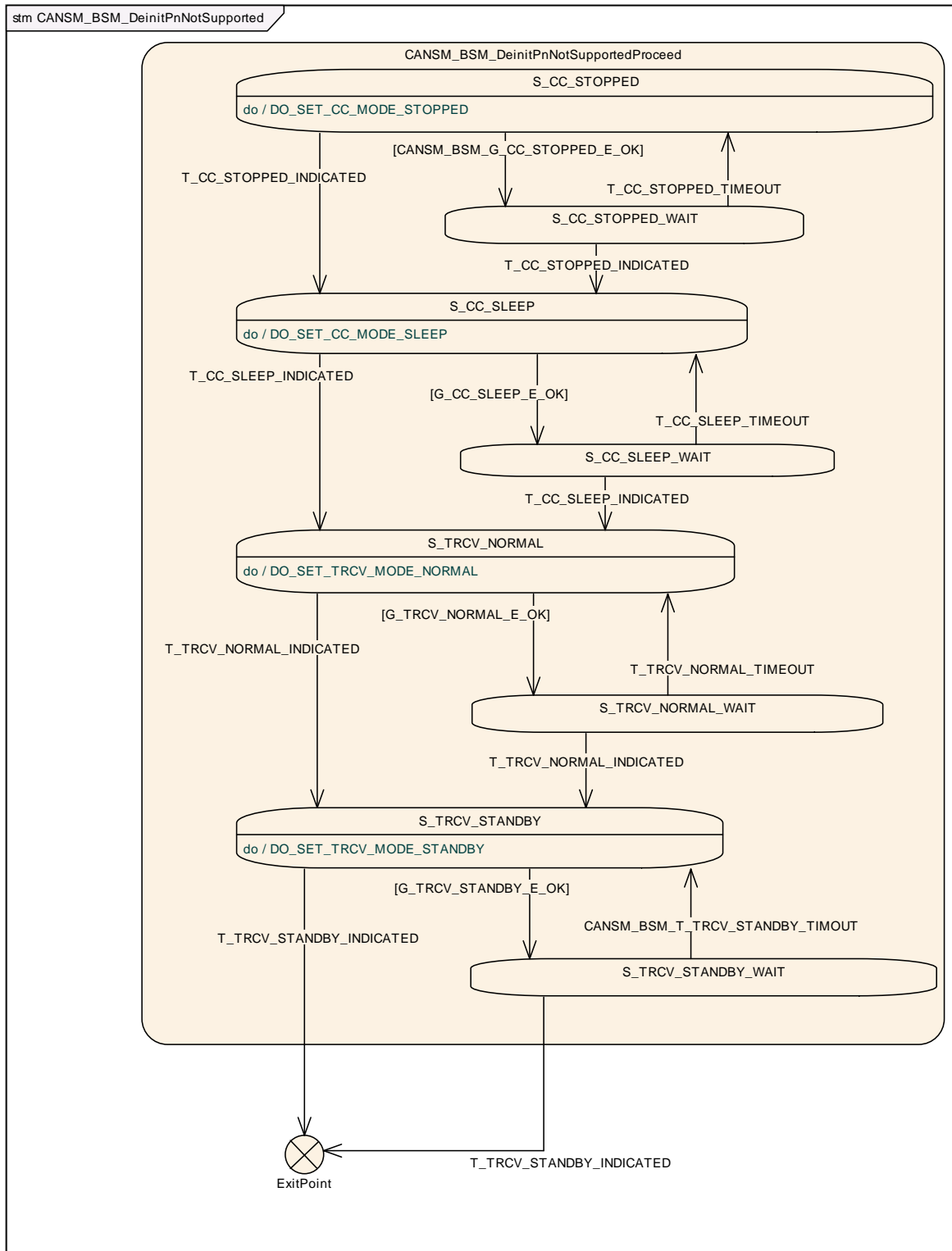


Figure 7-5) is in the state `S_TRCV_NORMAL`, the CanSM module shall operate the do action `DO_SET_TRCV_MODE_NORMAL` and therefore repeat for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) the API request `CanIf_SetTrcvMode` (ref. to chapter 8.5.1) with `TransceiverMode` equal to `CANTRCV_TRCVMODE_NORMAL`.] (SRS\_Can\_01142, SRS\_Can\_01145)



7.2.20.4.10 Guarding condition: G\_TRCV\_NORMAL\_E\_OK

[SWS\_CanSM\_00473] [ The guarding condition G\_TRCV\_NORMAL\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

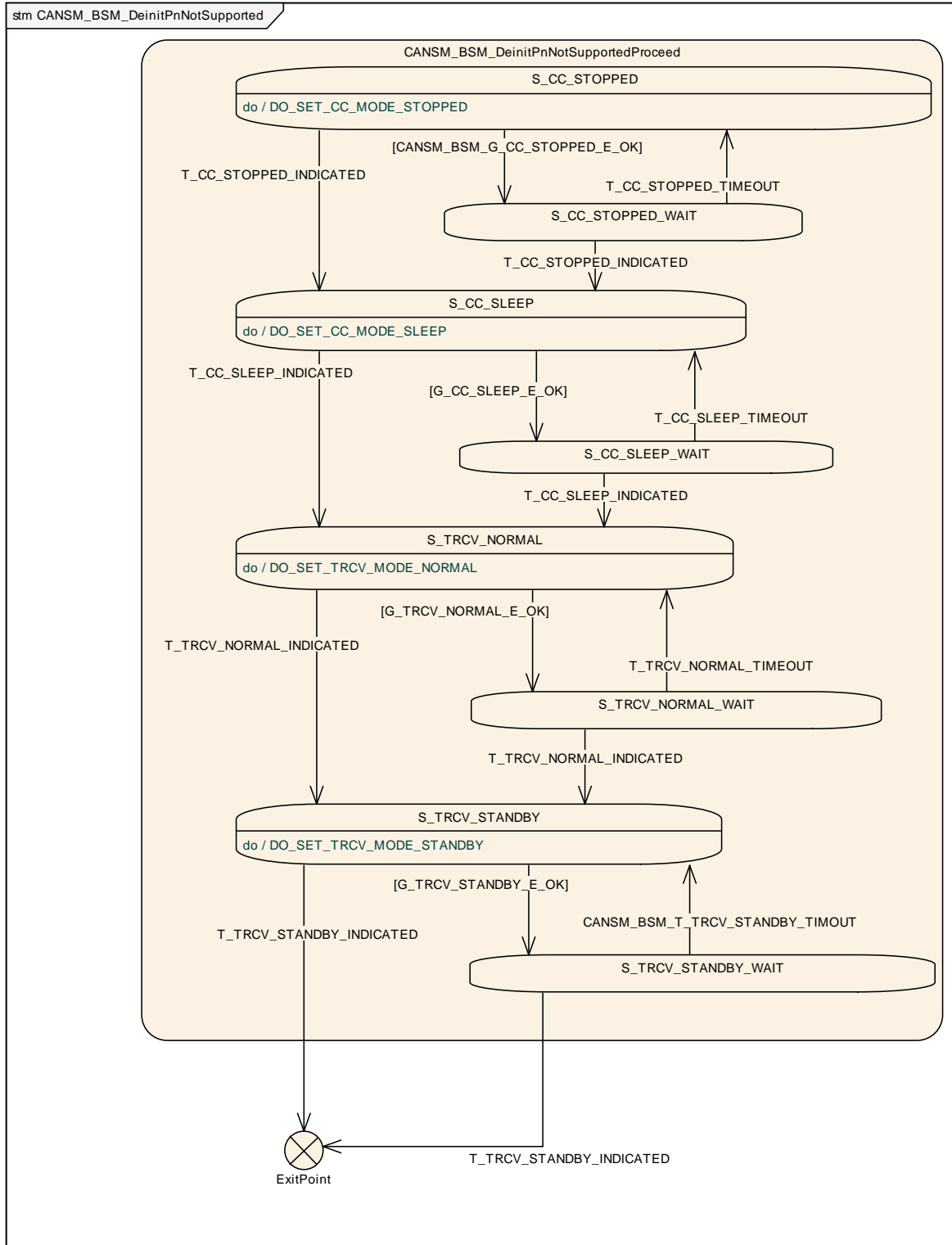


Figure 7-5) shall be passed, if the API call of [SWS\\_CanSM\\_00472](#) has returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

7.2.20.4.11 Trigger: T\_TRCV\_NORMAL\_INDICATED

**[SWS\_CanSM\_00474]** [ If CanSM module has got the CANTRCV\_TRCVMODE\_NORMAL mode indication (ref. to [SWS\\_CanSM\\_00399](#)) for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) after the respective request (ref. to [SWS\\_CanSM\\_00472](#)), this shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

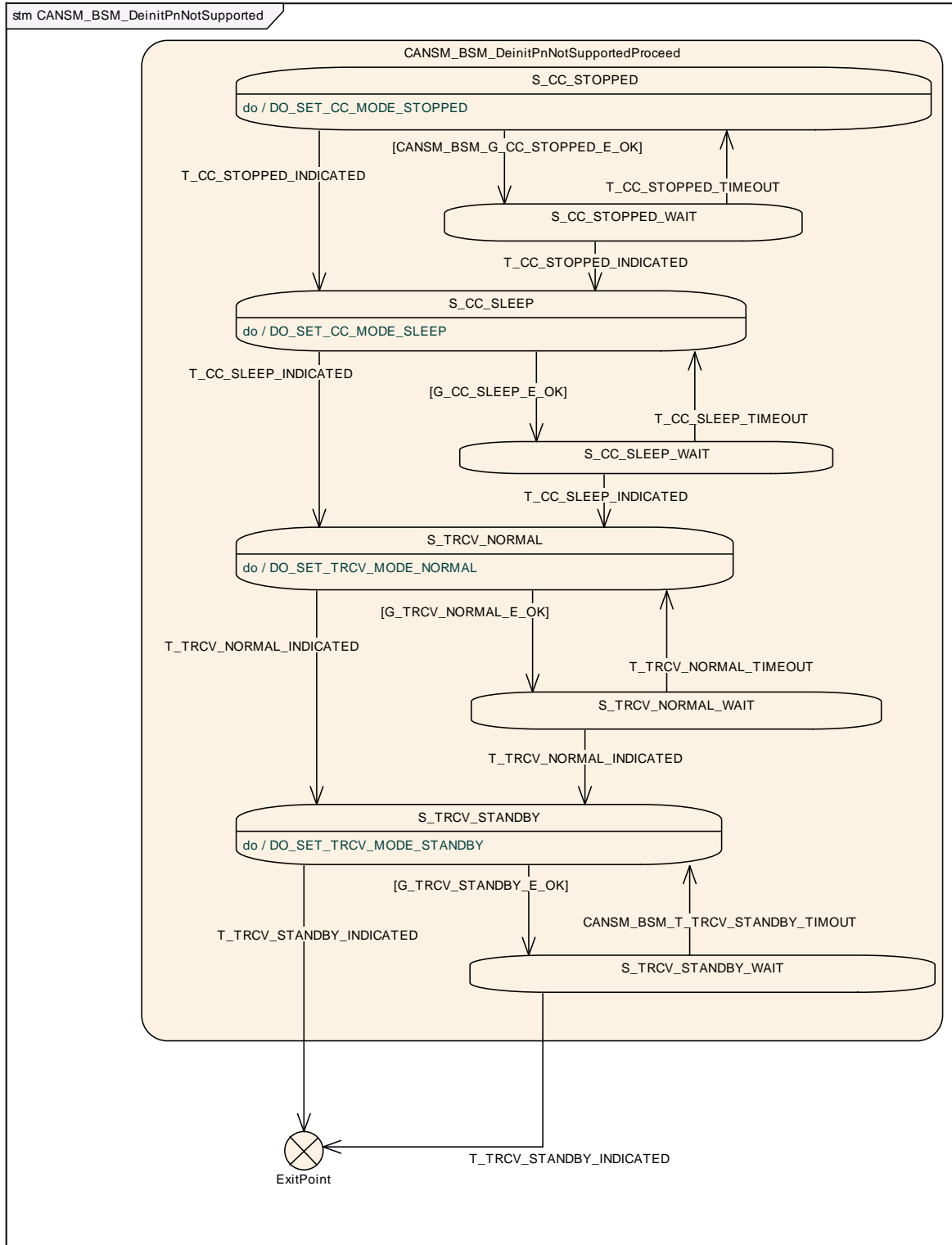


Figure 7-5) of the CAN network with  
T\_TRCV\_NORMAL\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**[SWS\_CanSM\_00556]** [ If no CAN Transceiver is configured for the CAN network, then this shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

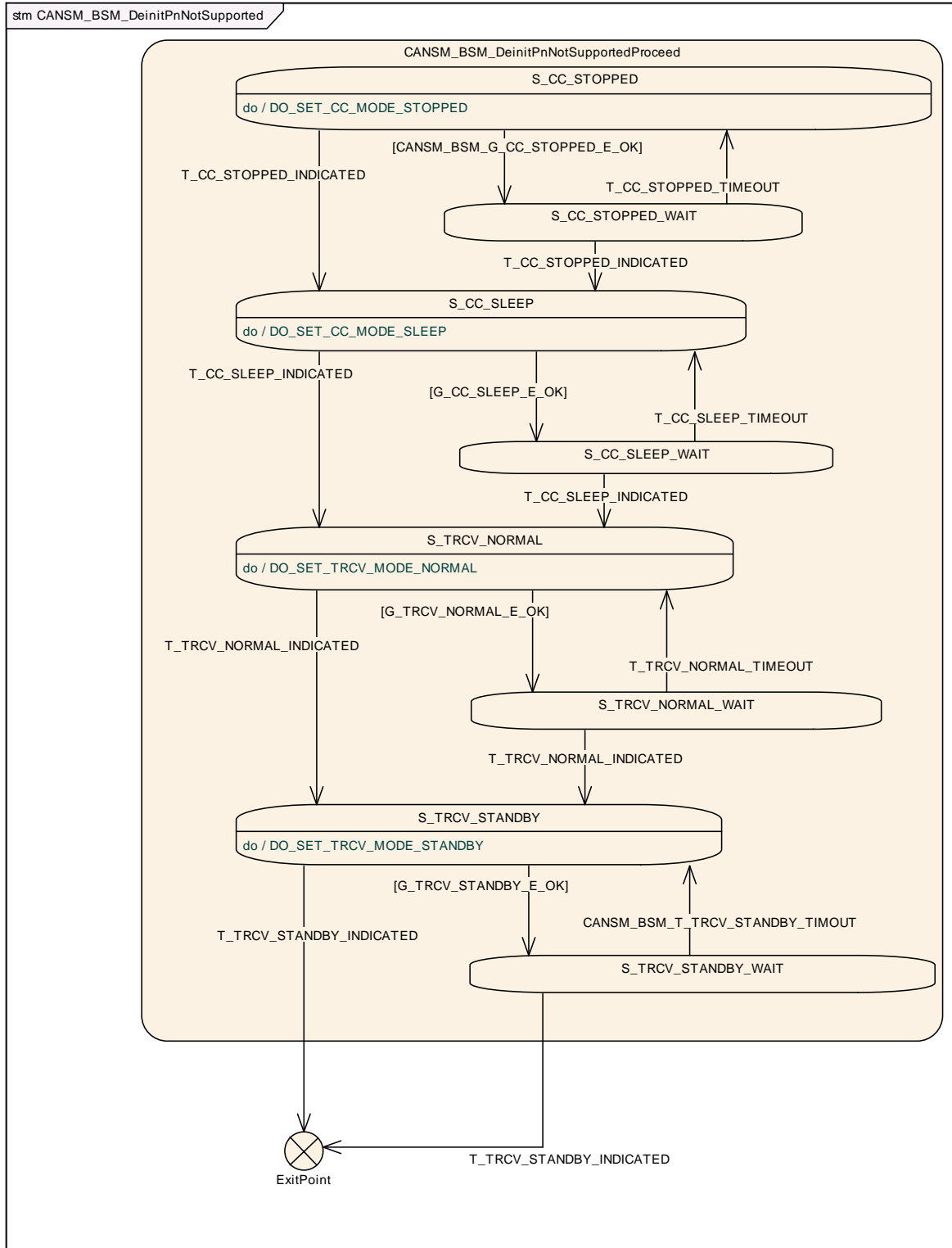


Figure 7-5) of the CAN network in the state `S_TRCV_NORMAL` with  
`T_TRCV_NORMAL_INDICATED`.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.4.12 Trigger: `T_TRCV_NORMAL_TIMEOUT`

**[SWS\_CanSM\_00475]** [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for the supposed transceiver normal indication (ref. to [SWS\\_CanSM\\_00474](#)), this condition shall trigger the sub state machine `CANSM_BSM_DeinitPnNotSupported` (ref. to

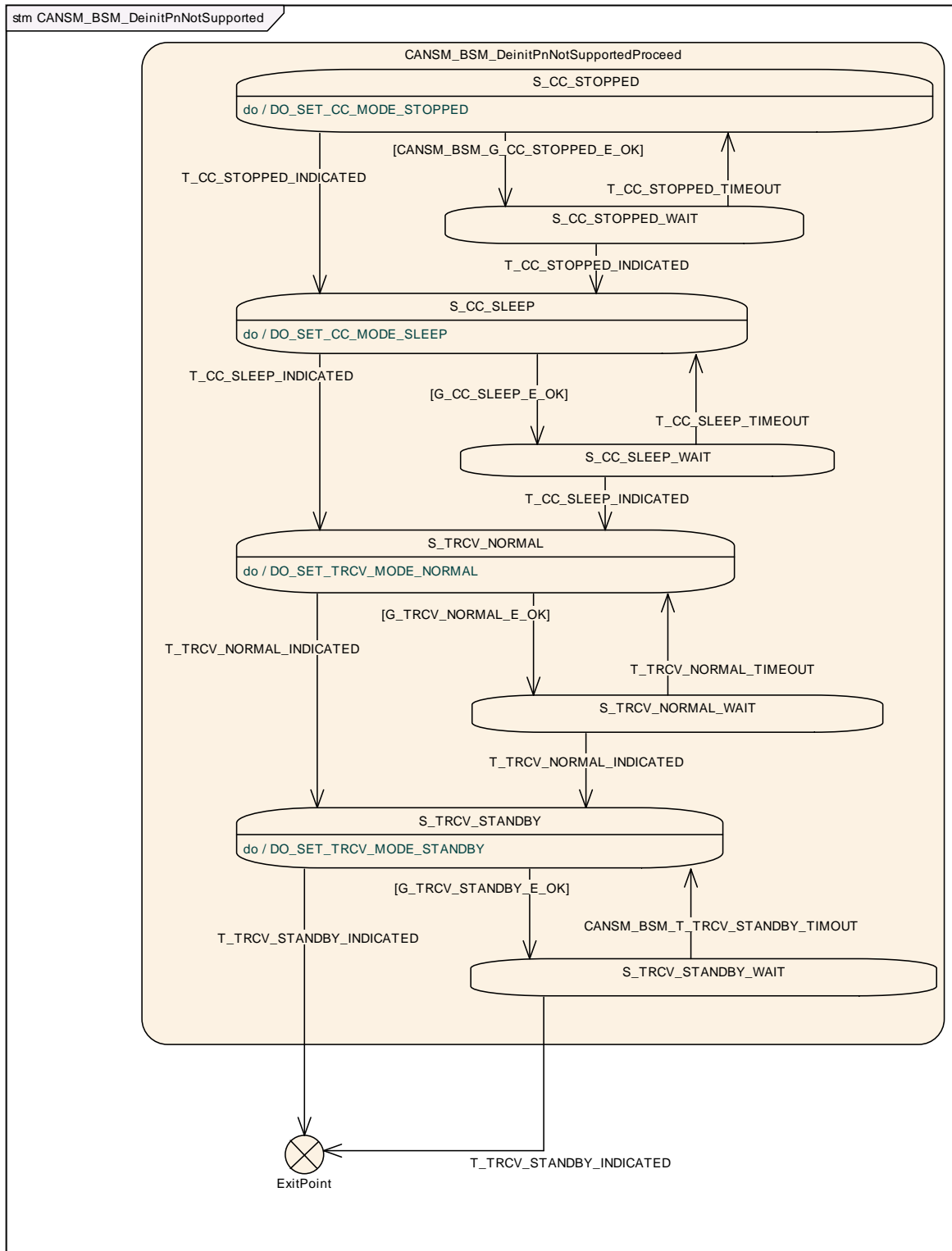


Figure 7-5) of the respective network with  
T\_TRCV\_NORMAL\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.20.4.13 State operation to do in: S\_TRCV\_STANDBY**

**[SWS\_CanSM\_00476]** [ If for the CAN network a CAN Transceiver is configured (ref. to [ECUC\\_CanSM\\_00137](#)), then as long the sub state machine

CANSM\_BSM\_DeinitPnNotSupported (ref. to

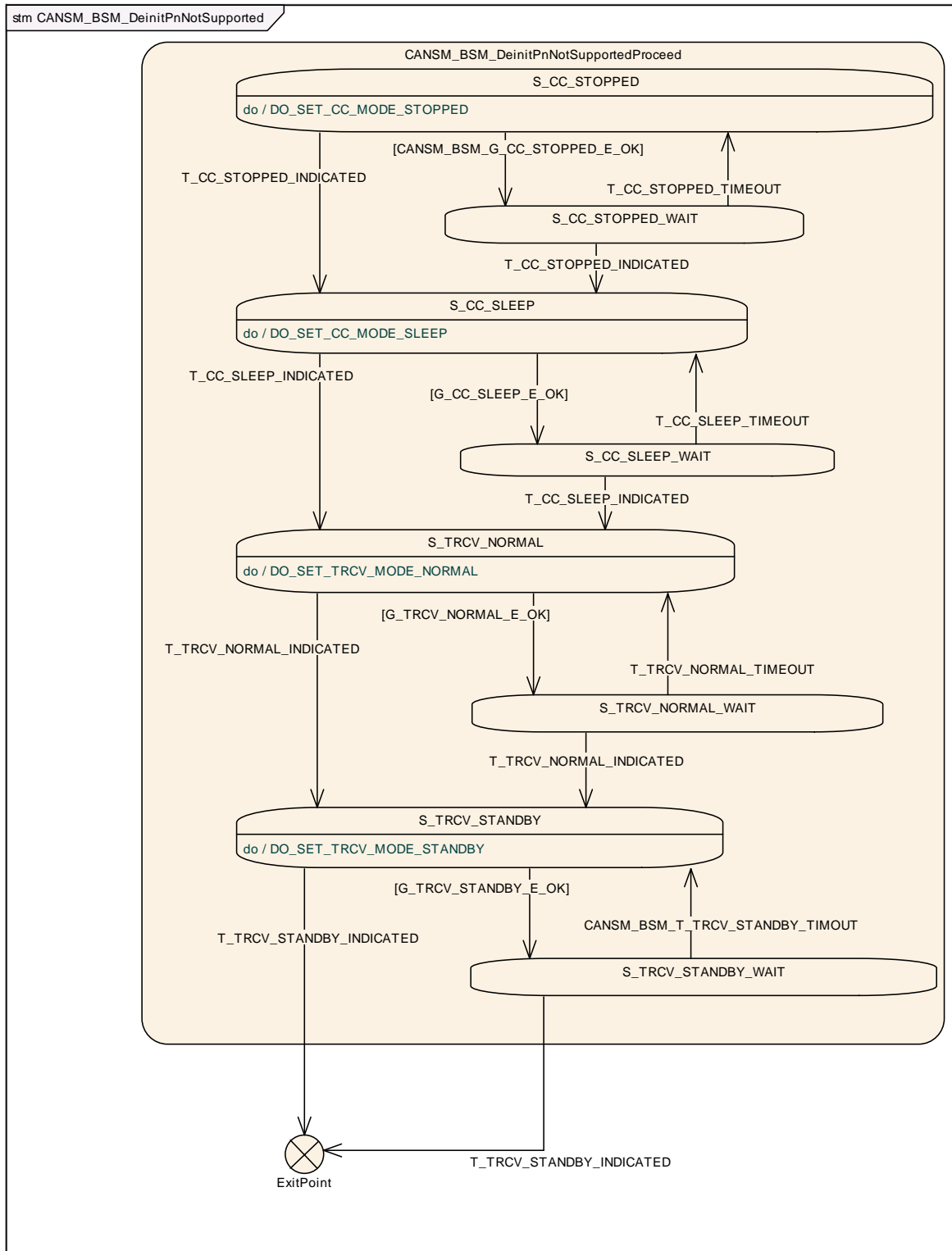


Figure 7-5) is in the state S\_TRCV\_STANDBY, the CanSM module shall operate the do action DO\_SET\_TRCV\_MODE\_STANDBY and therefore repeat for the configured CAN Transceiver of the CAN network (ref. to [ECUC CanSM\\_00137](#)) the API request CanIf\_SetTrcvMode (ref. to chapter 8.5.1) with TransceiverMode equal to CANTRCV\_TRCVMODE\_STANDBY.] (SRS\_Can\_01142, SRS\_Can\_01145)

7.2.20.4.14 Guarding condition: G\_TRCV\_STANDBY\_E\_OK

[SWS\_CanSM\_00477] [ The guarding condition G\_TRCV\_STANDBY\_E\_OK of the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

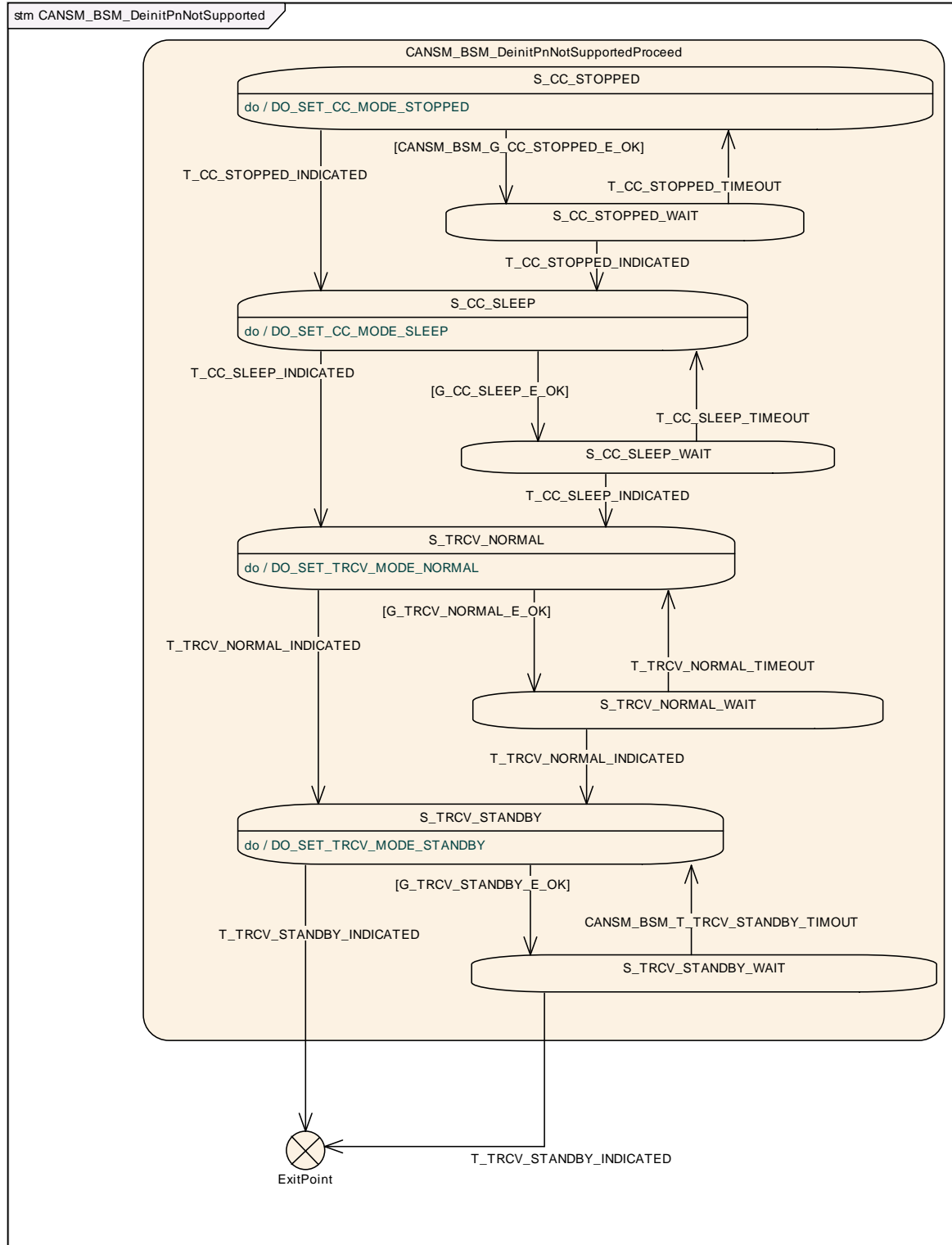


Figure 7-5) shall be passed, if the API call of [SWS\\_CanSM\\_00476](#) has returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145)

7.2.20.4.15 Trigger: T\_TRCV\_STANDBY\_INDICATED

**[SWS\_CanSM\_00478]** [ If CanSM module has got the CANTRCV\_TRCVMODE\_STANDBY mode indication (ref. to [SWS\\_CanSM\\_00399](#)) for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) after the respective request (ref. to [SWS\\_CanSM\\_00476](#)), this shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

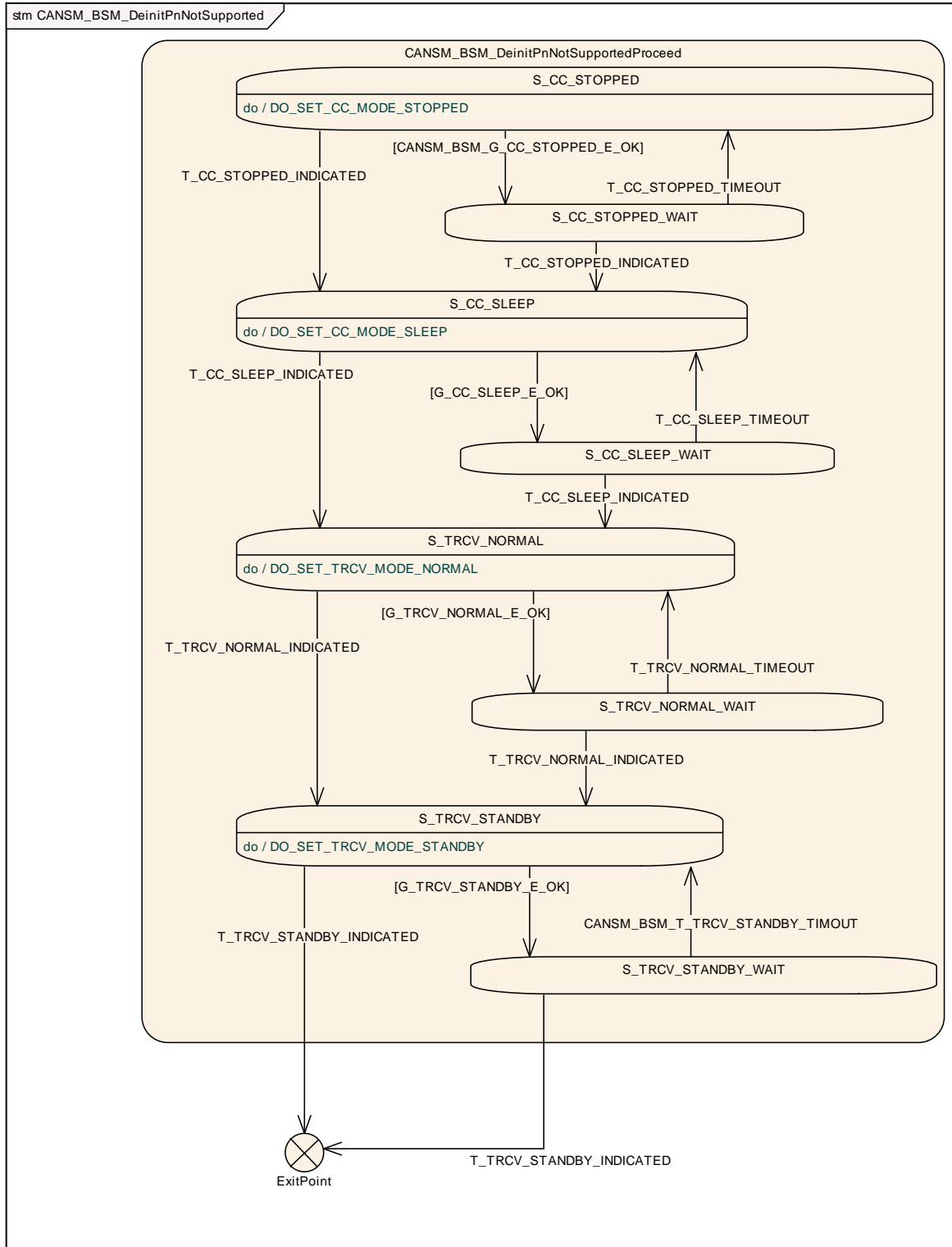




Figure 7-5) of the CAN network with  
T\_TRCV\_STANDBY\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

**[SWS\_CanSM\_00557]** | If no CAN Transceiver is configured for the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)), then this shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

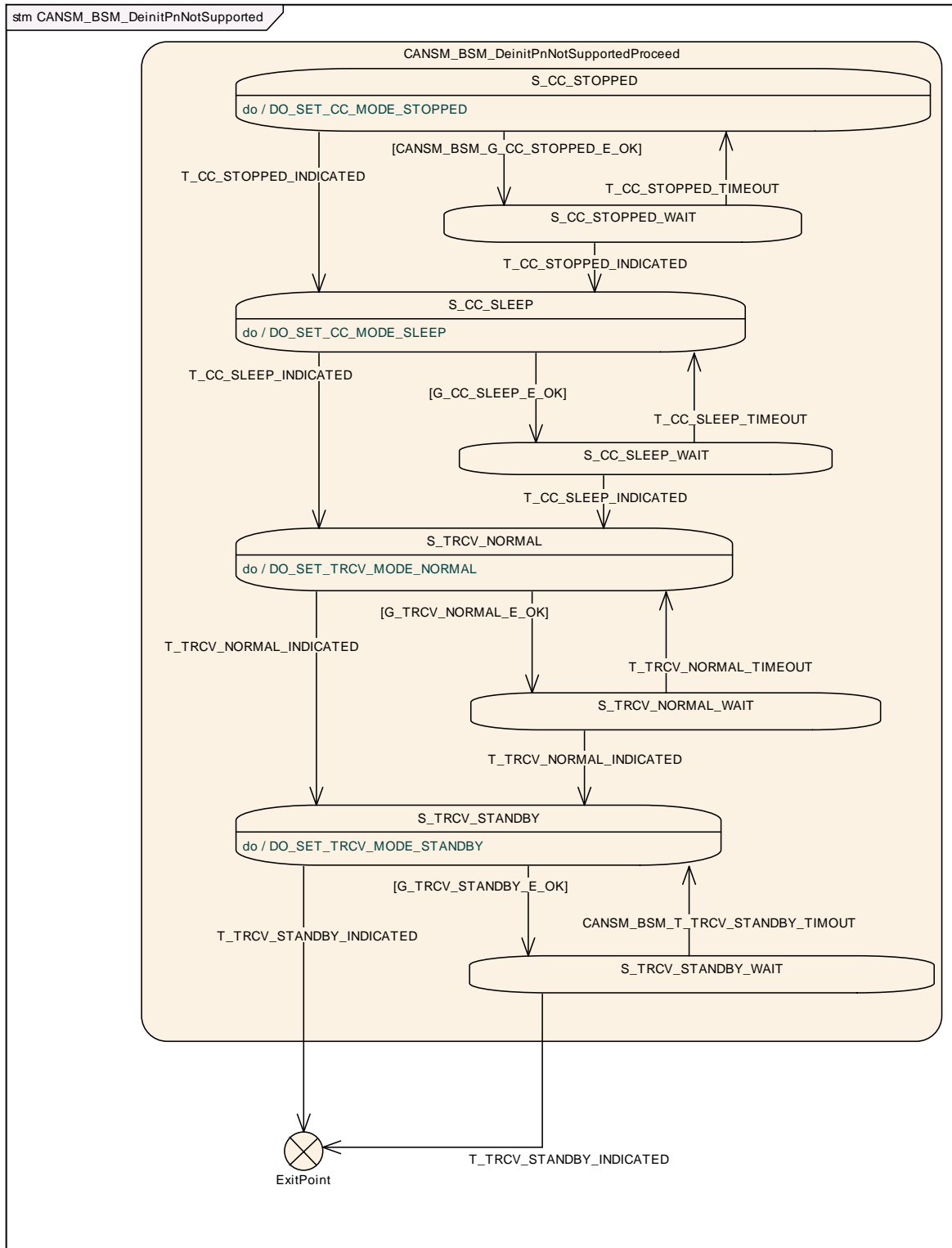


Figure 7-5) of the CAN network in the state S\_TRCV\_STANDBY with T\_TRCV\_STANDBY\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145)

#### 7.2.20.4.16 Trigger: CANSM\_BSM\_T\_TRCV\_STANDBY\_TIMEOUT

**[SWS\_CanSM\_00479]** [After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for the supposed transceiver standby indication (ref. to [SWS\\_CanSM\\_00478](#)), this condition shall trigger the sub state machine CANSM\_BSM\_DeinitPnNotSupported (ref. to

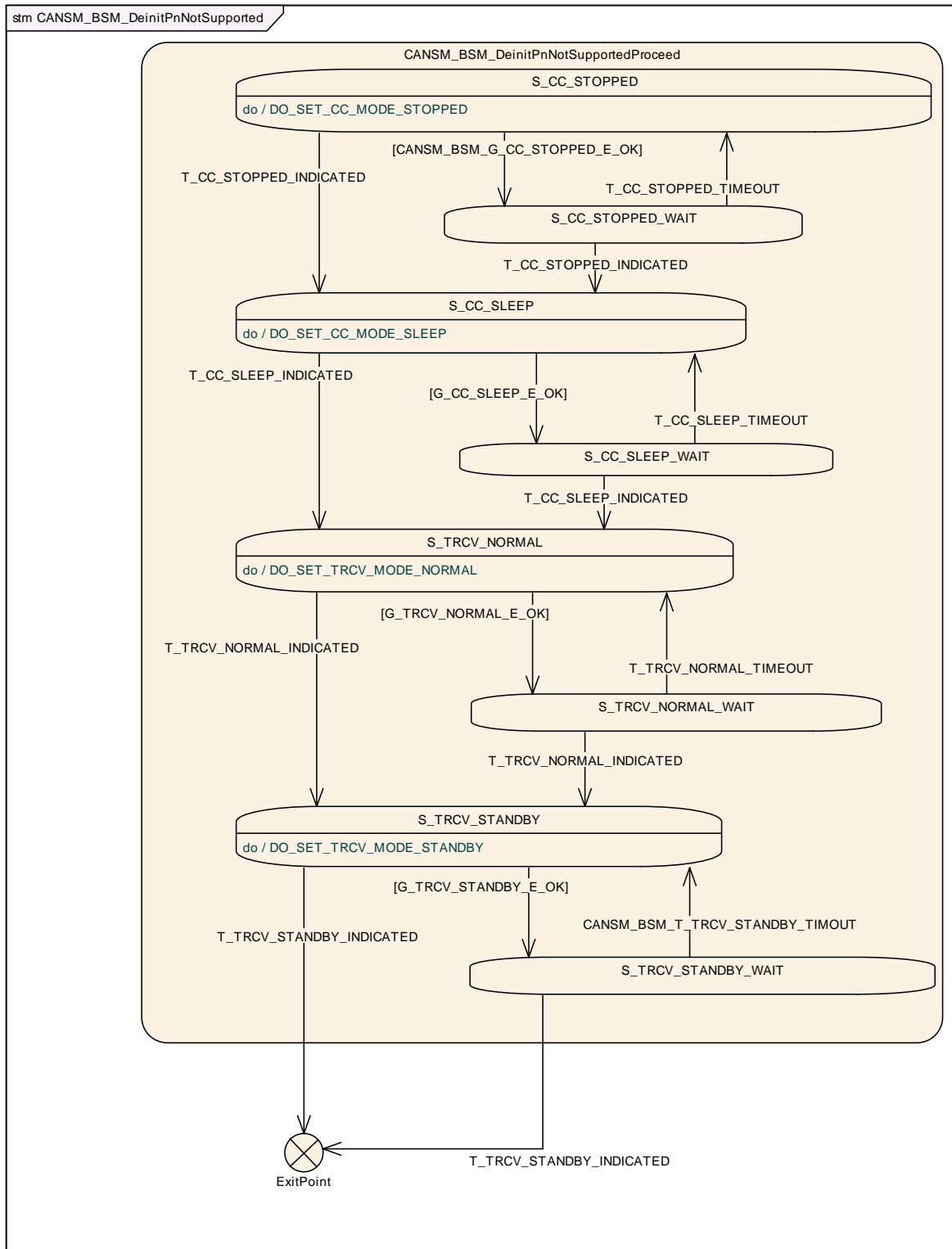
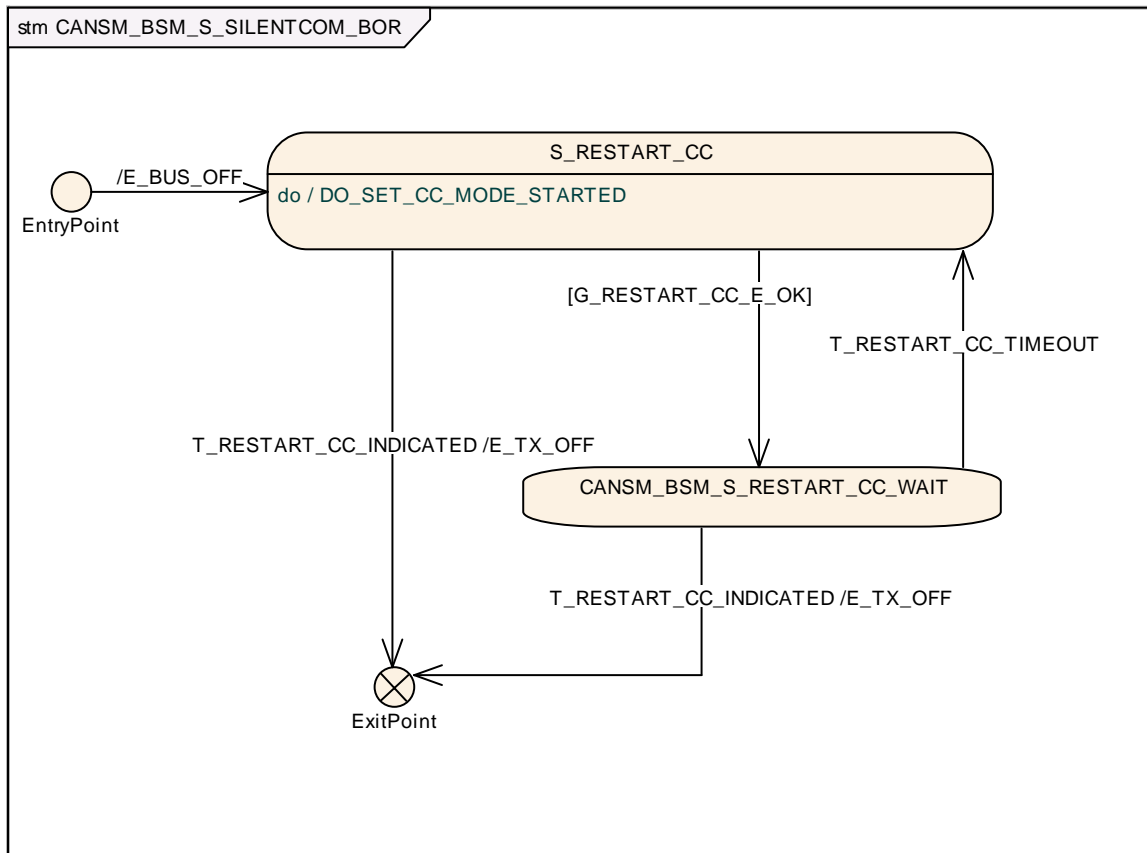


Figure 7-5) of the respective network with  
CANSMBSM\_T\_TRCV\_STANDBY\_TIMEOUT.](SRS\_Can\_01142, SRS\_Can\_01145)

**7.2.21 Sub state machine: CANSM\_BSM\_S\_SILENTCOM\_BOR**



**Figure 7-6: CANSM\_BSM\_S\_SILENTCOM\_BOR, sub state machine of CANSM\_BSM**

**7.2.21.1 Effect: E\_BUS\_OFF**

[SWS\_CanSM\_00605] The effect E\_BUS\_OFF of the sub state machine CANSM\_BSM\_S\_FULLCOM CANSM\_BSM\_S\_SILENTCOM\_BOR (ref. to Figure 7-6) shall invoke Dem\_SetEventStatus (ref. to chapter 8.5.1) with the parameters EventId := CANSM\_E\_BUS\_OFF (ref. to ECUC\_CanSM\_00070) and EventStatus := DEM\_EVENT\_STATUS\_PRE\_FAILED.] (SRS\_BSW\_00422)

**7.2.21.2 State operation: S\_RESTART\_CC**

[SWS\_CanSM\_00604] As long the sub state machine CANSM\_BSM\_S\_SILENTCOM\_BOR (ref. to Figure 7-6) is in the state S\_RESTART\_CC, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STARTED and therefore repeat for all configured CAN controllers of the CAN network (ref. to ECUC\_CanSM\_00141) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STARTED, if the current CAN controller mode (ref. to SWS\_CanSM\_00638) is different.] (SRS\_Can\_01142, SRS\_Can\_01145, SRS\_Can\_01144, SRS\_Can\_01146)

**7.2.21.3 G\_RESTART\_CC\_E\_OK**

[SWS\_CanSM\_00603] The guarding condition G\_RESTART\_CC\_OK of the sub state machine CANSM\_BSM\_S\_SILENTCOM\_BOR (ref. to Figure 7-6) shall be passed,

if all API calls of [SWS CanSM 00604](#) have returned E\_OK.] (SRS\_Can\_01142, SRS\_Can\_01145, SRS\_Can\_01144, SRS\_Can\_01146)

#### 7.2.21.4 Trigger: T\_RESTART\_CC\_INDICATED

[[SWS CanSM 00600](#)] If CanSM module has got all mode indications (ref. to [SWS CanSM 00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC CanSM 00141](#)) after the respective requests to start the CAN controllers of the CAN network (ref. to [SWS CanSM 00604](#)), this shall trigger the sub state CANSM\_BSM\_S\_SILENTCOM\_BOR (ref. to Figure 7-6) of the CAN network with T\_RESTART\_CC\_INDICATED.] (SRS\_Can\_01142, SRS\_Can\_01145, SRS\_Can\_01144, SRS\_Can\_01146)

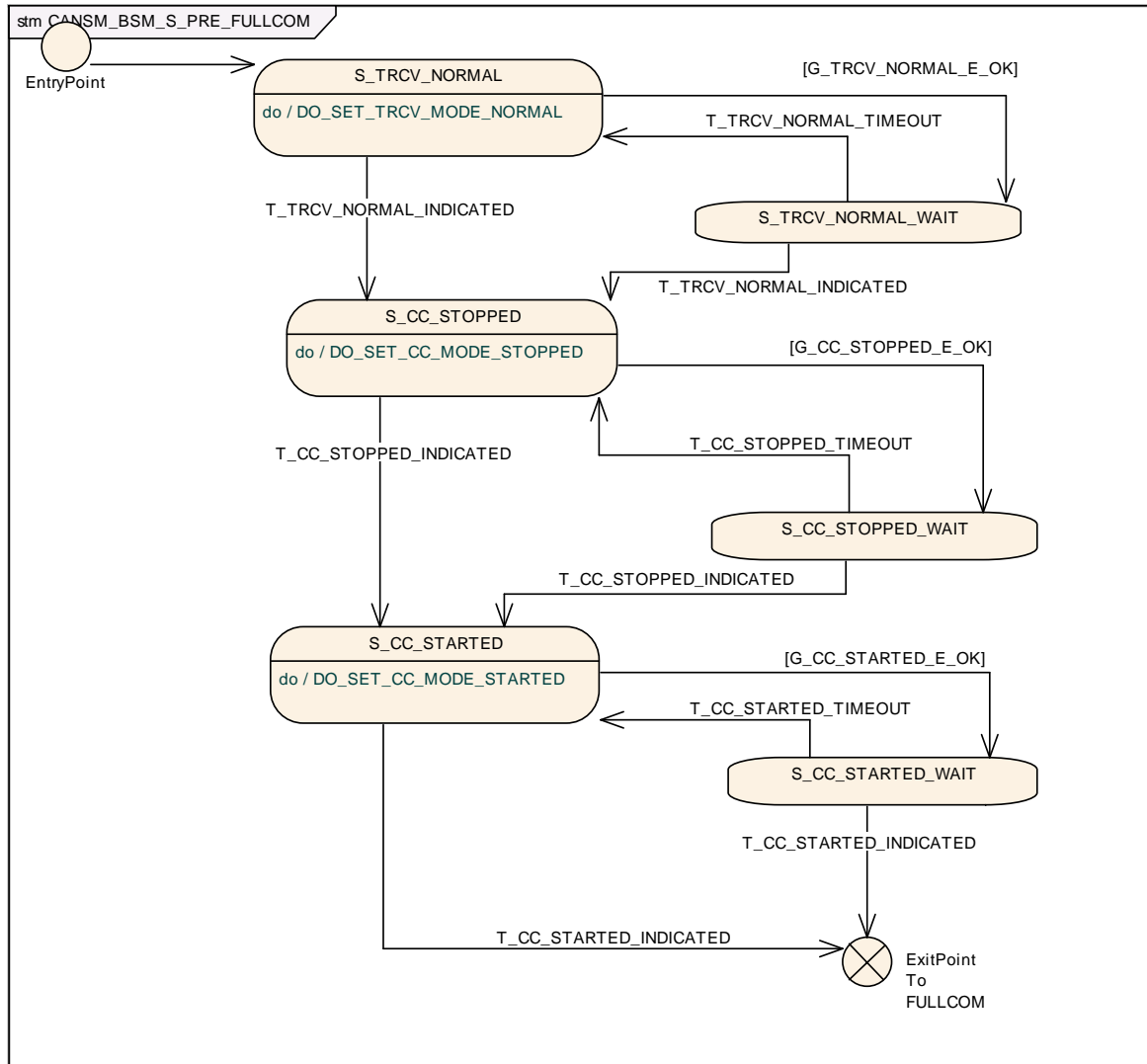
#### 7.2.21.5 T\_RESTART\_CC\_TIMEOUT

[[SWS CanSM 00602](#)] After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC CanSM 00336](#)) for all supposed controller started mode indications (ref. to [SWS CanSM 00600](#)), this condition shall trigger the sub state machine CANSM\_BSM\_S\_SILENTCOM\_BOR (ref. to Figure 7-6) of the respective network with T\_RESTART\_CC\_TIMEOUT.] (SRS\_Can\_01142, SRS\_Can\_01145, SRS\_Can\_01144, SRS\_Can\_01146)

#### 7.2.21.6 Effect: E\_TX\_OFF

The effect E\_TX\_OFF shall do nothing (default PDU mode after restart of CAN controller is already TX OFF, ref. to CanIf SWS).

**7.2.22 Sub state machine: CANSM\_BSM\_S\_PRE\_FULLCOM**



**Figure 7-7: CANSM\_BSM\_S\_PRE\_FULLCOM, sub state machine of CANSM\_BSM**

**7.2.22.1 State operation to do in: S\_TRCV\_NORMAL**

**[SWS\_CanSM\_00483]** [ If for the CAN network a CAN Transceiver is configured (ref. to [ECUC CanSM 00137](#)), then as long the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

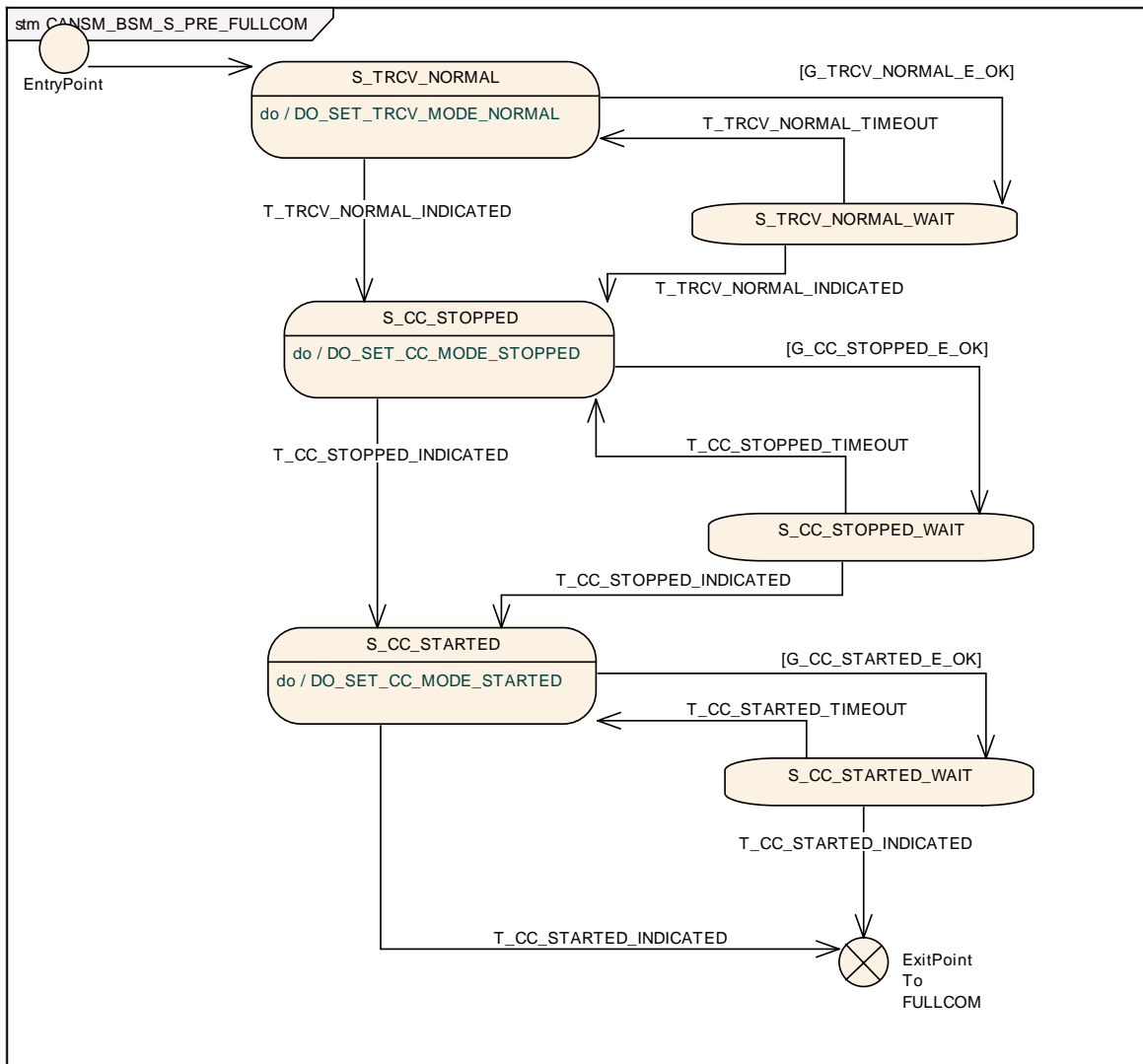


Figure 7-7) is in the state `S_TRCV_NORMAL`, the CanSM module shall operate the do action `DO_SET_TRCV_MODE_NORMAL` and therefore repeat for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) the API request `CanIf_SetTrcvMode` (ref. to chapter 8.5.1) with `TransceiverMode` equal to `CANTRCV_TRCVMODE_NORMAL.`] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.2 Guarding condition: G\_TRCV\_NORMAL\_E\_OK**

**[SWS\_CanSM\_00484]** [ The guarding condition `G_TRCV_NORMAL_E_OK` of the sub state machine `CANSM_BSM_S_PRE_FULLCOM` (ref. to

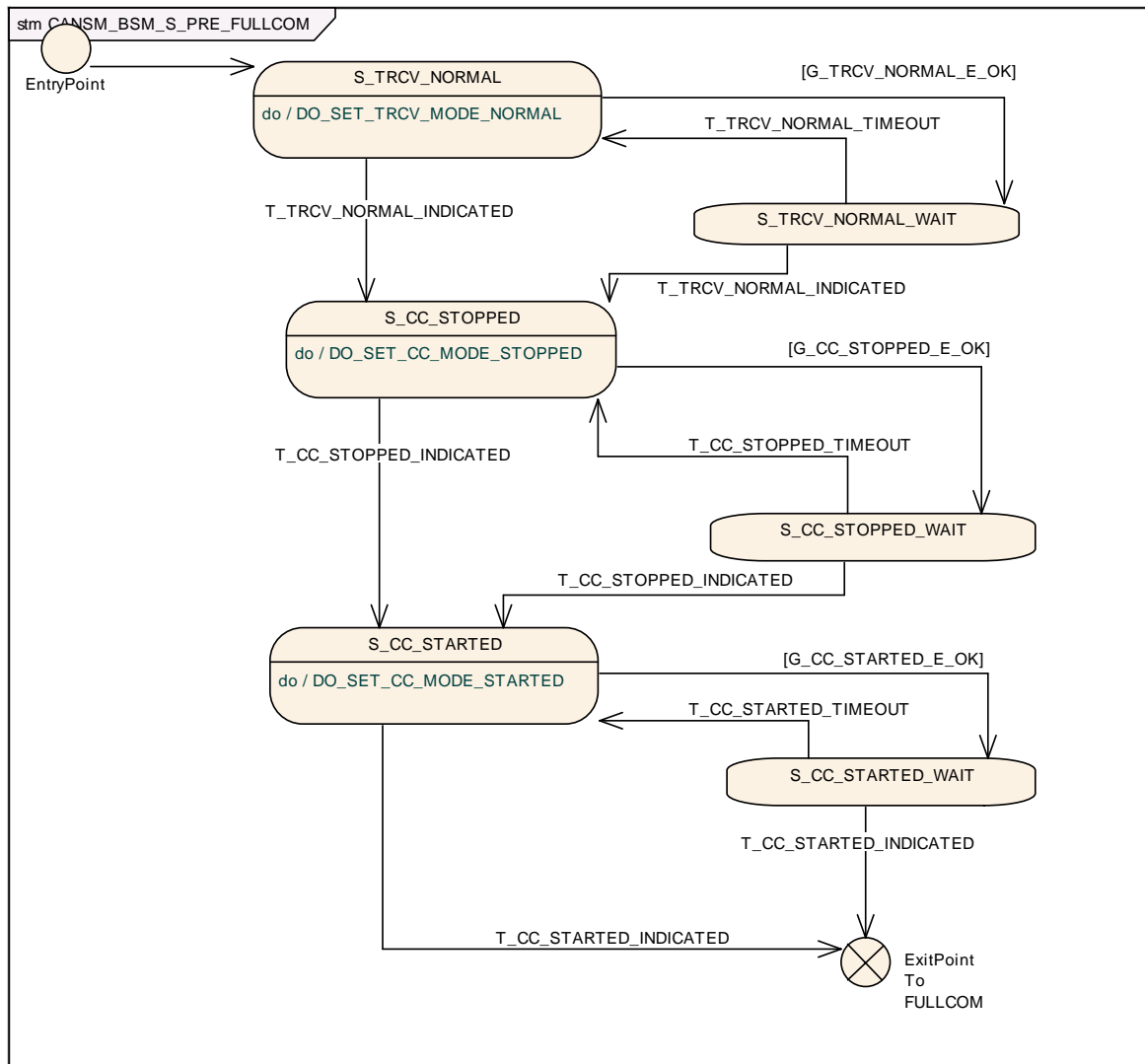


Figure 7-7) shall be passed, if the API call of [SWS CanSM\\_00483](#) has returned E\_OK.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.3 Trigger: T\_TRCV\_NORMAL\_INDICATED**

**[SWS\_CanSM\_00485]** [ If CanSM module has got the CANTRCV\_TRCVMODE\_NORMAL mode indication (ref. to [SWS\\_CanSM\\_00399](#)) for the configured CAN Transceiver of the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)) after the respective request (ref. to [SWS\\_CanSM\\_00483](#)), this shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to



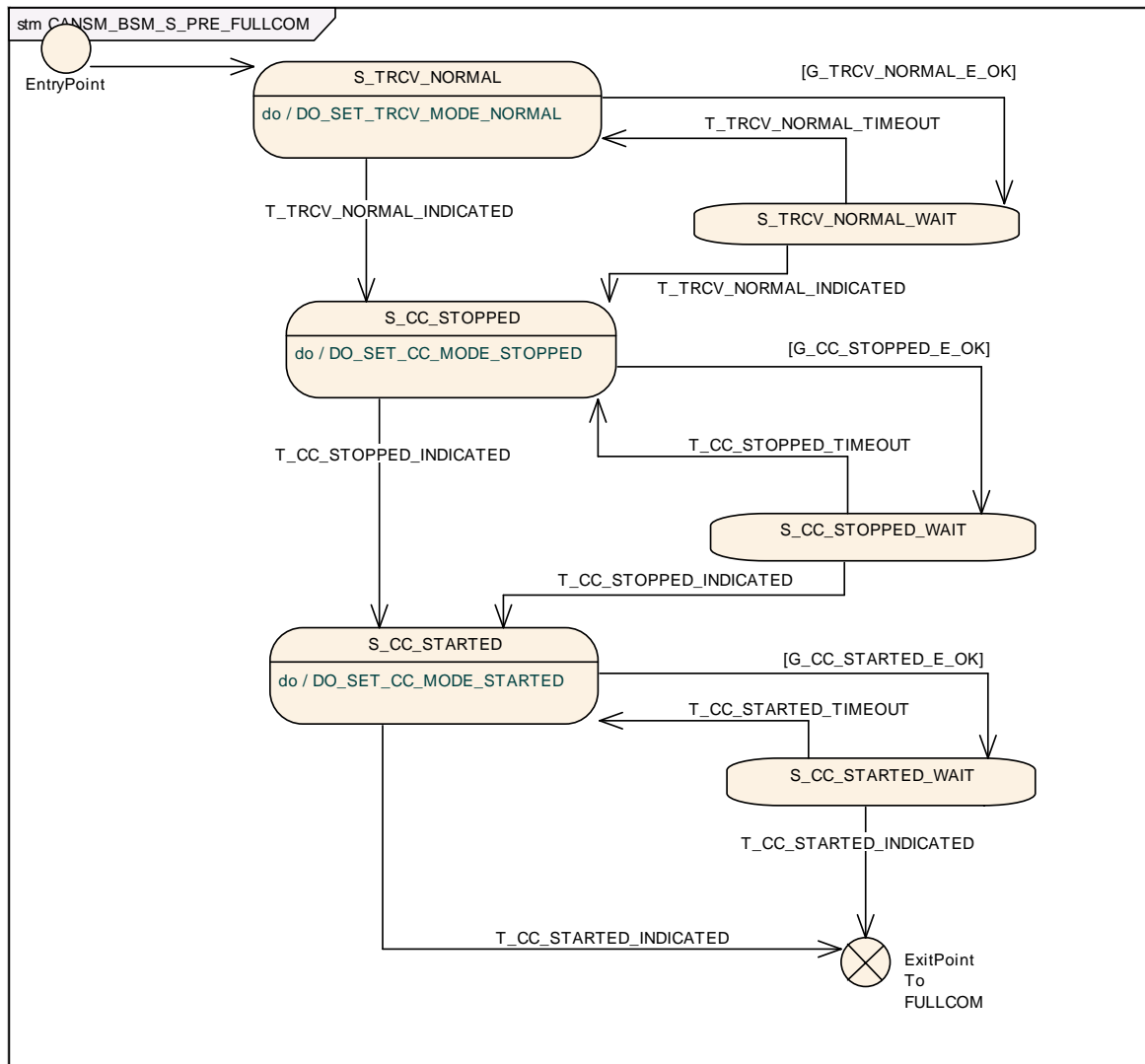


Figure 7-7) of the CAN network with  
T\_TRCV\_NORMAL\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00558]** [ If no CAN Transceiver is configured for the CAN network (ref. to [ECUC\\_CanSM\\_00137](#)), then this shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

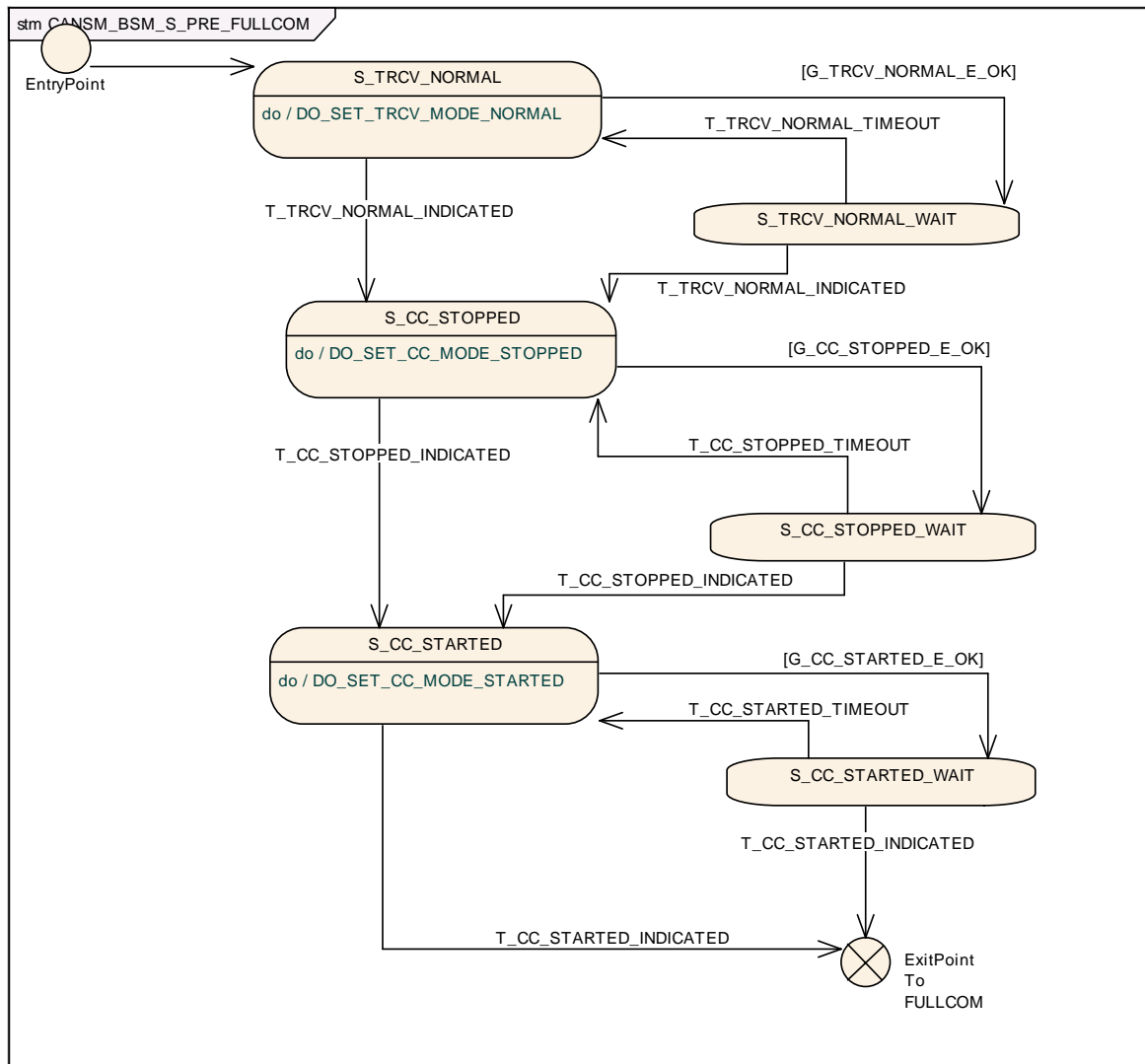


Figure 7-7) of the CAN network in the state S\_TRCV\_NORMAL with T\_TRCV\_NORMAL\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.4 Trigger: T\_TRCV\_NORMAL\_TIMEOUT**

**[SWS\_CanSM\_00486]** [ After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for the supposed transceiver normal indication (ref. to [SWS\\_CanSM\\_00485](#)), this condition shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

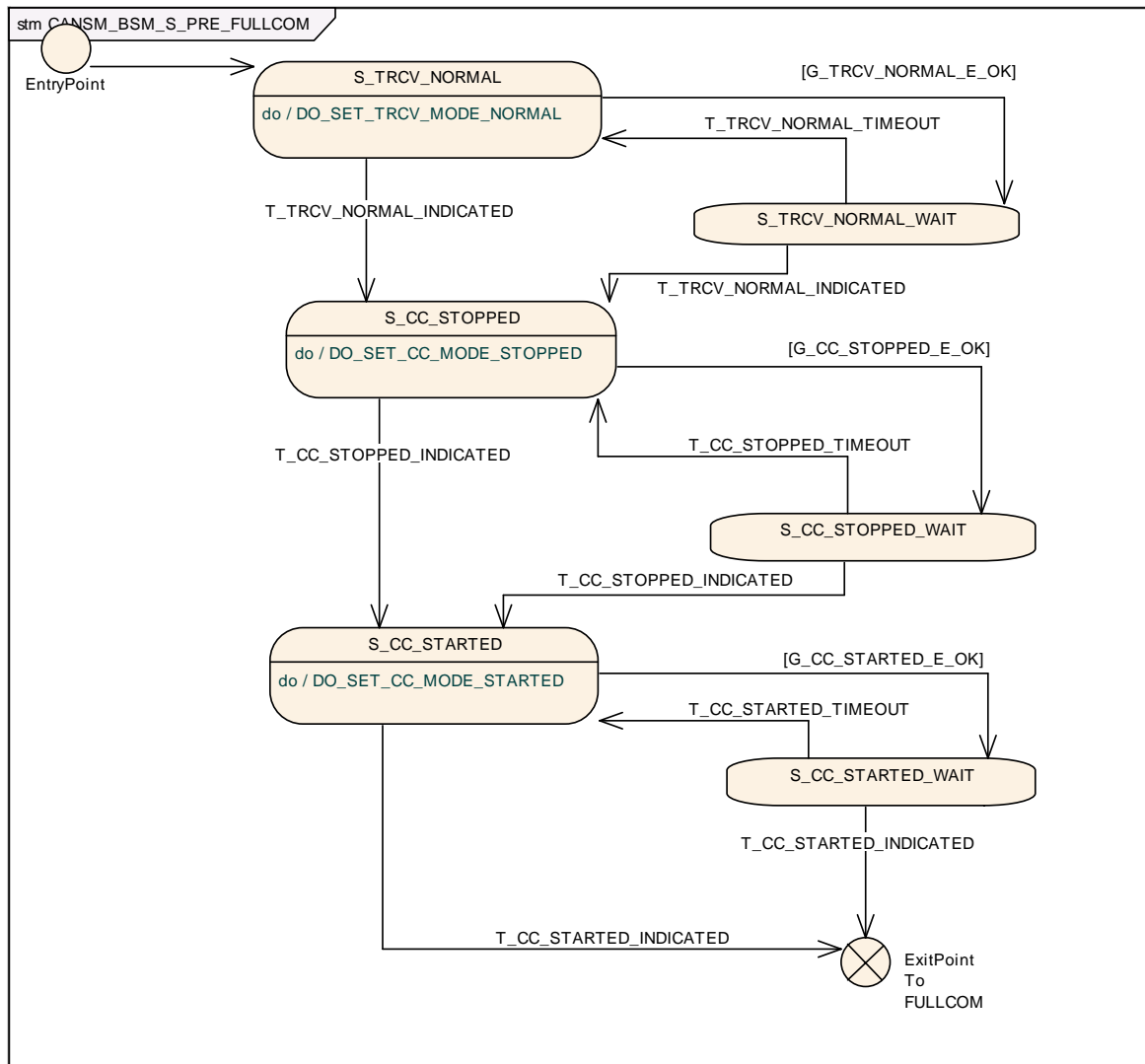


Figure 7-7) of the respective network with  
T\_TRCV\_NORMAL\_TIMEOUT.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.5 State operation to do in: S\_CC\_STOPPED**

[SWS\_CanSM\_00487] [ As long the sub state machine  
CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

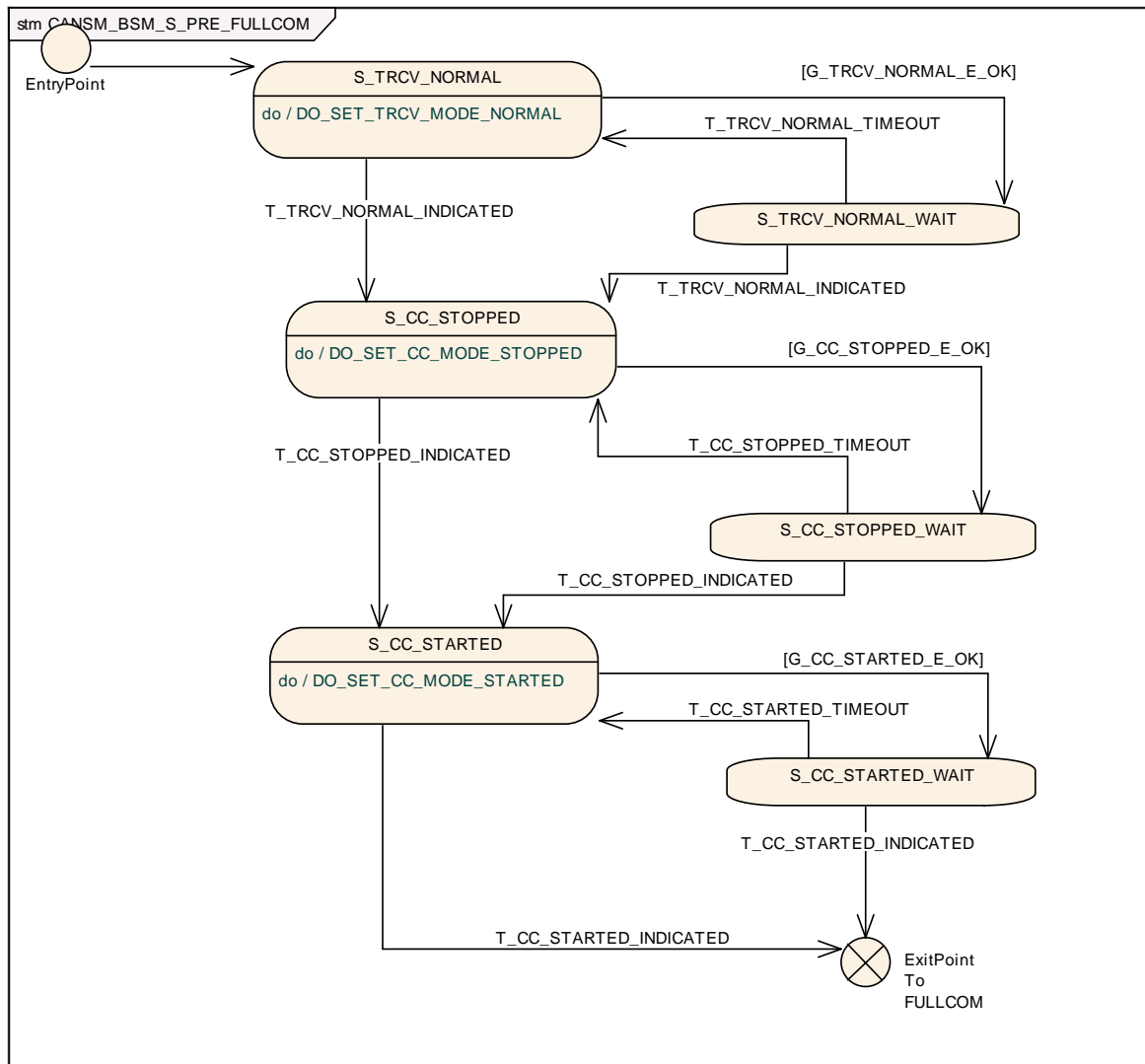


Figure 7-7) is in the state `S_CC_STOPPED`, the CanSM module shall operate the do action `DO_SET_CC_MODE_STOPPED` and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request `CanIf_SetControllerMode` (ref. to chapter 8.5.1) with `ControllerMode` equal to `CAN_CS_STOPPED`, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.6 Guarding condition: G\_CC\_STOPPED\_OK**

**[SWS\_CanSM\_00488]** [ The guarding condition `G_CC_STOPPED_OK` of the sub state machine `CANSM_BSM_S_PRE_FULLCOM` (ref. to

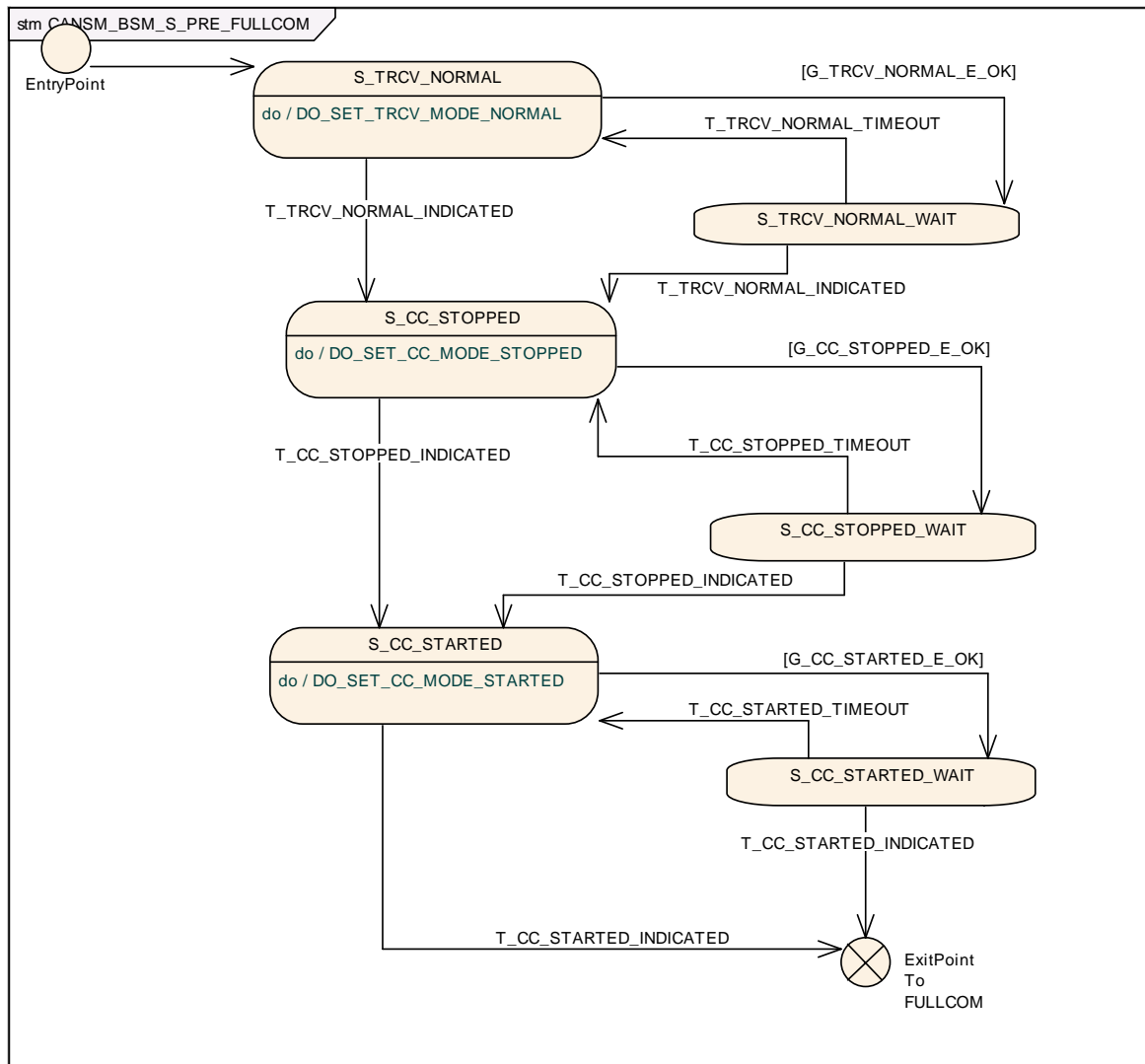


Figure 7-7) shall be passed, if all API calls of [SWS CanSM\\_00487](#) have returned E\_OK.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.7 Trigger: T\_CC\_STOPPED\_INDICATED**

**[SWS\_CanSM\_00489]** [ If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to stop the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00487](#)), this shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

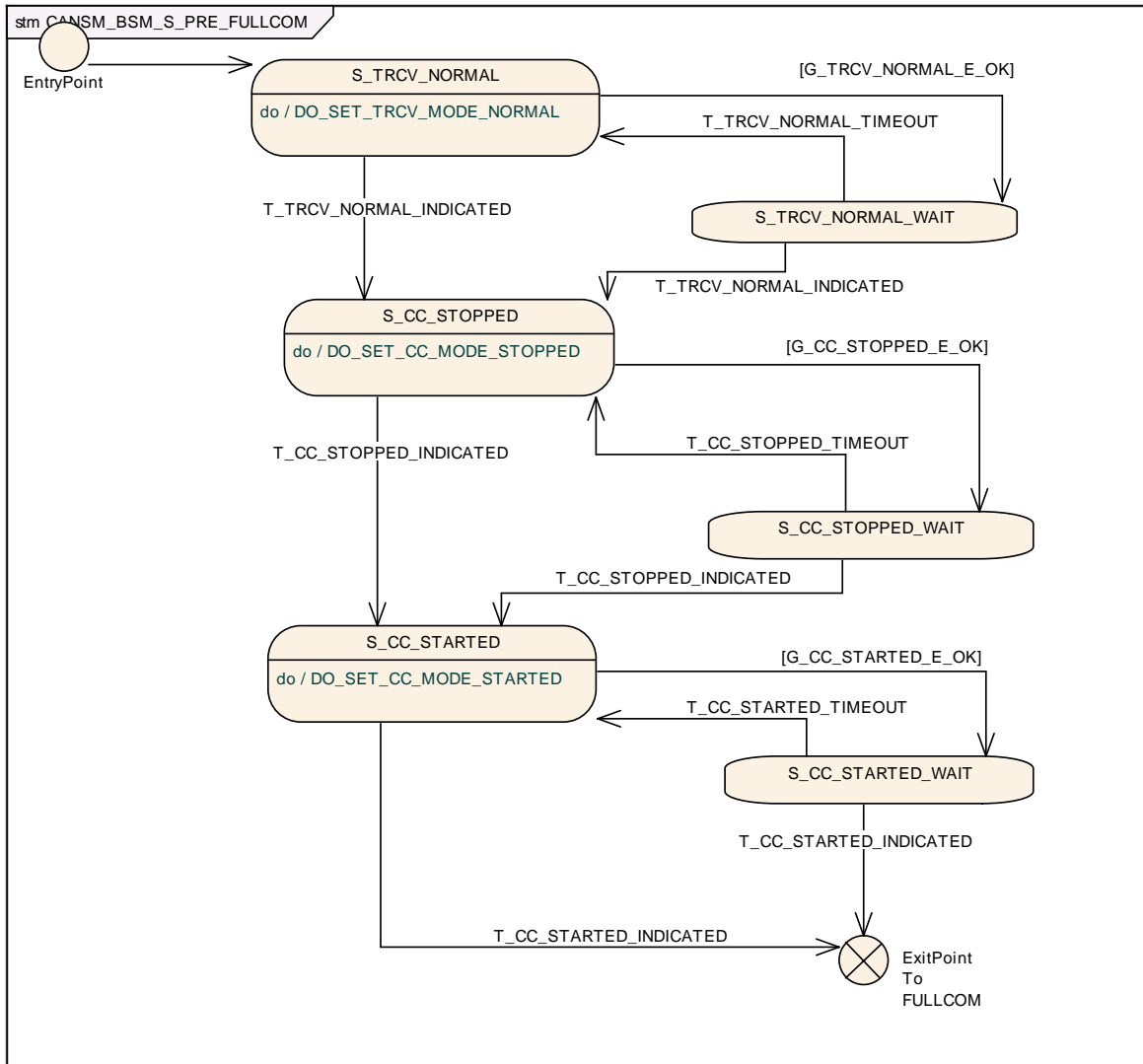


Figure 7-7) of the CAN network with  
T\_CC\_STOPPED\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.8 Trigger: T\_CC\_STOPPED\_TIMEOUT**

**[SWS\_CanSM\_00490]** [ After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller stopped mode indications (ref. to [SWS\\_CanSM\\_00489](#)), this condition shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

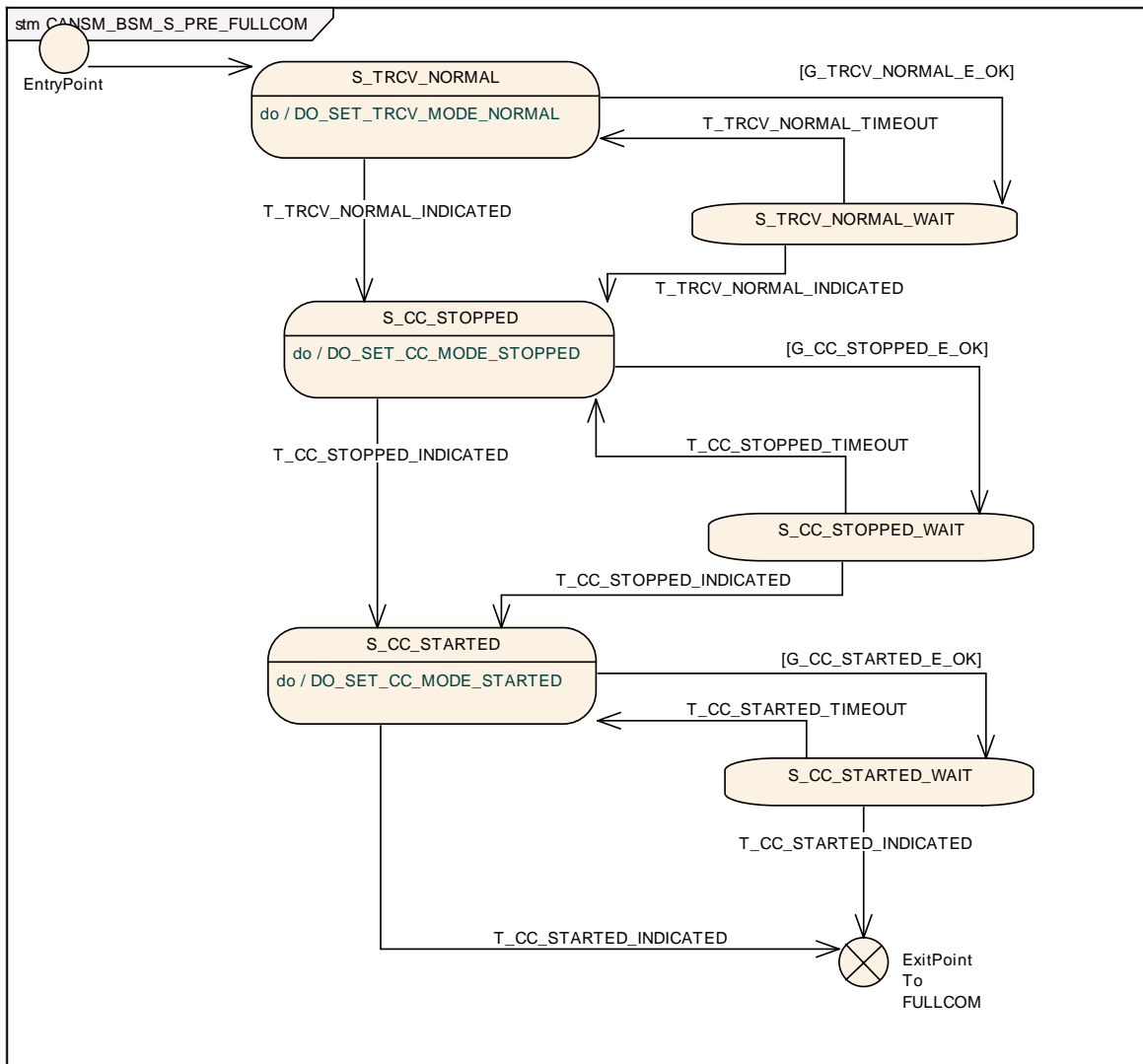


Figure 7-7) of the respective network with  
T\_CC\_STOPPED\_TIMEOUT.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.9 State operation to do in: S\_CC\_STARTED**

[SWS\_CanSM\_00491] | As long the sub state machine  
CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

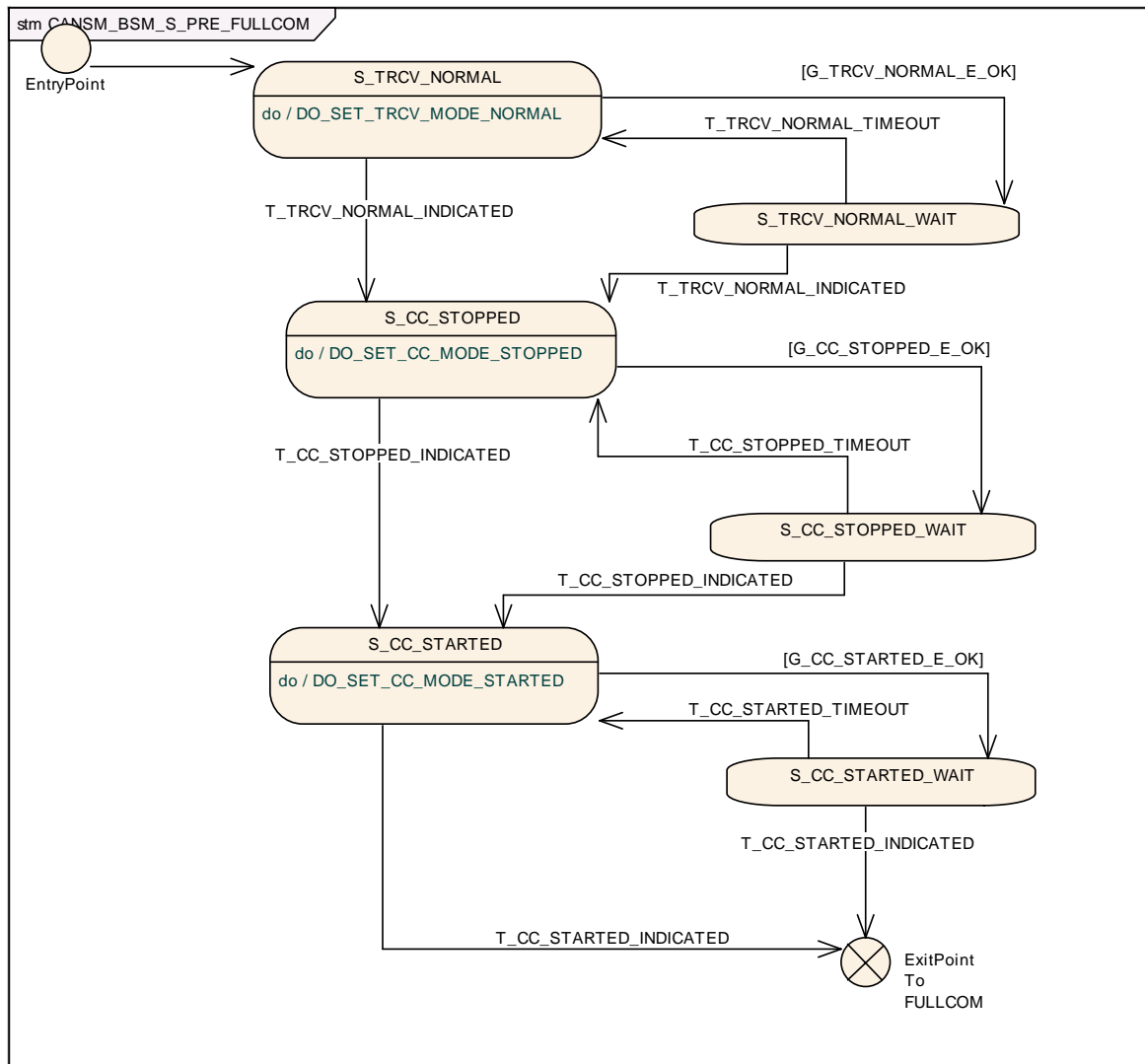


Figure 7-7) is in the state S\_CC\_STARTED, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STARTED and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC CanSM\\_00141](#)) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STARTED, if the current CAN controller mode (ref. to [SWS CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.10 Guarding condition: G\_CC\_STARTED\_OK**

**[SWS\_CanSM\_00492]** | The guarding condition G\_CC\_STARTED\_OK of the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to



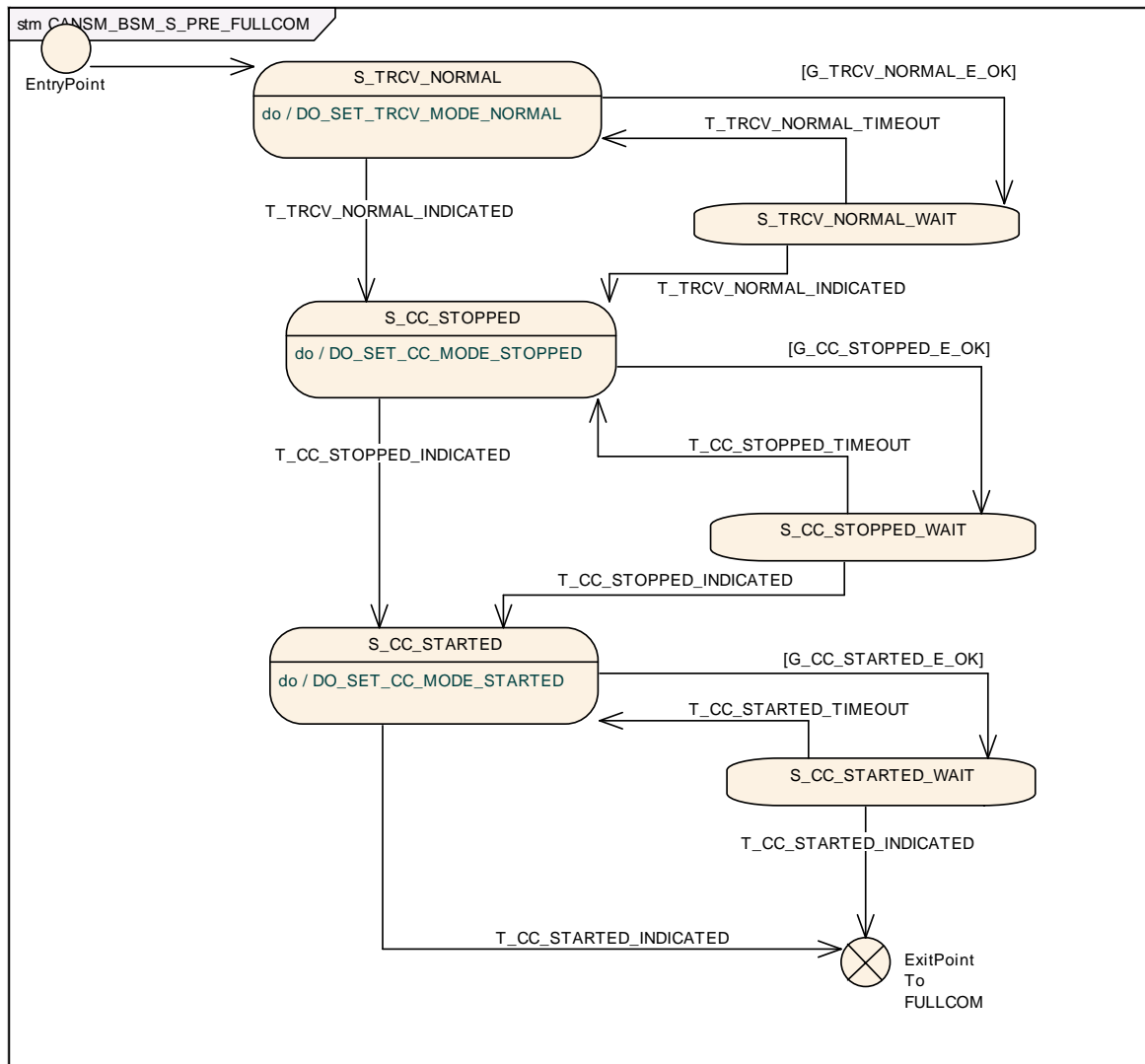


Figure 7-7) shall be passed, if all API calls of [SWS CanSM\\_00491](#) have returned E\_OK.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.11 Trigger: T\_CC\_STARTED\_INDICATED**

**[SWS\_CanSM\_00493]** [ If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to start the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00491](#)), this shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

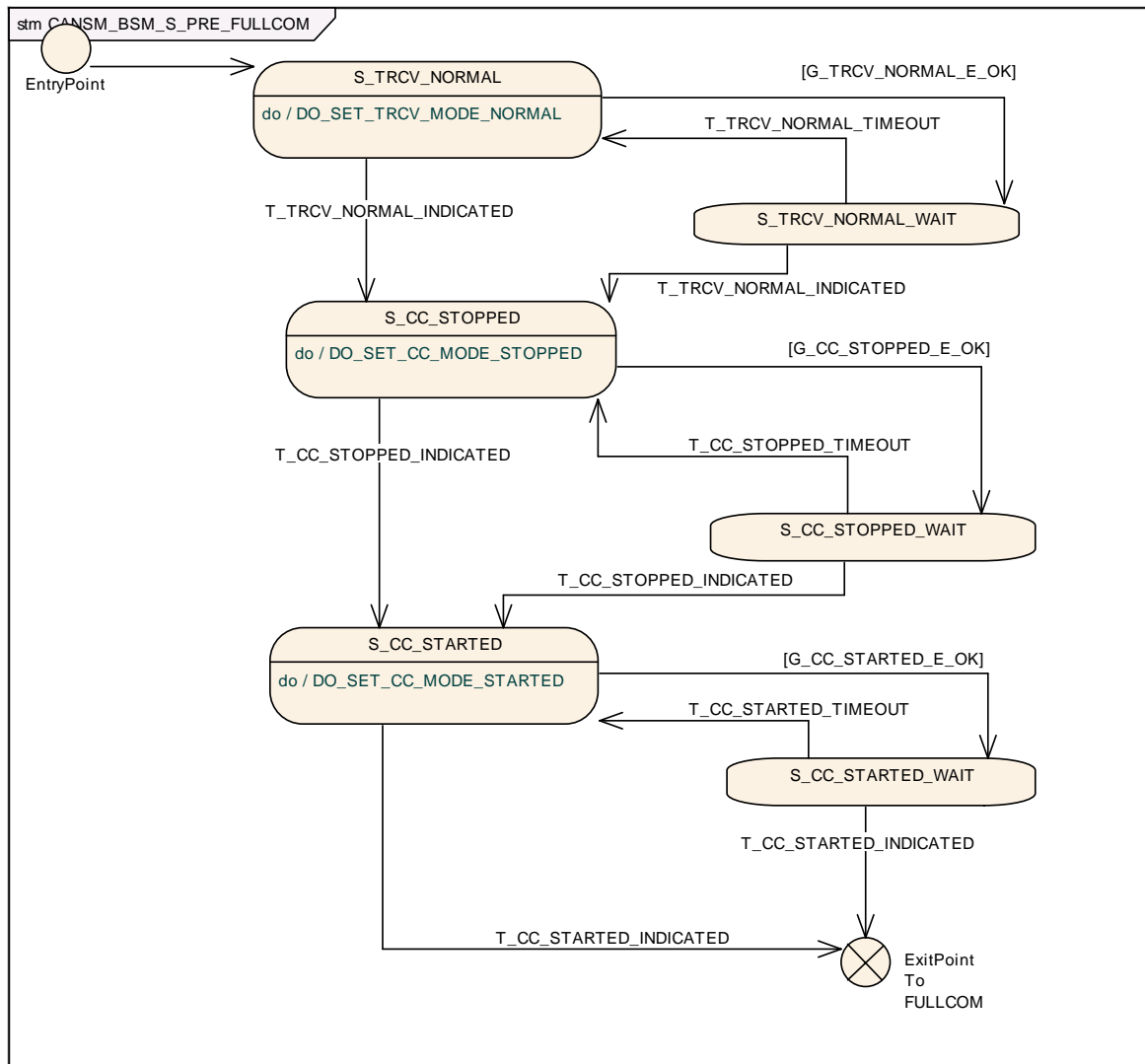


Figure 7-7) of the CAN network with  
T\_CC\_STARTED\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.22.12 Trigger: T\_CC\_STARTED\_TIMEOUT**

**[SWS\_CanSM\_00494]** [ After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller started mode indications (ref. to [SWS\\_CanSM\\_00493](#)), this condition shall trigger the sub state machine CANSM\_BSM\_S\_PRE\_FULLCOM (ref. to

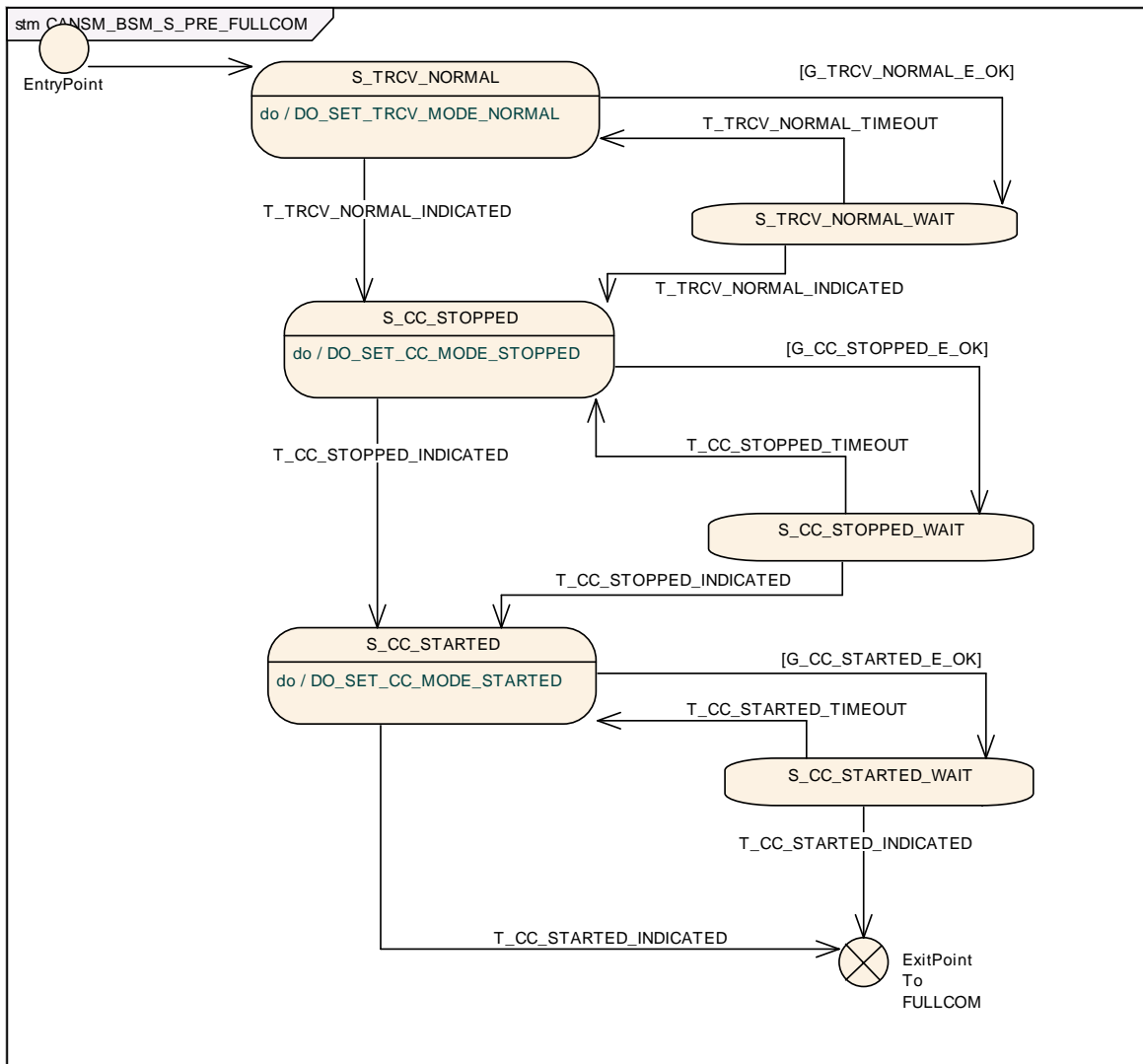
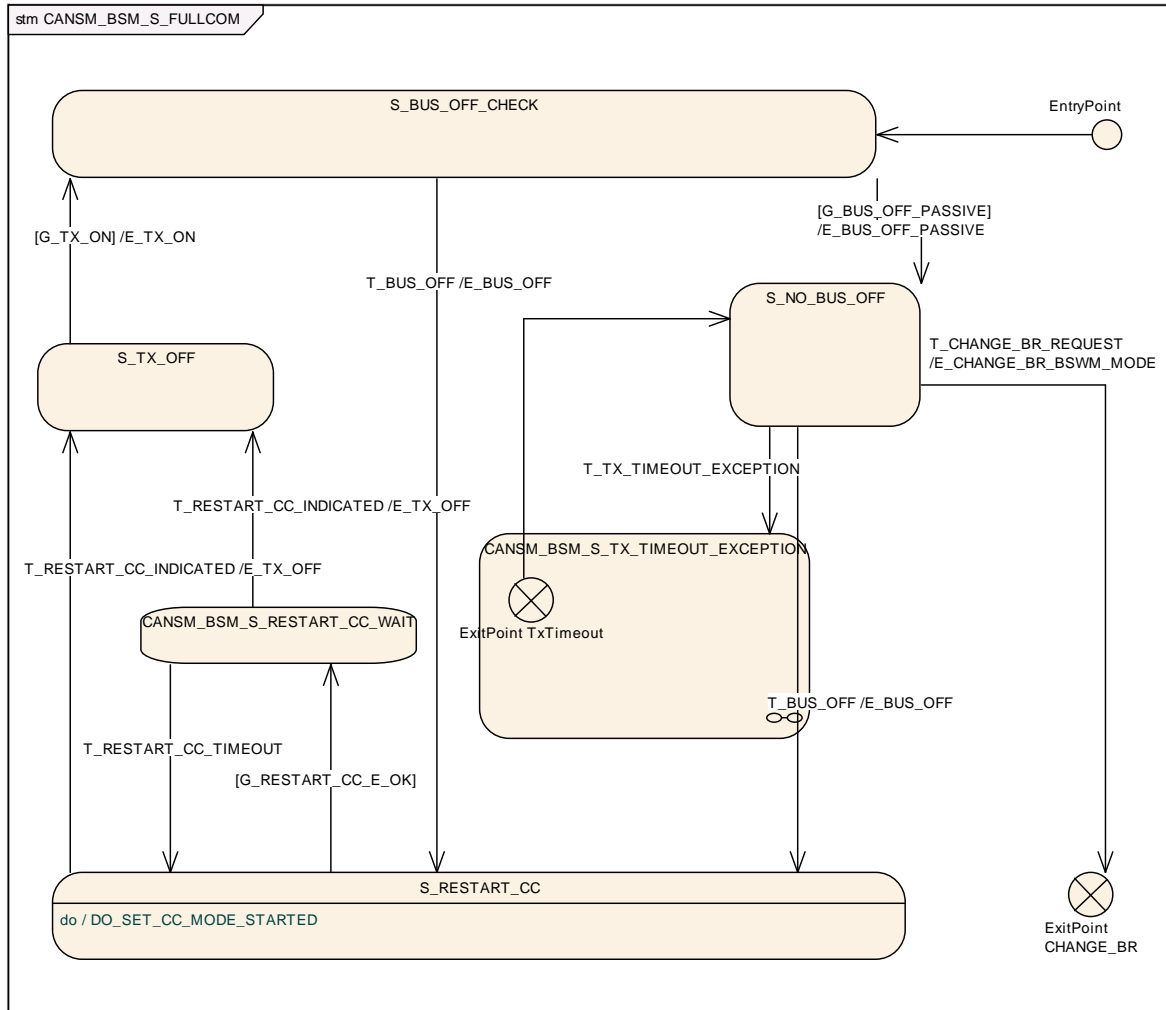


Figure 7-7) of the respective network with `T_CC_STARTED_TIMEOUT`.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23 Sub state machine CANSM\_BSM\_S\_FULLCOM**



**Figure 7-8: CANSM\_BSM\_S\_FULLCOM, sub state machine of CANSM\_BSM**

**7.2.23.1 Guarding condition: G\_BUS\_OFF\_PASSIVE**

**[SWS\_CanSM\_00496]** [ The guarding condition `G_BUS_OFF_PASSIVE` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall be passed, if `CANSM_BOR_TX_CONFIRMATION_POLLING` is disabled (ref. to [ECUC\\_CanSM\\_00339](#)) and the time duration since the effect `E_TX_ON` is greater or equal the configuration parameter `CANSM_BOR_TIME_TX_ENSURED` (ref. to [ECUC\\_CanSM\\_00130](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00497]** [ The guarding condition `G_BUS_OFF_PASSIVE` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall be passed, if `CANSM_BOR_TX_CONFIRMATION_POLLING` is enabled (ref. to [ECUC\\_CanSM\\_00339](#)) and the API `CanIf_GetTxConfirmationState` (ref. to chapter 8.5.1) returns `CANIF_TX_RX_NOTIFICATION` for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.2 Effect: E\_BUS\_OFF\_PASSIVE**

**[SWS\_CanSM\_00498]** [ The effect `E_BUS_OFF_PASSIVE` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall invoke `Dem_SetEventStatus` (ref. to chapter 8.5.1) with the parameters `EventId := CANSM_E_BUS_OFF` (ref. to [ECUC\\_CanSM\\_00070](#)) and `EventStatus := DEM_EVENT_STATUS_PASSED.`] (SRS\_BSW\_00422)

**7.2.23.3 Trigger: T\_CHANGE\_BR\_REQUEST**

**[SWS\_CanSM\_00507]** [ If no condition is present to deny the `CanSM_SetBaudrate` request (ref. to [SWS\\_CANSM\\_00503](#)), this shall trigger the state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) and respectively the parent state machine `CANSM_BSM` (ref. to

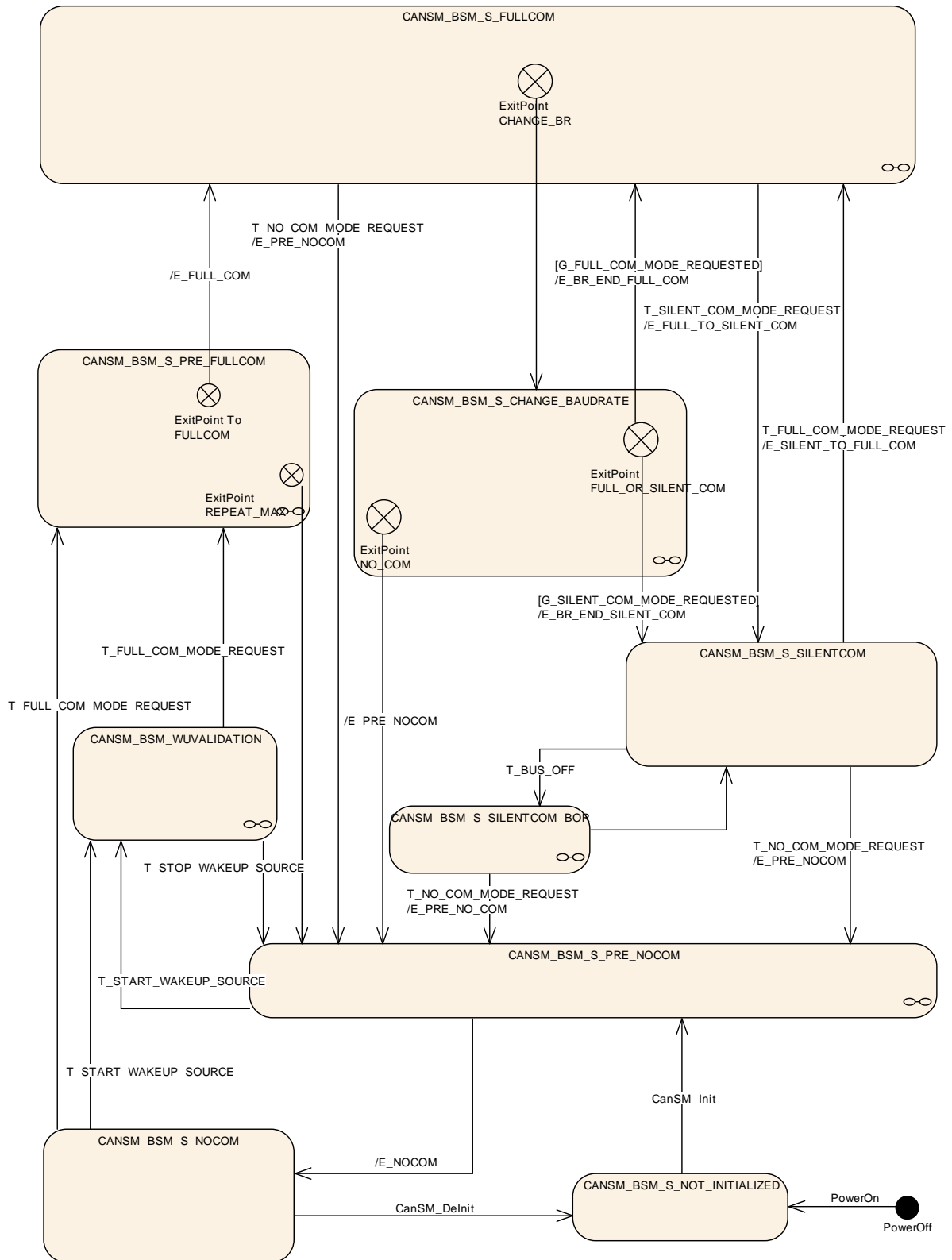


Figure 7-1) with **T\_CHANGE\_BR\_REQUEST** (causes either a direct baud rate change if possible via **CanIf\_SetBaudrate** or the start of the required asynchronous process to do that) (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.4 Effect: E\_CHANGE\_BR\_BSWM\_MODE

**[SWS\_CanSM\_00528]** | The effect `E_CHANGE_BR_BSWM_MODE` of the sub state machine `CANSM_BSM_S_FULLLCOM` (ref. to Figure 7-8) shall call for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANSM_BSWM_CHANGE_BAUDRATE.` (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.5 Trigger: T\_BUS\_OFF

**[SWS\_CanSM\_00500]** | The callback function `CanSM_ControllerBusOff` (ref. to [SWS\\_CanSM\\_00064](#)) shall trigger the sub state machine `CANSM_BSM_S_FULLLCOM` (ref. to Figure 7-8) for the CAN network with `T_BUS_OFF`, if one of its configured CAN controllers matches to the function parameter `ControllerId` of the callback function `CanSM_ControllerBusOff.` (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00653]** | If more than one CAN controller belongs to one CAN network and for one of its controllers a bus-off is indicated with `CanSM_ControllerBusOff`, then the `CanSM` shall stop in context of the effect `E_BUS_OFF` the other CAN controller(s) of the CAN network, too.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.6 Effect: E\_BUS\_OFF

**[SWS\_CanSM\_00508]** | The effect `E_BUS_OFF` of the sub state machine `CANSM_BSM_S_FULLLCOM` (ref. to Figure 7-8) shall call at 1<sup>st</sup> place for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANSM_BSWM_BUS_OFF.` (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00521]** | The effect `E_BUS_OFF` of the sub state machine `CANSM_BSM_S_FULLLCOM` (ref. to Figure 7-8) shall call at 2<sup>nd</sup> place for the corresponding CAN network the API `ComM_BusSM_ModeIndication` with the parameters `Channel := CanSMComMNetworkHandleRef` (ref. to [ECUC\\_CanSM\\_00161](#)) and `ComMode := COMM_SILENT_COMMUNICATION.` (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00522]** | The effect `E_BUS_OFF` of the sub state machine `CANSM_BSM_S_FULLLCOM` (ref. to Figure 7-8) shall invoke `Dem_SetEventStatus` (ref. to chapter 8.5.1) with the parameters `EventId := CANSM_E_BUS_OFF` (ref. to [ECUC\\_CanSM\\_00070](#)) and `EventStatus := DEM_EVENT_STATUS_PRE_FAILED.` (SRS\_BSW\_00422)

#### 7.2.23.7 State operation to do in: S\_RESTART\_CC

**[SWS\_CanSM\_00509]** | As long the sub state machine `CANSM_BSM_S_FULLLCOM` (ref. to Figure 7-8) is in the state `S_RESTART_CC`, the `CanSM` module shall operate

the do action `DO_SET_CC_MODE_STARTED` and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC CanSM\\_00141](#)) the API request `CanIf_SetControllerMode` (ref. to chapter 8.5.1) with `ControllerMode` equal to `CAN_CS_STARTED`, if the current CAN controller mode (ref. to [SWS CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.8 Guarding condition: `G_RESTART_CC_OK`

[[SWS CanSM\\_00510](#)] [ The guarding condition `G_RESTART_CC_OK` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall be passed, if all API calls of [SWS CanSM\\_00509](#) have returned `E_OK`.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.9 Trigger: `T_RESTART_CC_INDICATED`

[[SWS CanSM\\_00511](#)] [ If `CanSM` module has got all mode indications (ref. to [SWS CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC CanSM\\_00141](#)) after the respective requests to start the CAN controllers of the CAN network (ref. to [SWS CanSM\\_00509](#)), this shall trigger the sub state `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) of the CAN network with `T_RESTART_CC_INDICATED`.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.10 Trigger: `T_RESTART_CC_TIMEOUT`

[[SWS CanSM\\_00512](#)] [ After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC CanSM\\_00336](#)) for all supposed controller started mode indications (ref. to [SWS CanSM\\_00511](#)), this condition shall trigger the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) of the respective network with `T_RESTART_CC_TIMEOUT`.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.11 Effect: `E_TX_OFF`

The effect `E_TX_OFF` shall do nothing.

#### 7.2.23.12 Guarding condition: `G_TX_ON`

[[SWS CanSM\\_00514](#)] [ If `CanSMEnableBusOffDelay` is `FALSE`, then guarding condition `G_TX_ON` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall be passed after a time duration of `CanSMBorTimeL1` (ref. to [ECUC CanSM\\_00128](#)) related to the last `T_BUS_OFF`, if the count of bus-off recovery retries with `E_BUS_OFF` without passing the guarding condition `G_BUS_OFF_PASSIVE` is lower than `CanSMBorCounterL1ToL2` (ref. to [ECUC CanSM\\_00131](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

[[SWS CanSM\\_00515](#)] [ If `CanSMEnableBusOffDelay` is `FALSE`, then the guarding condition `G_TX_ON` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall be passed after a time duration of `CanSMBorTimeL2` (ref. to



[ECUC CanSM\\_00129](#)) related to the last T\_BUS\_OFF, if the count of bus-off recovery retries with E\_BUS\_OFF without passing the guarding condition G\_BUS\_OFF\_PASSIVE is greater than or equal to CanSMBorCounterL1ToL2 (ref. to [ECUC CanSM\\_00131](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00636]** [ If CanSMEnableBusOffDelay is TRUE, then the guarding conditions of [SWS\\_CANSMS\\_00514](#) and [SWS\\_CANSMS\\_00515](#) shall be passed after the specified time duration in each case plus the additional random delay value, which shall be requested after the bus-off event with the configured call back function <User\_GetBusOffDelay>.] (SRS\_Can\_01145, SRS\_Can\_01142)

### 7.2.23.13 Effect: E\_TX\_ON

**[SWS\_CanSM\_00516]** [ If ECU passive is FALSE (ref. to [SWS\\_CanSM\\_00646](#)), then the effect E\_TX\_ON of the sub state machine CANSM\_BSM\_S\_FULLCOM (ref. to Figure 7-8) shall call at 1<sup>st</sup> place for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API function CanIf\_SetPduMode (ref. to chapter 8.5.1) with the parameters ControllerId := CanSMControllerId (ref. to [ECUC\\_CanSM\\_00141](#)) and PduModeRequest := CANIF\_ONLINE.] (SRS\_Can\_01158)

**[SWS\_CanSM\_00648]** If ECU passive is TRUE (ref. to [SWS\\_CanSM\\_00646](#)), then the effect E\_TX\_ON of the sub state machine CANSM\_BSM\_S\_FULLCOM (ref. to Figure 7-8) shall call at 1<sup>st</sup> place for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API function CanIf\_SetPduMode (ref. to chapter 8.5.1) with the parameters ControllerId := CanSMControllerId (ref. to [ECUC\\_CanSM\\_00141](#)) and PduModeRequest := CANIF\_TX\_OFFLINE\_ACTIVE.] (SRS\_Can\_01158)

**[SWS\_CanSM\_00517]** [ The effect E\_TX\_ON of the sub state machine CANSM\_BSM\_S\_FULLCOM (ref. to Figure 7-8) shall call at 2<sup>nd</sup> place for the corresponding CAN network the API BswM\_CanSM\_CurrentState with the parameters Network := CanSMComMNetworkHandleRef and CurrentState := CANSM\_BSM\_FULL\_COMMUNICATION.] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00518]** [ The effect E\_TX\_ON of the sub state machine CANSM\_BSM\_S\_FULLCOM (ref. to Figure 7-8) shall call at 3<sup>rd</sup> place the API ComM\_BusSM\_ModeIndication with the parameters Channel := CanSMComMNetworkHandleRef (ref. to [ECUC\\_CanSM\\_00161](#)) and ComMode := COMM\_FULL\_COMMUNICATION.] (SRS\_Can\_01145, SRS\_Can\_01142)

### 7.2.23.14 Trigger: T\_TX\_TIMEOUT\_EXCEPTION

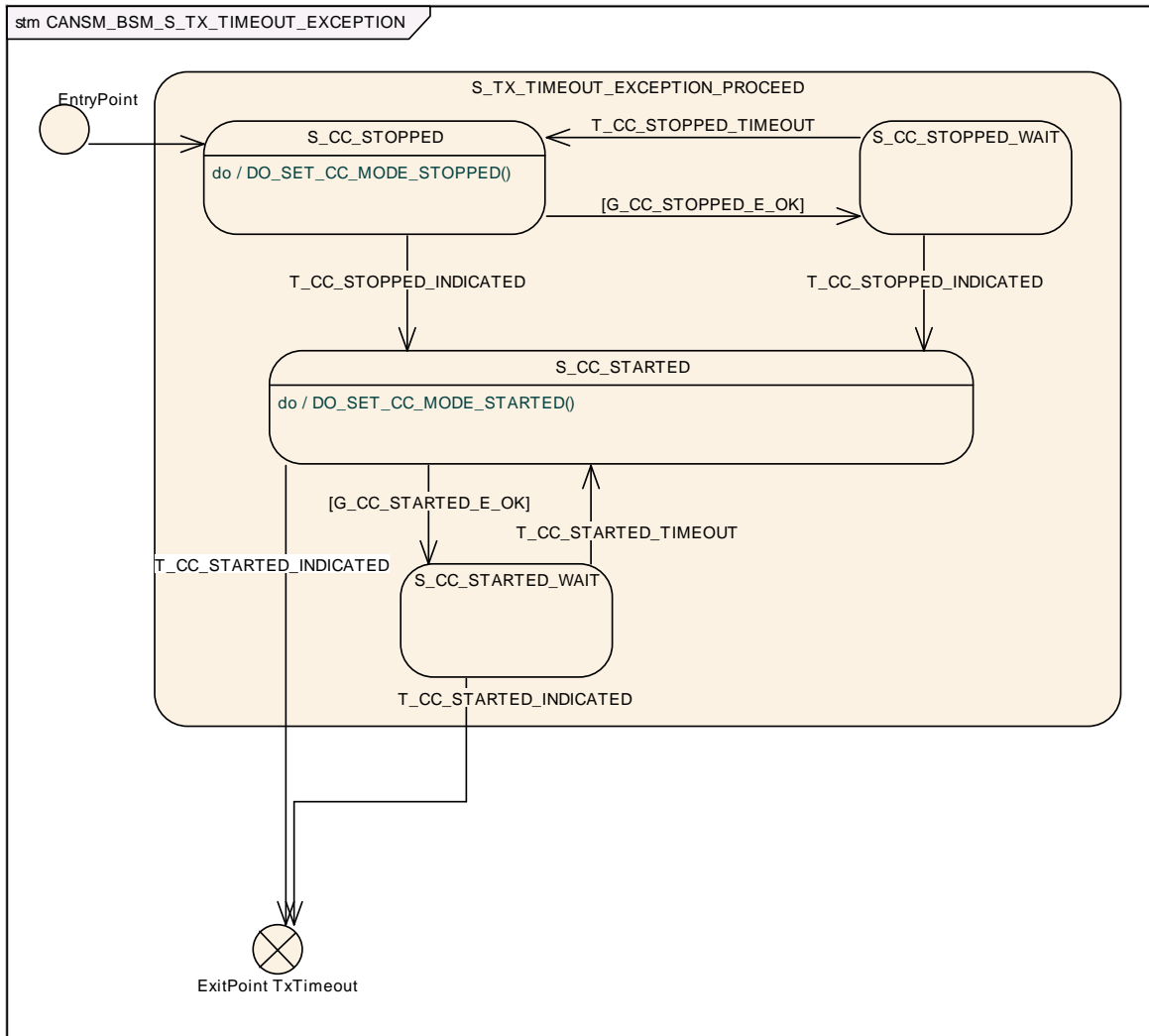
**[SWS\_CanSM\_00584]** [ The callback function CanSM\_TxTimeoutException (ref. to [SWS\\_CANSMS\\_00410](#)) shall trigger the sub state machine

CANSM\_BSM\_S\_FULLCOM (ref. to Figure 7-8) with T\_TX\_TIMEOUT\_EXCEPTION.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.15 Notes**

In the state S\_NO\_BUS\_OFF no state operation is required for the CanSM module.

**7.2.23.16 Sub state machine: CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION**



**Figure 7-9: CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION, sub state machine of CANSM\_BSM\_S\_FULLCOM**

**7.2.23.16.1 Trigger: T\_CC\_STOPPED\_TIMEOUT**

[SWS\_CanSM\_00576] After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to ECUC\_CanSM\_00336) for all supposed controller stopped mode indications (ref. to SWS\_CanSM\_00579), this condition shall trigger the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) of the respective network with T\_CC\_STOPPED\_TIMEOUT.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.16.2 Guarding condition: G\_CC\_STOPPED\_E\_OK**

[SWS\_CanSM\_00577] | The guarding condition G\_CC\_STOPPED\_E\_OK of the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) shall be passed, if all API calls of [SWS\\_CanSM\\_00578](#) have returned E\_OK.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.16.3 State operation: DO\_SET\_CC\_MODE\_STOPPED()**

[SWS\_CanSM\_00578] | As long the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) is in the state S\_CC\_STOPPED, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STOPPED and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STOPPED, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.16.4 Trigger: T\_CC\_STOPPED\_INDICATED**

[SWS\_CanSM\_00579] | If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to stop the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00524](#)), this shall trigger the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) of the CAN network with T\_CC\_STOPPED\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.16.5 Trigger: T\_CC\_STARTED\_INDICATED**

[SWS\_CanSM\_00580] | If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to start the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00582](#)), this shall trigger the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) of the CAN network with T\_CC\_STARTED\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.16.6 Guarding condition: G\_CC\_STARTED\_E\_OK**

[SWS\_CanSM\_00581] | The guarding condition G\_CC\_STARTED\_E\_OK of the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) shall be passed, if all API calls of [SWS\\_CanSM\\_00582](#) have returned E\_OK.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.23.16.7 State operation: DO\_SET\_CC\_MODE\_STARTED**

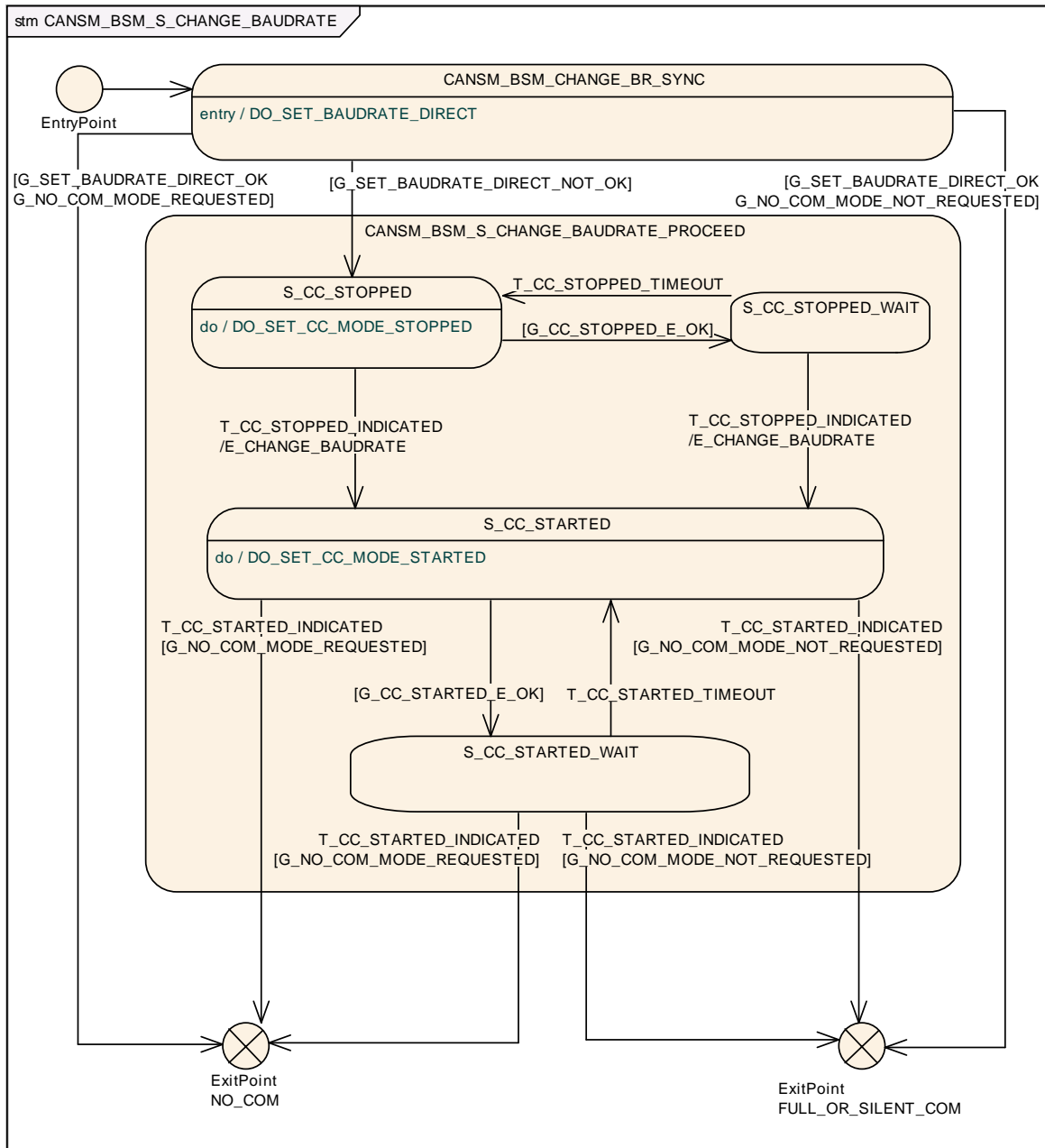
[SWS\_CanSM\_00582] | As long the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) is in the state S\_CC\_STARTED, the CanSM module shall operate the do action

DO\_SET\_CC\_MODE\_STARTED and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STARTED, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.23.16.8 ExitPoint: TxTimeout

[SWS\_CanSM\_00655] If the sub state machine CANSM\_BSM\_S\_TX\_TIMEOUT\_EXCEPTION (ref. to Figure 7-9) is triggered with T\_CC\_STARTED\_INDICATED, the API CanIf\_SetPduMode() shall be called with CANIF\_ONLINE] ()

**7.2.24 Sub state machine: CANSM\_BSM\_S\_CHANGE\_BAUDRATE**



**Figure 7-10: CANSM\_BSM\_S\_CHANGE\_BAUDRATE, sub state machine of CANSM\_BSM**

**7.2.24.1 State operation to do on entry: DO\_SET\_BAUDRATE\_DIRECT**

[SWS\_CanSM\_00639] The state operation DO\_SET\_BAUDRATE\_DIRECT (ref. to Figure 7-10) shall call the API request CanIf\_SetBaudrate (ref. to chapter 8.5.2) for all configured CAN controllers of the CAN network (ref. to ECUC\_CanSM\_00141 with the respective ControllerId parameter. It shall use as BaudRateConfigID parameter the respective function parameter BaudRateConfigID from the call CanSM\_SetBaudrate ().] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.24.2 Guarding condition: G\_SET\_BAUDRATE\_DIRECT\_OK**

[SWS\_CanSM\_00641] If all `CanIf_SetBaudrate` (ref. to [SWS\\_CanSM\\_00639](#)) requests returned with `E_OK`, the guarding condition `G_SET_BAUDRATE_DIRECT_OK` shall be passed.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.24.3 Guarding conditions: G\_SET\_BAUDRATE\_DIRECT\_NOT\_OK**

[SWS\_CanSM\_00642] If any of the `CanIf_SetBaudrate` (ref. to [SWS\\_CanSM\\_00639](#)) requests did return with `E_NOT_OK`, the guarding condition `G_SET_BAUDRATE_NOT_OK` of the state `CANSM_BSM_CHANGE_BR_SYNC` (ref. to Figure 7-10) shall be passed.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.24.4 State operation to do in: S\_CC\_STOPPED**

[SWS\_CanSM\_00524] As long the sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) is in the state `S_CC_STOPPED`, the `CanSM` module shall operate the do action `DO_SET_CC_MODE_STOPPED` and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request `CanIf_SetControllerMode` (ref. to chapter 8.5.1) with `ControllerMode` equal to `CAN_CS_STOPPED`, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.24.5 Guarding condition: G\_CC\_STOPPED\_OK**

[SWS\_CanSM\_00525] The guarding condition `G_CC_STOPPED_OK` of the sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) shall be passed, if all API calls of [SWS\\_CanSM\\_00524](#) have returned `E_OK`.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.24.6 Trigger: T\_CC\_STOPPED\_INDICATED**

[SWS\_CanSM\_00526] If `CanSM` module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to stop the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00524](#)), this shall trigger the sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) of the CAN network with `T_CC_STOPPED_INDICATED`.] (SRS\_Can\_01145, SRS\_Can\_01142)

**7.2.24.7 Trigger: T\_CC\_STOPPED\_TIMEOUT**

[SWS\_CanSM\_00527] After a timeout of `CANSM_MODEREQ_REPEAT_TIME` (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller stopped mode indications (ref. to [SWS\\_CanSM\\_00526](#)), this condition shall trigger the sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) of the respective network with `T_CC_STOPPED_TIMEOUT`.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.24.8 Effect: E\_CHANGE\_BAUDRATE

**[SWS\_CanSM\_00529]** [ The effect E\_CHANGE\_BAUDRATE of the sub state machine CANSM\_BSM\_S\_CHANGE\_BAUDRATE (ref. to Figure 7-10) shall call at 1<sup>st</sup> place for the corresponding CAN network the API ComM\_BusSM\_ModeIndication with the parameters Channel := CanSMComMNetworkHandleRef (ref. to [ECUC\\_CanSM\\_00161](#)) and ComMode := COMM\_NO\_COMMUNICATION.] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00531]** [ The effect E\_CHANGE\_BAUDRATE of the sub state machine CANSM\_BSM\_S\_CHANGE\_BAUDRATE (ref. to Figure 7-10) shall call at 2<sup>nd</sup> place for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request CanIf\_SetBaudrate (ref. to chapter 8.5.2) with the respective ControllerId parameter and shall use as BaudRateConfigID parameter the remembered BaudRateConfigID from the call CanSM\_SetBaudrate ()] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.24.9 State operation to do in: S\_CC\_STARTED

**[SWS\_CanSM\_00532]** [ As long the sub state machine CANSM\_BSM\_S\_CHANGE\_BAUDRATE (ref. to Figure 7-10) is in the state S\_CC\_STARTED, the CanSM module shall operate the do action DO\_SET\_CC\_MODE\_STARTED and therefore repeat for all configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) the API request CanIf\_SetControllerMode (ref. to chapter 8.5.1) with ControllerMode equal to CAN\_CS\_STARTED, if the current CAN controller mode (ref. to [SWS\\_CanSM\\_00638](#)) is different.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.24.10 Guarding condition: G\_CC\_STARTED\_OK

**[SWS\_CanSM\_00533]** [ The guarding condition G\_CC\_STARTED\_OK of the sub state machine CANSM\_BSM\_S\_CHANGE\_BAUDRATE (ref. to Figure 7-10) shall be passed, if all API calls of [SWS\\_CanSM\\_00532](#) have returned E\_OK.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.24.11 Trigger: T\_CC\_STARTED\_INDICATED

**[SWS\_CanSM\_00534]** [ If CanSM module has got all mode indications (ref. to [SWS\\_CanSM\\_00396](#)) for the configured CAN controllers of the CAN network (ref. to [ECUC\\_CanSM\\_00141](#)) after the respective requests to start the CAN controllers of the CAN network (ref. to [SWS\\_CanSM\\_00532](#)), this shall trigger the sub state machine CANSM\_BSM\_S\_CHANGE\_BAUDRATE (ref. to Figure 7-10) of the CAN network with T\_CC\_STARTED\_INDICATED.] (SRS\_Can\_01145, SRS\_Can\_01142)

#### 7.2.24.12 Trigger: T\_CC\_STARTED\_TIMEOUT

**[SWS\_CanSM\_00535]** [ After a timeout of CANSM\_MODEREQ\_REPEAT\_TIME (ref. to [ECUC\\_CanSM\\_00336](#)) for all supposed controller started mode indications (ref.

to [SWS\\_CanSM\\_00534](#)), this condition shall trigger the sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) of the respective network with `T_CC_STARTED_TIMEOUT.`] (SRS\_Can\_01145, SRS\_Can\_01142)

### 7.2.24.13 Guarding condition: `G_NO_COM_MODE_REQUESTED`

**[SWS\_CanSM\_00542]** [ The sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) shall pass the guarding condition `G_NO_COM_MODE_REQUESTED`, if the latest accepted communication mode request with `CanSM_RequestComMode` (ref. to [SWS\\_CanSM\\_00635](#)) for the respective network handle of the state machine has been with the parameter `ComM_Mode` equal to `COMM_NO_COMMUNICATION.`] (SRS\_Can\_01145, SRS\_Can\_01142)

### 7.2.24.14 Guarding condition: `G_NO_COM_MODE_NOT_REQUESTED`

**[SWS\_CanSM\_00543]** [ The sub state machine `CANSM_BSM_S_CHANGE_BAUDRATE` (ref. to Figure 7-10) shall pass the guarding condition `G_NO_COM_MODE_NOT_REQUESTED`, if the latest accepted communication mode request with `CanSM_RequestComMode` (ref. to [SWS\\_CanSM\\_00635](#)) for the respective network handle of the state machine has been with the parameter `ComM_Mode` equal to `COMM_SILENT_COMMUNICATION` or `COMM_FULL_COMMUNICATION.`] (SRS\_Can\_01145, SRS\_Can\_01142)

## 7.3 Error classification

Section 7.x "Error Handling" of the document "General Specification of Basic Software Modules" describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types, which may occur in BSW modules.

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

### 7.3.1 Development Errors

This chapter shall list all Development Errors that can be detected within this software module. For each error, a value shall be defined.

**[SWS\_CanSM\_00654]**

<i>Type or error</i>	<i>Relevance</i>	<i>Related error code</i>	<i>Value [hex]</i>
API service used without module initialization	Development	<code>CANSM_E_UNINIT</code>	0x01
API service called with wrong pointer	Development	<code>CANSM_E_PARAM_POINTER</code>	0x02
API service called with wrong parameter	Development	<code>CANSM_E_INVALID_NETWORK_HANDLE</code>	0x03
API service called with wrong parameter	Development	<code>CANSM_E_PARAM_CONTROLLER</code>	0x04



API service called with wrong parameter	Development	CANSM_E_PARAM_TRANSCEIVER	0x05
Delnit API service called when not all CAN networks are in state CANSM_NO_COMMUNICATION	Development	CANSM_E_NOT_IN_NO_COM	0x0B

└ (SRS\_BSW\_00337)

## 7.3.2 Runtime Errors

### 7.3.2.1 CANSM\_E\_MODE\_REQUEST\_TIMEOUT

[SWS\_CanSM\_00664]

Type or error	Related error code	Value [hex]
Mode request for a network failed more often than allowed by configuration	CANSM_E_MODE_REQUEST_TIMEOUT	0x0A

└ (SRS\_BSW\_00466)

## 7.3.3 Transient Faults

There are no transient faults

## 7.3.4 Production Errors

## 7.3.5 Extended Production Errors

### 7.3.5.1 CANSM\_E\_BUS\_OFF

[SWS\_CanSM\_00666]

<b>Error Name:</b>	CANSM_E_BUS_OFF (ref. to <a href="#">ECUC_CanSM_00070</a> )	
<b>Short Description:</b>	Bus-off detection	
<b>Long Description:</b>	The bus-off recovery state machine of a CAN network has detected a certain amount of sequential bus-offs without successful recovery	
<b>Recommended DTC:</b>	Assigned by DEM	
<b>Detection Criteria:</b>	Fail	PRE_FAILED when CanSM_ControllerBusOff is called (T_BUS_OFF/E_BUS_OFF), debouncing to be defined by OEM in DEM
	Pass	After successful transmission of a CAN frame (G_BUS_OFF_PASSIVE/E_BUS_OFF_PASSIVE)
<b>Secondary Parameters:</b>	None	
<b>Time Required:</b>	PRE_FAILED immediately (in error interrupt context), FAILED depending on debounce configuration of DEM	
<b>Monitor Frequency</b>	Continuous	
<b>MIL illumination:</b>	Assigned by DEM	

└ ()

## 7.4 Pretended Networking function

### 7.4.1 Activation

[SWS\_CanSM\_00588] To activate Pretended Networking the CanSM module shall request an ICOM configuration by calling `CanIf_SetIcomConfiguration.` (SRS\_Can\_01145, SRS\_Can\_01142)

[SWS\_CanSM\_00589] The CanSM shall inform the BswM about the activation status by calling `BswM_CanSM_CurrentIcomConfiguration.` (SRS\_Can\_01145, SRS\_Can\_01142)

### 7.4.2 Deactivation

[SWS\_CanSM\_00590] The CanSM shall call the provided API `CanIf_SetIcomConfiguration` to deactivate the Pretended Networking and to set back the ICOM configuration to 0. (SRS\_Can\_01145, SRS\_Can\_01142)

[SWS\_CanSM\_00591] The CanSM shall inform BswM about the deactivation status by calling `BswM_CanSM_CurrentIcomConfiguration.` (SRS\_Can\_01145, SRS\_Can\_01142)

## 7.5 ECU online active / passive mode

[SWS\_CanSM\_00646] The CanSM state manager shall store the state of the requested ECU passive mode (ref. to chapter 8: [SWS\\_CanSM\\_00644](#)). (SRS\_Can\_01158)

[SWS\_CanSM\_00649] If `CanSM_SetEcuPassive` called with `CanSM_Passive=true`; (ref. to chapter 8: [SWS\\_CanSM\\_00644](#)), then the CanSM shall change all PDU modes of the configured CAN controllers, which are `CANIF_ONLINE` at the moment to `CANIF_TX_OFFLINE_ACTIVE` by calling the API `CanIf_SetPduMode` (ref. to chapter 8.5.1) with the parameters `ControllerId := CanSMControllerId` (ref. to [ECUC\\_CanSM\\_00141](#)) and `PduModeRequest := CANIF_TX_OFFLINE_ACTIVE.` (SRS\_Can\_01158)

[SWS\_CanSM\_00650] If `CanSM_SetEcuPassive` called with `CanSM_Passive=false`; (ref. to chapter 8: [SWS\\_CanSM\\_00644](#)), then the CanSM shall change all PDU modes of the configured CAN controllers, which are `CANIF_TX_OFFLINE_ACTIVE` at the moment to `CANIF_ONLINE` by calling the API `CanIf_SetPduMode` (ref. to chapter 8.5.1) with the parameters `ControllerId := CanSMControllerId` (ref. to [ECUC\\_CanSM\\_00141](#)) and `PduModeRequest := CANIF_ONLINE.` (SRS\_Can\_01158)

[SWS\_CanSM\_00656] If the CanSM needs informations about the actual PduMode, the CanSM shall call the API `CanIf_GetPduMode` to get the current Pdu Mode of the `CanIf.` (SRS\_Can\_01158)

## 7.6 Error detection

[For details refer to *SWS\_BSWGeneral*] ( SRS\_BSW\_00337 )

## 7.7 Error notification

[For details refer to the *SWS\_BSWGeneral*] ( SRS\_BSW\_00369, SRS\_BSW\_00385, SRS\_BSW\_00327 )

## 7.8 Non-functional design rules

The CanSM shall cover the software module design requirements of the SRS General [3].

## 8 API specification

### 8.1 Imported types

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

[SWS\_CanSM\_00243]

<i>Module</i>	<i>Header File</i>	<i>Imported Type</i>
Can_GeneralTypes	Can_GeneralTypes.h	CanTrcv_TrcvModeType
	Can_GeneralTypes.h	Can_ControllerStateType
CanIf	CanIf.h	CanIf_NotifStatusType
	CanIf.h	CanIf_PduModeType
ComM	Rte_ComM_Type.h	ComM_ModeType
ComStack_Types	ComStack_Types.h	IcomConfigIdType
	ComStack_Types.h	IcomSwitch_ErrorType
	ComStack_Types.h	NetworkHandleType
Dem	Rte_Dem_Type.h	Dem_EventIdType
	Rte_Dem_Type.h	Dem_EventStatusType
Std	Std_Types.h	Std_ReturnType
	Std_Types.h	Std_VersionInfoType

l()

## 8.2 Type definitions

The following tables contain the type definitions of the CanSM module.

### 8.2.1 CanSM\_StateType

[SWS\_CanSM\_00596]

<b>Name</b>	CanSM_StateType		
<b>Kind</b>	Enumeration		
<b>Range</b>	CANSM_INITED	--	--
	CANSM_UNINITED	--	--
<b>Description</b>	Defines the values of the internal states of the CanSM module		
<b>Available via</b>	CanSM.h		

|(SRS\_BSW\_00405, SRS\_BSW\_00101, SRS\_BSW\_00406, SRS\_BSW\_00358, SRS\_BSW\_00414, SRS\_BSW\_00404)

### 8.2.2 CanSM\_ConfigType

[SWS\_CanSM\_00597]

<b>Name</b>	CanSM_ConfigType		
<b>Kind</b>	Structure		
<b>Elements</b>	--		
	<b>Type</b>		--
	<b>Comment</b>		--
<b>Description</b>	This type defines a data structure for the post build parameters of the CanSM. At initialization the CanSM gets a pointer to a structure of this type to get access to its configuration data, which is necessary for initialization.		
<b>Available via</b>	CanSM.h		

|(SRS\_BSW\_00400, SRS\_BSW\_00438)

### 8.2.3 CanSM\_BswMCurrentStateType

[SWS\_CanSM\_00598]

<b>Name</b>	CanSM_BswMCurrentStateType		
<b>Kind</b>	Enumeration		

<b>Range</b>	CANSM_BSWM_NO_COMMUNICATION	--	--
	CANSM_BSWM_SILENT_COMMUNICATION	--	--
	CANSM_BSWM_FULL_COMMUNICATION	--	--
	CANSM_BSWM_BUS_OFF	--	--
	CANSM_BSWM_CHANGE_BAUDRATE	--	--
<b>Description</b>	Can specific communication modes / states notified to the BswM module		
<b>Available via</b>	CanSM.h		

](SRS\_ModeMgm\_09251)

### 8.3 Function definitions

The following sections specify the provided API functions of the CanSM module.

#### 8.3.1 CanSM\_Init

[SWS\_CanSM\_00023]

<b>Service Name</b>	CanSM_Init	
<b>Syntax</b>	<pre>void CanSM_Init (     const CanSM_ConfigType* ConfigPtr )</pre>	
<b>Service ID [hex]</b>	0x00	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	ConfigPtr	Pointer to init structure for the post build parameters of the CanSM
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This service initializes the CanSM module	
<b>Available via</b>	CanSM.h	

|(SRS\_BSW\_00405, SRS\_BSW\_00101, SRS\_BSW\_00406, SRS\_BSW\_00358, SRS\_BSW\_00414, SRS\_BSW\_00404, SRS\_BSW\_00400, SRS\_BSW\_00438)

#### 8.3.2 CanSM\_DeInit

[SWS\_CanSM\_91001]

<b>Service Name</b>	CanSM_DeInit	
<b>Syntax</b>	<pre>void CanSM_DeInit (     void )</pre>	
<b>Service ID [hex]</b>	0x14	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	None	
<b>Parameters (inout)</b>	None	

<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	This service de-initializes the CanSM module.
<b>Available via</b>	CanSM.h

](SRS\_Can\_01164, SRS\_BSW\_00336)

Note: General behavior and constraints on de-initialization functions are specified by [SWS\_BSW\_00152], [SWS\_BSW\_00072], [SWS\_BSW\_00232], [SWS\_BSW\_00233].

Caveat: Caller of the CanSM\_Delnit function has to ensure all CAN networks are in the state CANSM\_NO\_COMMUNICATION.

[SWS\_CanSM\_00660] If development error detection for the CanSM module is enabled: The function CanSM\_Delnit shall raise the error CANSM\_E\_NOT\_IN\_NO\_COM if not all CAN networks are in state CANSM\_NO\_COMMUNICATION. ] (SRS\_BSW\_00369)

### 8.3.3 CanSM\_RequestComMode

[SWS\_CanSM\_00062]

<b>Service Name</b>	CanSM_RequestComMode	
<b>Syntax</b>	<pre>Std_ReturnType CanSM_RequestComMode (     NetworkHandleType network,     ComM_ModeType ComM_Mode )</pre>	
<b>Service ID [hex]</b>	0x02	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant (only for different network handles)	
<b>Parameters (in)</b>	network	Handle of destined communication network for request
	ComM_Mode	Requested communication mode
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
<b>Description</b>	This service shall change the communication mode of a CAN network to the requested one.	



<b>Available via</b>	CanSM.h
----------------------	---------

](SRS\_Can\_01145, SRS\_Can\_01142)

Remark: Please refer to [10] for a detailed description of the communication modes.

**[SWS\_CanSM\_00369]** [ The function `CanSM_RequestComMode` shall accept its request, if the `NetworkHandle` parameter of the request is a handle contained in the configuration of the CanSM module (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00370]** [ The function `CanSM_RequestComMode` shall deny its request, if the `NetworkHandle` parameter of the request is not a handle contained in the configuration of the CanSM module (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00555]** [ The CanSM module shall deny the API request `CanSM_RequestComMode`, if the initial transition for the requested CAN network is not finished yet after the `CanSM_Init` request (ref. to [SWS\\_CanSM\\_00423](#), [SWS\\_CanSM\\_00430](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00183]** [ The function `CanSM_RequestComMode` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_INVALID_NETWORK_HANDLE`, if it does not accept the network handle of the request.] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00182]** [ If the function `CanSM_RequestComMode` accepts the request, the request shall be considered by the CanSM state machine (ref. to [SWS\\_CanSM\\_00635](#)).] (SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00184]** [ If the CanSM module is not initialized, when the function `CanSM_RequestComMode` is called, then this function shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT`.] (SRS\_BSW\_00406)

### 8.3.4 CanSM\_GetCurrentComMode

**[SWS\_CanSM\_00063]**

<b>Service Name</b>	CanSM_GetCurrentComMode
<b>Syntax</b>	<pre>Std_ReturnType CanSM_GetCurrentComMode (     NetworkHandleType network,     ComM_ModeType* ComM_ModePtr )</pre>
<b>Service ID [hex]</b>	0x03
<b>Sync/Async</b>	Synchronous

<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	network	Network handle, whose current communication mode shall be put out
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	ComM_Mode Ptr	Pointer, where to put out the current communication mode
<b>Return value</b>	Std_Return-Type	E_OK: Service accepted E_NOT_OK: Service denied
<b>Description</b>	This service shall put out the current communication mode of a CAN network.	
<b>Available via</b>	CanSM.h	

|(SRS\_ModeMgm\_09084)

**[SWS\_CanSM\_00282]** | The CanSM module shall return E\_NOT\_OK for the API request `CanSM_GetCurrentComMode` until the call of the provided API `CanSM_Init` (ref. to [SWS\\_CANSM\\_00023](#)).| (SRS\_Can\_01142)

**[SWS\_CanSM\_00371]** | The function `CanSM_GetCurrentComMode` shall accept its request, if the `NetworkHandle` parameter of the request is a handle contained in the configuration of the CanSM module (ref. to [ECUC\\_CanSM\\_00161](#)).| (SRS\_Can\_01142)

**[SWS\_CanSM\_00372]** | The function `CanSM_GetCurrentComMode` shall deny its request, if the `NetworkHandle` parameter of the request is not a handle contained in the configuration of the CanSM module (ref. to [ECUC\\_CanSM\\_00161](#)).| (SRS\_Can\_01142)

**[SWS\_CanSM\_00187]** | The function `CanSM_GetCurrentComMode` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_INVALID_NETWORK_HANDLE`, if it does not accept the network handle of the request.| (SRS\_Can\_01142)

**[SWS\_CanSM\_00186]** | The function `CanSM_GetCurrentComMode` shall put out the current communication mode for the network handle (ref. to [SWS\\_CanSM\\_00266](#)) to the designated pointer of type `ComM_ModeType`, if it accepts the request.| (SRS\_Can\_01142)

**[SWS\_CanSM\_00188]** | If the CanSM module is not initialized (ref. to [SWS\\_CANSM\\_00282](#)), when the function `CanSM_GetCurrentComMode` is called, then this function shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT`.| (SRS\_Can\_01142)

**[SWS\_CanSM\_00360]** | The function `CanSM_GetCurrentComMode` shall report the development error `CANSM_E_PARAM_POINTER` to the DET, if the user of this function hands over a NULL-pointer as `ComM_ModePtr`.] (SRS\_Can\_01142)

### 8.3.5 CanSM\_StartWakeupSource

**[SWS\_CanSM\_00609]**

<b>Service Name</b>	CanSM_StartWakeupSource	
<b>Syntax</b>	Std_ReturnType CanSM_StartWakeupSource ( NetworkHandleType network )	
<b>Service ID [hex]</b>	0x11	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	network	Affected CAN network
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request denied
<b>Description</b>	This function shall be called by EcuM when a wakeup source shall be started.	
<b>Available via</b>	CanSM.h	

] (SRS\_Can\_01145)

**[SWS\_CanSM\_00611]** | The API function `CanSM_StartWakeupSource` shall return `E_NOT_OK`, if the `CanSM` module is not initialized yet with `CanSM_Init` (ref. to [SWS\\_CANSM\\_00023](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00617]** | The function `CanSM_StartWakeupSource` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT`, if the `CanSM` module is not initialized yet with `CanSM_Init` (ref. to [SWS\\_CANSM\\_00023](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00612]** | The function `CanSM_StartWakeupSource` shall return `E_NOT_OK`, if the `CanSM` module is initialized and the `network` parameter of the request is not a handle contained in the configuration of the `CanSM` module (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00613]** | The function `CanSM_StartWakeupSource` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_INVALID_NETWORK_HANDLE`, if the `CanSM` module is initialized and the

requested handle is invalid concerning the CanSM configuration (ref. to [ECUC CanSM\\_00161](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00616]**] The function `CanSM_StartWakeupSource` shall return `E_OK` and it shall be considered as trigger (ref. to [SWS\\_CanSM\\_00607](#)) for the state machine of the related network, if the CanSM module is initialized and the requested handle is valid concerning the CanSM configuration (ref. to [ECUC\\_CanSM\\_00161](#)). ] (SRS\_Can\_01145)

### 8.3.6 CanSM\_StopWakeupSource

**[SWS\_CanSM\_00610]**

<b>Service Name</b>	CanSM_StopWakeupSource	
<b>Syntax</b>	Std_ReturnType CanSM_StopWakeupSource ( NetworkHandleType network )	
<b>Service ID [hex]</b>	0x12	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	network	Affected CAN network
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request denied
<b>Description</b>	This function shall be called by EcuM when a wakeup source shall be stopped.	
<b>Available via</b>	CanSM.h	

](SRS\_Can\_01145)

**[SWS\_CanSM\_00618]**] The API function `CanSM_StopWakeupSource` shall return `E_NOT_OK`, if the CanSM module is not initialized yet with `CanSM_Init` (ref. to [SWS\\_CanSM\\_00023](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00619]**] The function `CanSM_StopWakeupSource` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT`, if the CanSM module is not initialized yet with `CanSM_Init` (ref. to [SWS\\_CanSM\\_00023](#)). ] (SRS\_Can\_01145)

**[SWS\_CanSM\_00620]**] The function `CanSM_StopWakeupSource` shall return `E_NOT_OK`, if the CanSM module is initialized and the `network` parameter of the request is not a handle contained in the configuration of the CanSM module (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00621]** The function `CanSM_StopWakeupSource` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_INVALID_NETWORK_HANDLE`, if the CanSM module is initialized and the requested handle is invalid concerning the CanSM configuration (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00622]** The function `CanSM_StopWakeupSource` shall return `E_OK` and it shall be considered as trigger (ref. to [SWS\\_CanSM\\_00608](#)) for the state machine of the related network, if the CanSM module is initialized and the requested handle is valid concerning the CanSM configuration (ref. to [ECUC\\_CanSM\\_00161](#)).] (SRS\_Can\_01145)

### 8.3.7 Optional

#### 8.3.7.1 CanSM\_GetVersionInfo

[SWS\_CanSM\_00024]

<b>Service Name</b>	CanSM_GetVersionInfo	
<b>Syntax</b>	<pre>void CanSM_GetVersionInfo (     Std_VersionInfoType* VersionInfo )</pre>	
<b>Service ID [hex]</b>	0x01	
<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>	This service puts out the version information of this module (module ID, vendor ID, vendor specific version numbers related to BSW00407)	
<b>Available via</b>	CanSM.h	

](SRS\_BSW\_00407, SRS\_BSW\_00003)

[SWS\_CanSM\_00374] | The function `CanSM_GetVersionInfo` shall report the development error `CANSM_E_PARAM_POINTER` to the DET, if the user of this function hands over a NULL-pointer as `VersionInfo`.] (SRS\_BSW\_00407, SRS\_BSW\_00003)

#### 8.3.7.2 CanSM\_SetBaudrate

[SWS\_CanSM\_00561]

<b>Service Name</b>	CanSM_SetBaudrate	
<b>Syntax</b>	<pre>Std_ReturnType CanSM_SetBaudrate (     NetworkHandleType Network,     uint16 BaudRateConfigID )</pre>	
<b>Service ID [hex]</b>	0x0d	
<b>Sync/Async</b>	Synchronous	

<b>Reentrancy</b>	Reentrant for different Networks. Non reentrant for the same Network.	
<b>Parameters (in)</b>	Network	Handle of the addressed CAN network for the baud rate change
	BaudRateConfigID	references a baud rate configuration by ID (see CanController BaudRateConfigID)
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	E_OK: Service request accepted, setting of (new) baud rate started E_NOT_OK: Service request not accepted
<b>Description</b>	This service shall start an asynchronous process to change the baud rate for the configured CAN controllers of a certain CAN network. Depending on necessary baud rate modifications the controllers might have to reset.	
<b>Available via</b>	CanSM.h	

](SRS\_Can\_01142)

**[SWS\_CanSM\_00569]** | The CanSM module shall provide the API function `CanSM_SetBaudrate`, if the `CANSM_SET_BAUDRATE_API` parameter (ref. to [ECUC\\_CanSM\\_00343](#)) is configured with the value `TRUE`. | (SRS\_Can\_01142)

**[SWS\_CanSM\_00570]** The CanSM module shall not provide the API function `CanSM_SetBaudrate`, if the `CANSM_SET_BAUDRATE_API` parameter (ref. to [ECUC\\_CanSM\\_00343](#)) is configured with the value `FALSE`. | (SRS\_Can\_01142)

**[SWS\_CanSM\_00502]** | The CanSM module shall deny the `CanSM_SetBaudrate` API request, if the `NetworkHandle` parameter does not match to the configured Network handles of the CanSM module (ref. to [ECUC\\_CanSM\\_00161](#)). | (SRS\_Can\_01142)

**[SWS\_CanSM\_00504]** [ The function `CanSM_SetBaudrate` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_INVALID_NETWORK_HANDLE` (ref. to chapter 7.3), if it does not accept the network handle of the request.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00505]** [ The function `CanSM_SetBaudrate` shall deny its request, if the requested CAN network is not in the communication mode `COMM_FULL_COMMUNICATION`.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00530]** [ The CanSM module shall deny the `CanSM_SetBaudrate` API request, if the CanSM module is not initialized.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00506]** [ If the function `CanSM_SetBaudrate` is called and the CanSM module is not initialized, then this function shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT` (ref. to chapter 7.3).] (SRS\_Can\_01142)

**[SWS\_CanSM\_00503]** [ If no condition is present to deny the `CanSM_SetBaudrate` request according to [SWS\\_CANSM\\_00502](#) and [SWS\\_CANSM\\_00505](#), [SWS\\_CANSM\\_00530](#), then the CanSM module shall return `E_OK` and operate the process for the requested baud rate change as specified with [SWS\\_CANSM\\_00507](#).] (SRS\_Can\_01142)

### 8.3.7.3 CanSM\_SetIcomConfiguration

**[SWS\_CanSM\_00586]**

<b>Service Name</b>	CanSM_SetIcomConfiguration	
<b>Syntax</b>	<pre>Std_ReturnType CanSM_SetIcomConfiguration (     NetworkHandleType Network,     IcomConfigIdType ConfigurationId )</pre>	
<b>Service ID [hex]</b>	0x0f	
<b>Sync/Async</b>	Asynchronous	
<b>Reentrancy</b>	Reentrant only for different network handles	
<b>Parameters (in)</b>	Network	Handle of destined communication network for request
	ConfigurationId	Requested Configuration
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_ReturnType	<code>E_OK</code> : Request accepted <code>E_NOT_OK</code> : Request denied
<b>Description</b>	This service shall change the Icom Configuration of a CAN network to the requested one.	
<b>Available via</b>	CanSM.h	



](SRS\_Can\_01142)

**[SWS\_CanSM\_00599]** The CanSM module shall provide the API function `CanSM_SetIcomConfiguration`, if the `CANSM_ICOM_SUPPORT` parameter (ref. to [ECUC\\_CanSM\\_00345](#)) is configured with the value `TRUE`.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00593]** If the requested Network is configured for the CanSM module, the API `CanSM_SetIcomConfiguration` shall request an ICOM configuration for a given channel in order to activate or deactivate Pretended Networking (ref. to chapter 7.4) and return `E_OK` or `E_NOT_OK` depending on the return value of the requested `CanIf` API.] (SRS\_Can\_01142)

**[SWS\_CanSM\_00594]** If the requested Network is not configured for the CanSM module, the API `CanSM_SetIcomConfiguration` shall return `E_NOT_OK` and notify the DET error `CANSM_E_INVALID_NETWORK_HANDLE`.] (SRS\_Can\_01142)

### 8.3.7.4 CanSM\_SetEcuPassive

**[SWS\_CanSM\_00644]**

<b>Service Name</b>	CanSM_SetEcuPassive	
<b>Syntax</b>	<pre>Std_ReturnType CanSM_SetEcuPassive (     boolean CanSM_Passive )</pre>	
<b>Service ID [hex]</b>	0x13	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Non Reentrant	
<b>Parameters (in)</b>	CanSM_Passive	TRUE: set all CanSM channels to passive, i.e. receive only FALSE: set all CanSM channels back to non-passive
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	Std_Return-Type	<code>E_OK</code> : Request accepted <code>E_NOT_OK</code> : Request not accepted
<b>Description</b>	This function can be used to set all CanSM channels of the ECU to a receive only mode. This mode will be kept either until it is set back, or the ECU is reset.	
<b>Available via</b>	CanSM.h	

](SRS\_Can\_01158)

**[SWS\_CanSM\_00645]** The CanSM module shall provide the API function `CanSM_SetEcuPassive`, if the `CanSMTxOfflineActiveSupport` parameter (ref. to [ECUC\\_CanSM\\_00349](#)) is configured with the value `TRUE`.] (SRS\_Can\_01158)

### 8.3.8 Call-back notifications

This is a list of functions provided for other modules.

### 8.3.9 CanSM\_ControllerBusOff

#### [SWS\_CanSM\_00064]

<b>Service Name</b>	CanSM_ControllerBusOff	
<b>Syntax</b>	<pre>void CanSM_ControllerBusOff (     uint8 ControllerId )</pre>	
<b>Service ID [hex]</b>	0x04	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant (only for different CanControllers)	
<b>Parameters (in)</b>	ControllerId	CAN controller, which detected a bus-off event
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This callback function notifies the CanSM about a bus-off event on a certain CAN controller, which needs to be considered with the specified bus-off recovery handling for the impacted CAN network.	
<b>Available via</b>	CanSM_CanIf.h	

](SRS\_BSW\_00359, SRS\_BSW\_00333)

**[SWS\_CanSM\_00189]** [ If the function `CanSM_ControllerBusOff` gets a `Controller`, which is not configured as `CanSMControllerId` in the configuration of the CanSM module, it shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_PARAM_CONTROLLER.`] (SRS\_BSW\_00359, SRS\_BSW\_00333)

**[SWS\_CanSM\_00190]** [ If the CanSM module is not initialized, when the function `CanSM_ControllerBusOff` is called, then the function `CanSM_ControllerBusOff` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT.`] (SRS\_BSW\_00359, SRS\_BSW\_00333)

**[SWS\_CanSM\_00235]** [ If the CanSM module is initialized and the input parameter `Controller` is one of the CAN controllers configured with the parameter

CanSMControllerId, this bus-off event shall be considered by the CAN Network state machine (ref. to [SWS\\_CanSM\\_00500](#)).] (SRS\_BSW\_00359, SRS\_BSW\_00333)

Additional remarks:

- 1.) The call context is either on interrupt level (interrupt mode) or on task level (polling mode).
- 2.) Reentrancy is necessary for multiple CAN controller usage.

### 8.3.10 CanSM\_ControllerModeIndication

[SWS\_CanSM\_00396]

<b>Service Name</b>	CanSM_ControllerModeIndication	
<b>Syntax</b>	<pre>void CanSM_ControllerModeIndication (     uint8 ControllerId,     Can_ControllerStateType ControllerMode )</pre>	
<b>Service ID [hex]</b>	0x07	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant (only for different CAN controllers)	
<b>Parameters (in)</b>	ControllerId	CAN controller, whose mode has changed
	ControllerMode	Notified CAN controller mode
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This callback shall notify the CanSM module about a CAN controller mode change.	
<b>Available via</b>	CanSM_CanIf.h	

](SRS\_Can\_01145)

[SWS\_CanSM\_00397] [ If the function CanSM\_ControllerModeIndication gets a ControllerId, which is not configured as CanSMControllerId in the configuration of the CanSM module, it shall call the function Det\_ReportError with ErrorId parameter CANSM\_E\_PARAM\_CONTROLLER.] (SRS\_Can\_01145)

[SWS\_CanSM\_00398] [ If the CanSM module is not initialized, when the function CanSM\_ControllerModeIndication is called, then the function CanSM\_ControllerModeIndication shall call the function Det\_ReportError with ErrorId parameter CANSM\_E\_UNINIT.] (SRS\_Can\_01145)

### 8.3.11 CanSM\_TransceiverModeIndication

#### [SWS\_CanSM\_00399]

<b>Service Name</b>	CanSM_TransceiverModeIndication	
<b>Syntax</b>	<pre>void CanSM_TransceiverModeIndication (     uint8 TransceiverId,     CanTrcv_TrcevModeType TransceiverMode )</pre>	
<b>Service ID [hex]</b>	0x09	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different CAN Transceivers	
<b>Parameters (in)</b>	TransceiverId	CAN transceiver, whose mode has changed
	TransceiverMode	Notified CAN transceiver mode
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This callback shall notify the CanSM module about a CAN transceiver mode change.	
<b>Available via</b>	CanSM_CanIf.h	

|(SRS\_Can\_01145, SRS\_Can\_01142)

**[SWS\_CanSM\_00400]** | If the function `CanSM_TransceiverModeIndication` gets a `TransceiverId`, which is not configured as `CanSMTransceiverId` in the configuration of the CanSM module, it shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_PARAM_TRANSCEIVER.` (SRS\_Can\_01145)

**[SWS\_CanSM\_00401]** | If the CanSM module is not initialized, when the function `CanSM_TransceiverModeIndication` is called, then the function `CanSM_TransceiverModeIndication` shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_UNINIT.` (SRS\_Can\_01145)

### 8.3.12 CanSM\_TxTimeoutException

#### [SWS\_CanSM\_00410]

<b>Service Name</b>	CanSM_TxTimeoutException	
<b>Syntax</b>	<pre>void CanSM_TxTimeoutException (     NetworkHandleType Channel )</pre>	
<b>Service ID</b>	0x0b	

<b>[hex]</b>	
<b>Sync/Async</b>	Synchronous
<b>Reentrancy</b>	Reentrant
<b>Parameters (in)</b>	Channel                      Affected CAN network
<b>Parameters (inout)</b>	None
<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	This function shall notify the CanSM module, that the CanNm has detected for the affected partial CAN network a tx timeout exception, which shall be recovered within the respective network state machine of the CanSM module.
<b>Available via</b>	CanSM_CanIf.h

|(SRS\_Can\_01142, SRS\_Can\_01145)

**[SWS\_CanSM\_00411]** | The function `CanSM_TxTimeoutException` shall report `CANSM_E_UNINIT` to the DET, if the CanSM is not initialized yet. | (SRS\_Can\_01145)

**[SWS\_CanSM\_00412]** | If the function `CanSM_TxTimeoutException` is referenced with a `Channel`, which is not configured as `CanSMNetworkHandle` in the CanSM configuration, it shall report `CANSM_E_INVALID_NETWORK_HANDLE` to the DET. | (SRS\_Can\_01145)

Remarks: Reentrancy is necessary for different Channels.

### 8.3.13 CanSM\_ClearTrcvWufFlagIndication

**[SWS\_CanSM\_00413]** |

<b>Service Name</b>	CanSM_ClearTrcvWufFlagIndication	
<b>Syntax</b>	<pre>void CanSM_ClearTrcvWufFlagIndication (     uint8 Transceiver )</pre>	
<b>Service ID [hex]</b>	0x08	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different CAN Transceivers	
<b>Parameters (in)</b>	Transceiver	Requested Transceiver
<b>Parameters (inout)</b>	None	

<b>Parameters (out)</b>	None
<b>Return value</b>	None
<b>Description</b>	This callback function shall indicate the CanIf_ClearTrcvWufFlag API process end for the notified CAN Transceiver.
<b>Available via</b>	CanSM_CanIf.h

](SRS\_Can\_01145)

**[SWS\_CanSM\_00414]** [ The function `CanSM_ClearTrcvWufFlagIndication` shall report `CANSM_E_UNINIT` to the DET, if the CanSM is not initialized yet.] (SRS\_Can\_01145)

**[SWS\_CanSM\_00415]** [ If the function `CanSM_ClearTrcvWufFlagIndication` gets a `TransceiverId`, which is not configured (ref. to [ECUC\\_CanSM\\_00137](#)) in the configuration of the CanSM module, it shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_PARAM_TRANSCEIVER`.] (SRS\_Can\_01145)

### 8.3.14 CanSM\_CheckTransceiverWakeFlagIndication

**[SWS\_CanSM\_00416]**

<b>Service Name</b>	CanSM_CheckTransceiverWakeFlagIndication	
<b>Syntax</b>	<pre>void CanSM_CheckTransceiverWakeFlagIndication (     uint8 Transceiver )</pre>	
<b>Service ID [hex]</b>	0x0a	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different CAN Transceivers	
<b>Parameters (in)</b>	Transceiver	Requested Transceiver
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This callback function indicates the CanIf_CheckTrcvWakeFlag API process end for the notified CAN Transceiver.	
<b>Available via</b>	CanSM_CanIf.h	

](SRS\_Can\_01145)

**[SWS\_CanSM\_00417]** [ The function `CanSM_CheckTransceiverWakeFlagIndication` shall report `CANSM_E_UNINIT` to the DET, if the `CanSM` module is not initialized yet.] (SRS\_Can\_01145)

**[SWS\_CanSM\_00418]** [ If the function `CanSM_CheckTransceiverWakeFlagIndication` gets a `TransceiverId`, which is not configured (ref. to [ECUC\\_CanSM\\_00137](#)) in the configuration of the `CanSM` module, it shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_PARAM_TRANSCEIVER`.] (SRS\_Can\_01145)

### 8.3.15 CanSM\_ConfirmPnAvailability

**[SWS\_CanSM\_00419]**

<b>Service Name</b>	CanSM_ConfirmPnAvailability	
<b>Syntax</b>	<pre>void CanSM_ConfirmPnAvailability (     uint8 TransceiverId )</pre>	
<b>Service ID [hex]</b>	0x06	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant	
<b>Parameters (in)</b>	TransceiverId	CAN transceiver, which was checked for PN availability
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This callback function indicates that the transceiver is running in PN communication mode.	
<b>Available via</b>	CanSM_CanIf.h	

](SRS\_Can\_01145)

**[SWS\_CanSM\_00546]** [ The function `CanSM_ConfirmPnAvailability` shall notify the `CanNm` module (ref. to [SWS\\_CanSM\\_00422](#)), if it is called with a configured `Transceiver` as input parameter (ref. to [ECUC\\_CanSM\\_00137](#)).] (SRS\_Can\_01145)

**[SWS\_CanSM\_00420]** [

The function `CanSM_ConfirmPnAvailability` shall report `CANSM_E_UNINIT` to the DET, if the `CanSM` module is not initialized yet.] (SRS\_Can\_01145)

**[SWS\_CanSM\_00421]** [

If the function `CanSM_ConfirmPnAvailability` gets a `TransceiverId`, which is not configured (ref. to [ECUC\\_CanSM\\_00137](#)) in the configuration of the `CanSM`

module, it shall call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_PARAM_TRANSCEIVER.`] (SRS\_Can\_01145)

### 8.3.16 CanSM\_CurrentIcomConfiguration

[SWS\_CanSM\_00587]

<b>Service Name</b>	CanSM_CurrentIcomConfiguration	
<b>Syntax</b>	<pre>void CanSM_CurrentIcomConfiguration (     uint8 ControllerId,     IcomConfigIdType ConfigurationId,     IcomSwitch_ErrorType Error )</pre>	
<b>Service ID [hex]</b>	0x10	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant only for different network handles	
<b>Parameters (in)</b>	ControllerId	CAN Controller Id, whose configuration has changed.
	Configuration Id	Changed Configuration Id
	Error	ICOM_SWITCH_E_OK: No Error ICOM_SWITCH_E_FAILED: Switch to requested Configuration failed. Severe Error.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	None	
<b>Return value</b>	None	
<b>Description</b>	This service shall inform about the change of the Icom Configuration of a CAN network.	
<b>Available via</b>	CanSM.h	

](SRS\_Can\_01142)

[SWS\_CanSM\_00595] If the `CANSM_ICOM_SUPPORT` parameter (ref. to [ECUC\\_CanSM\\_00345](#)) is configured with the value `TRUE`, then the callback function `BswM_CanSM_CurrentIcomConfiguration` shall notify the BswM about the status of activation or deactivation of Pretended Networking (ref. to chapter 7.4) for the CAN Network, which contains the notified `ControllerId` in its configuration. It shall transfer the `ConfigurationId` and `Error` parameter to the BswM therefore.] (SRS\_Can\_01142)



## 8.4 Scheduled functions

For details refer to the chapter 8.5 “Scheduled functions” in *SWS\_BSWGeneral*.

### 8.4.1 CanSM\_MainFunction

[SWS\_CanSM\_00065]

<b>Service Name</b>	CanSM_MainFunction
<b>Syntax</b>	void CanSM_MainFunction ( void )
<b>Service ID [hex]</b>	0x05
<b>Description</b>	Scheduled function of the CanSM
<b>Available via</b>	SchM_CanSM.h

](SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_Can\_01145, SRS\_Can\_01142)  
[SWS\_CanSM\_00167] [ The main function of the CanSM module shall operate the effects of the CanSM state machine (ref. to chapter 7.2), which the CanSM module shall implement for each configured CAN Network.] (SRS\_BSW\_00424, SRS\_BSW\_00425, SRS\_Can\_01145, SRS\_Can\_01142)

## 8.5 Expected Interfaces

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

### 8.5.1 Mandatory Interfaces

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

<b>API Function</b>	<b>Header File</b>	<b>Description</b>
BswM_CanSM_- CurrentIcom- Configuration	BswM_ CanSM.h	Function to inform BswM about the switch of Icom Configuration.
BswM_CanSM_- CurrentState	BswM_ CanSM.h	Function called by CanSM to indicate its current state.
CanIf_CheckTrcv- WakeFlag	CanIf.h	Requests the CanIf module to check the Wake flag of the designated CAN transceiver.
CanIf_ClearTrcvWuf- Flag	CanIf.h	Requests the CanIf module to clear the WUF flag of the designated CAN transceiver.
CanIf_GetPduMode	CanIf.h	This service reports the current mode of a requested PDU channel.

CanIf_GetTx-ConfirmationState	CanIf.h	This service reports, if any TX confirmation has been done for the whole CAN controller since the last CAN controller start.
CanIf_SetController-Mode	CanIf.h	This service calls the corresponding CAN Driver service for changing of the CAN controller mode.
CanIf_SetPduMode	CanIf.h	This service sets the requested mode at the L-PDUs of a predefined logical PDU channel.
CanIf_SetTrcvMode	CanIf.h	This service changes the operation mode of the transceiver TransceiverId, via calling the corresponding CAN Transceiver Driver service.
CanNm_ConfirmPn-Availability	CanNm.h	Enables the PN filter functionality on the indicated NM channel. Availability: The API is only available if CanNmGlobalPnSupport is TRUE.
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.
Dem_SetEventStatus	Dem.h	Called by SW-Cs or BSW modules to report monitor status information to the Dem. BSW modules calling Dem_SetEventStatus can safely ignore the return value.
Det_ReportRuntime-Error	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.

J()

### 8.5.1.1 Remark: Usage of CanIf\_SetPduMode

Although the CanIf module provides more requestable PDU modes, the CanSM module only uses the parameters CANIF\_ONLINE, CANIF\_TX\_OFFLINE\_ACTIVE and CANIF\_TX\_OFFLINE for the call of the API CanIf\_SetPduMode.

The CANIF\_OFFLINE mode is assumed automatically by CanIf and needs not to be set by CanSM.

## 8.5.2 Optional Interfaces

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

<b>API Function</b>	<b>Header File</b>	<b>Description</b>
CanIf_SetBaudrate	CanIf.h	This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.
CanIf_SetIcom-Configuration	CanIf.h	This service shall change the Icom Configuration of a CAN controller to the requested one.
Det_ReportError	Det.h	Service to report development errors.

J()

### 8.5.3 Configurable Interfaces

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

#### 8.5.3.1 <User\_GetBusOffDelay>

[SWS\_CanSM\_00637]

<b>Service Name</b>	<User_GetBusOffDelay>	
<b>Syntax</b>	<pre>void &lt;User_GetBusOffDelay&gt; (     NetworkHandleType network,     uint8* delayCyclesPtr )</pre>	
<b>Sync/Async</b>	Synchronous	
<b>Reentrancy</b>	Reentrant for different networks	
<b>Parameters (in)</b>	network	CAN network where a BusOff occurred.
<b>Parameters (inout)</b>	None	
<b>Parameters (out)</b>	delayCyclesPtr	Number of CanSM base cycles to wait additionally to L1/L2 after a BusOff occurred.
<b>Return value</b>	None	
<b>Description</b>	This callout function returns the number of CanSM base cycles to wait additionally to L1/L2 after a BusOff occurred.	
<b>Available via</b>	configurable	

](SRS\_Can\_01144, SRS\_Can\_01146)

## 9 Sequence diagrams

All interactions of the CanSM module with the depending modules CanIf, ComM, BswM, Dem and CanNm are specified in the state machine diagrams (ref. to

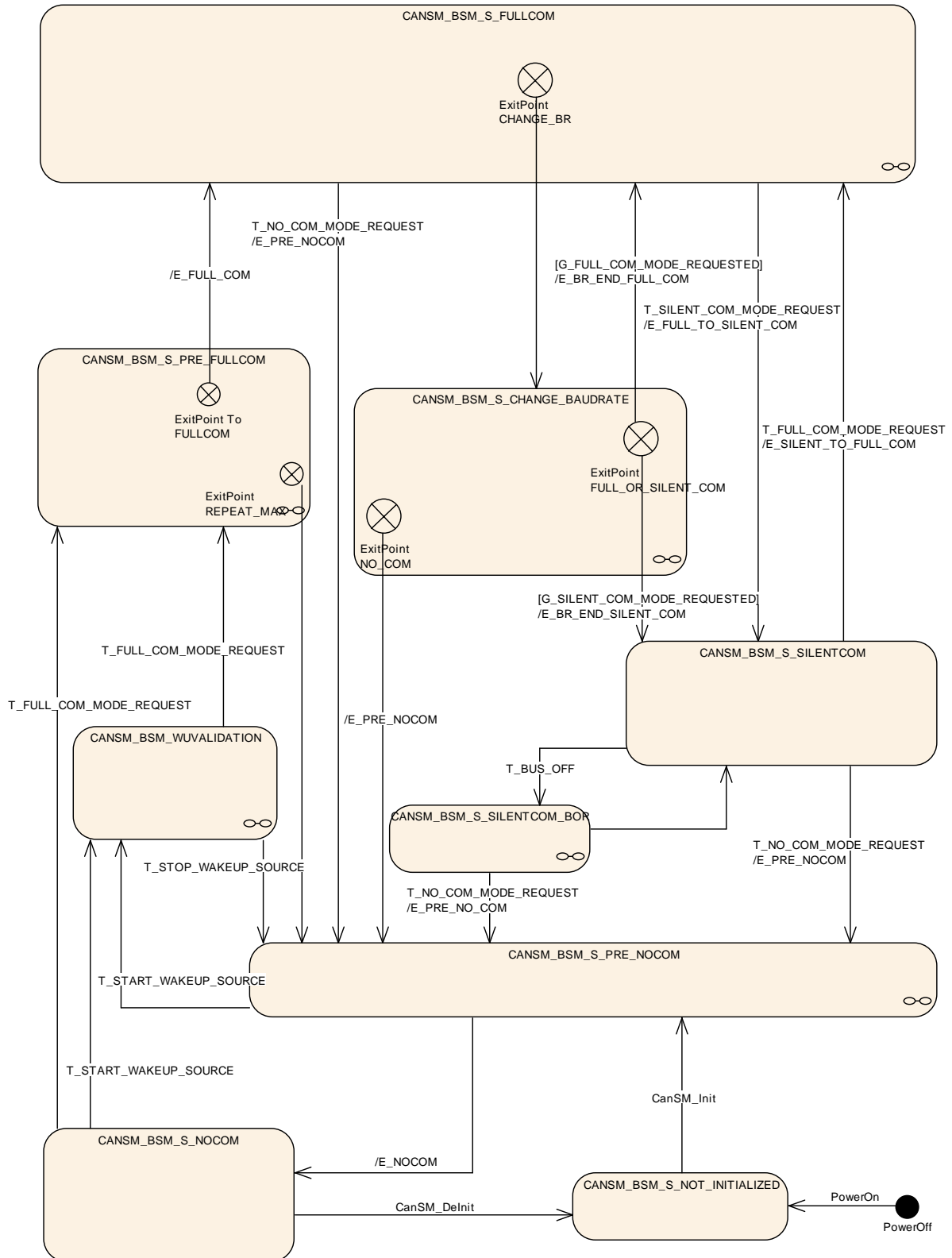


Figure 7-1- Figure 7-10). Therefore the CanSM SWS provides only some exemplary

sequences for the use case to start and to stop the CAN controller(s) of a CAN network.

Remark: For the special use case of CAN network deinitialization with partial network support please refer to chapter 9 of [9] (Specification of CAN Transceiver Driver).

### 9.1 Sequence diagram CanSm\_StartCanController

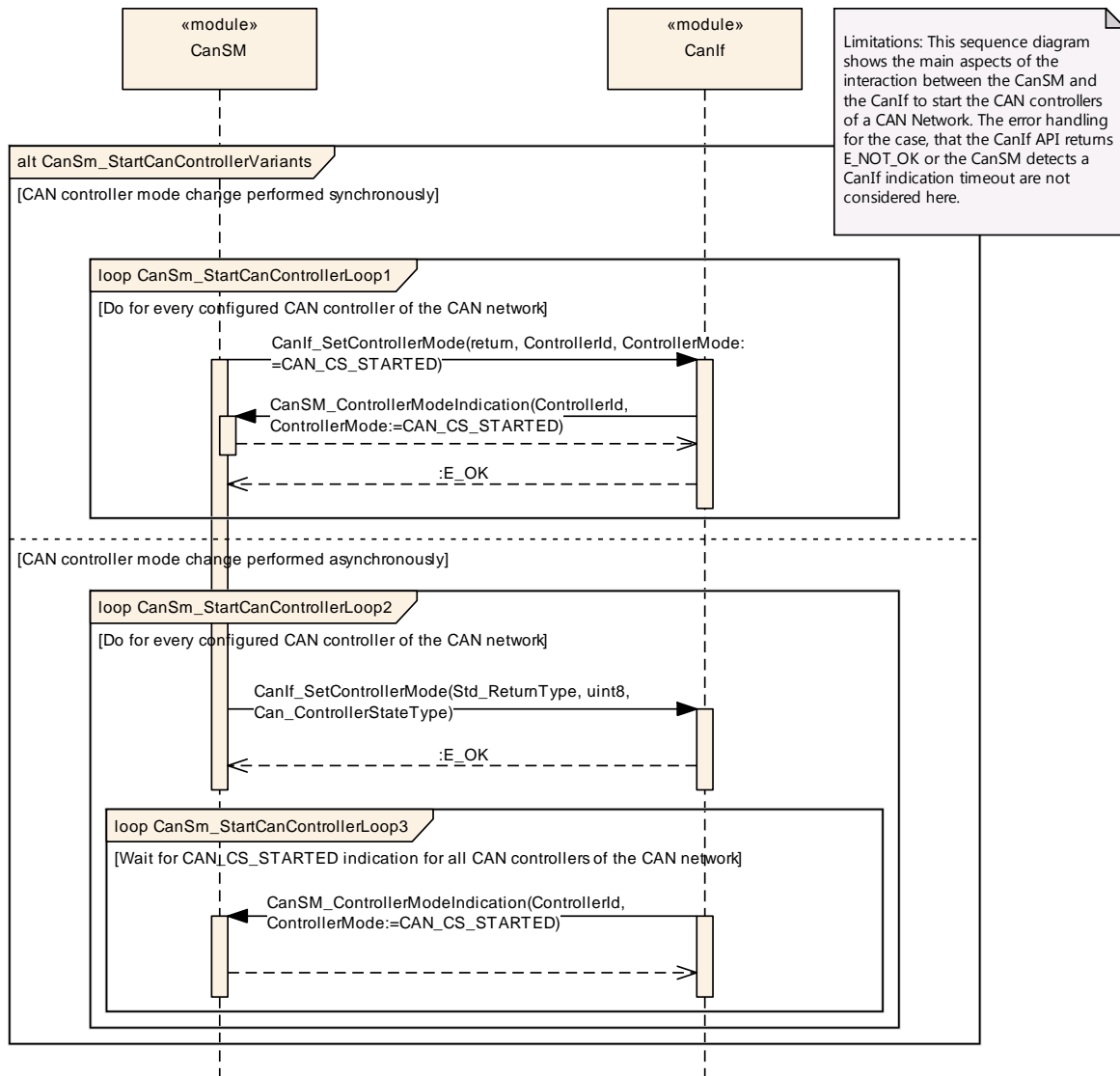


Figure 9-1: Sequence diagram CanSm\_StartCanController

## 9.2 Sequence diagram CanSm\_StopCanController

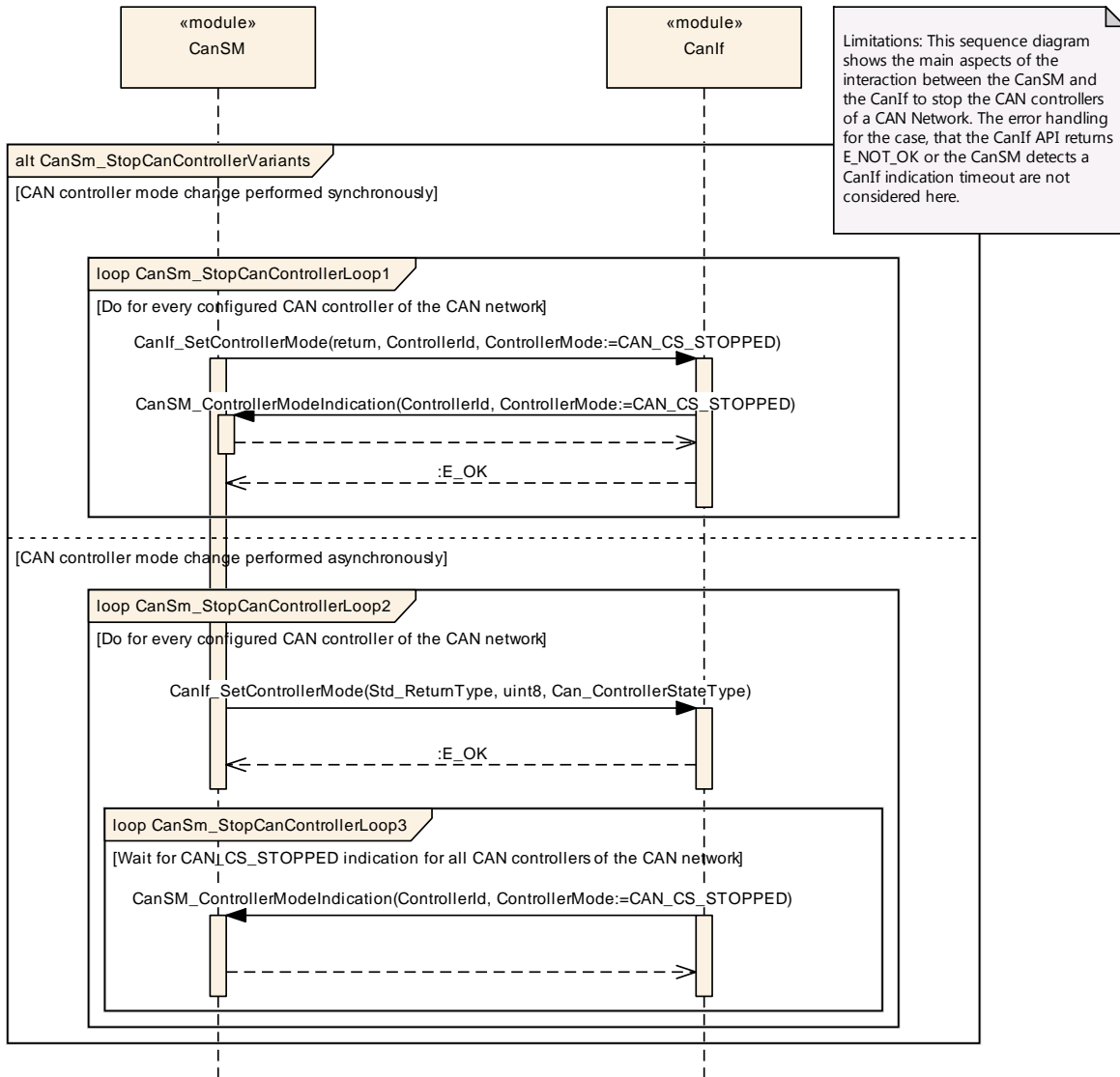


Figure 9-2: Sequence diagram CanSm\_StopCanController

## 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. It also specifies a template (table) you shall use for the parameter specification. We intend to leave Chapter 10.1 in the specification to guarantee comprehension.

Chapter 10.2 specifies the structure (containers) and the parameters of the module CanSM.

Chapter 10.3 specifies published information of the module CanSM.

### 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 of the CanSM module. The detailed meanings of the parameters describe chapter 7 and chapter 8.

#### 10.2.1 CanSM

<b>SWS Item</b>	<b>ECUC_CanSM_00351 :</b>
<b>Module Name</b>	<i>CanSM</i>
<b>Module Description</b>	Configuration of the CanSM module
<b>Post-Build Variant Support</b>	true
<b>Supported Config Variants</b>	VARIANT-LINK-TIME, VARIANT-POST-BUILD, VARIANT-PRE-COMPILE

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanSMConfiguration	1	This container contains the global parameters of the CanSM and sub containers, which are for the CAN network specific configuration.
CanSMGeneral	1	Container for general pre-compile parameters of the CanSM module

#### 10.2.2 CanSMConfiguration

<b>SWS Item</b>	<b>ECUC_CanSM_00123 :</b>
<b>Container Name</b>	CanSMConfiguration
<b>Parent Container</b>	CanSM
<b>Description</b>	This container contains the global parameters of the CanSM and sub containers, which are for the CAN network specific configuration.
<b>Configuration Parameters</b>	

<b>SWS Item</b>	<b>ECUC_CanSM_00335 :</b>		
<b>Name</b>	CanSMModeRequestRepetitionMax		
<b>Parent Container</b>	CanSMConfiguration		
<b>Description</b>	Specifies the maximal amount of mode request repetitions without a respective mode indication from the CanIf module until the CanSM module reports a Development Error to the Det and tries to go back to no communication.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<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_CanSM_00336 :</b>		
<b>Name</b>	CanSMModeRequestRepetitionTime		
<b>Parent Container</b>	CanSMConfiguration		
<b>Description</b>	Specifies in which time duration the CanSM module shall repeat mode change requests by using the API of the CanIf module.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. 65.535]		
<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>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanSMManagerNetwork	1..*	This container contains the CAN network specific parameters of each CAN network

### 10.2.3 CanSMGeneral

<b>SWS Item</b>	<b>ECUC_CanSM_00314 :</b>		
<b>Container Name</b>	CanSMGeneral		
<b>Parent Container</b>	CanSM		
<b>Description</b>	Container for general pre-compile parameters of the CanSM module		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanSM_00133 :</b>		
<b>Name</b>	CanSMDevErrorDetect		
<b>Parent Container</b>	CanSMGeneral		
<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> </ul>		



	<ul style="list-style-type: none"> <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_CanSM_00347 :</b>		
<b>Name</b>	CanSMGetBusOffDelayFunction		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	This parameter configures the name of the <User_GetBusOffDelay> callout function, which is used by CanSM to acquire an additional L1/L2 delay time. This function is only called for channels where CanSMEnableBusOffDelay is enabled.		
<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	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_CanSM_00348 :</b>		
<b>Name</b>	CanSMGetBusOffDelayHeader		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	This parameter configures the header file containing the prototype of the <User_GetBusOffDelay> callout function.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucStringParamDef		
<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	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_CanSM_00345 :</b>		
<b>Name</b>	CanSMIcomSupport		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	Selects support of Pretended Network features in CanSM. True: Enabled False: Disabled		
<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: ECU		

<b>SWS Item</b>	<b>ECUC_CanSM_00312 :</b>		
<b>Name</b>	CanSMMainFunctionTimePeriod		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	This parameter defines the cycle time of the function CanSM_MainFunction in seconds		
<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	All Variants
	<b>Link time</b>	--	
	<b>Post-build time</b>	--	
<b>Scope / Dependency</b>	scope: local		

<b>SWS Item</b>	<b>ECUC_CanSM_00344 :</b>		
<b>Name</b>	CanSMPncSupport		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	Enables or disables support of partial networking. False: Partial Networking is disabled True: Partial Networking is enabled		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<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 dependency: This parameter shall be available only if ComMPncSupport is enabled in ComM		

<b>SWS Item</b>	<b>ECUC_CanSM_00343 :</b>		
<b>Name</b>	CanSMSetBaudrateApi		
<b>Parent Container</b>	CanSMGeneral		

<b>Description</b>	The support of the Can_SetBaudrate API is optional. If this parameter is set to true the Can_SetBaudrate API shall be supported. Otherwise the API is not supported.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<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: ECU		

<b>SWS Item</b>	<b>ECUC_CanSM_00349 :</b>		
<b>Name</b>	CanSMTxOfflineActiveSupport		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	Determines whether the ECU passive feature is supported by CanSM. True: Enabled False: Disabled		
<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 dependency: CanIfTxOfflineActiveSupport		

<b>SWS Item</b>	<b>ECUC_CanSM_00311 :</b>		
<b>Name</b>	CanSMVersionInfoApi		
<b>Parent Container</b>	CanSMGeneral		
<b>Description</b>	Activate/Deactivate the version information API (CanSM_GetVersionInfo). true: version information API activated false: version information API deactivated		
<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**

### 10.2.4 CanSMManagerNetwork

<b>SWS Item</b>	<b>ECUC_CanSM_00126 :</b>		
<b>Container Name</b>	CanSMManagerNetwork		
<b>Parent Container</b>	CanSMConfiguration		
<b>Description</b>	This container contains the CAN network specific parameters of each CAN network		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanSM_00131 :</b>		
<b>Name</b>	CanSMBorCounterL1ToL2		
<b>Parent Container</b>	CanSMManagerNetwork		
<b>Description</b>	This threshold defines the count of bus-offs until the bus-off recovery switches from level 1 (short recovery time) to level 2 (long recovery time).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucIntegerParamDef		
<b>Range</b>	0 .. 255		
<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_CanSM_00128 :</b>		
<b>Name</b>	CanSMBorTimeL1		
<b>Parent Container</b>	CanSMManagerNetwork		
<b>Description</b>	This time parameter defines in seconds the duration of the bus-off recovery time in level 1 (short recovery time).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. 65.535]		
<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_CanSM_00129 :</b>		
<b>Name</b>	CanSMBorTimeL2		
<b>Parent Container</b>	CanSMManagerNetwork		
<b>Description</b>	This time parameter defines in seconds the duration of the bus-off recovery time in level 2 (long recovery time).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. 65.535]		
<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_CanSM_00130 :</b>		
<b>Name</b>	CanSMBorTimeTxEnsured		

<b>Parent Container</b>	CanSMManagerNetwork		
<b>Description</b>	This parameter defines in seconds the duration of the bus-off event check. This check assesses, if the recovery has been successful after the recovery reenables the transmit path. If a new bus-off occurs during this time period, the CanSM assesses this bus-off as sequential bus-off without successful recovery. Because a bus-off only can be detected, when PDUs are transmitted, the time has to be great enough to ensure that PDUs are transmitted again (e. g. time period of the fastest cyclic transmitted PDU of the COM module / ComTxModeTimePeriodFactor).		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucFloatParamDef		
<b>Range</b>	[0 .. 65.535]		
<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: CANSMBOR_TX_CONFIRMATION_POLLING disabled		

<b>SWS Item</b>	<b>ECUC_CanSM_00339 :</b>		
<b>Name</b>	CanSMBorTxConfirmationPolling		
<b>Parent Container</b>	CanSMManagerNetwork		
<b>Description</b>	This parameter shall configure, if the CanSM polls the CanIf_GetTxConfirmationState API to decide the bus-off state to be recovered instead of using the CanSMBorTimeTxEnsured parameter for this decision.		
<b>Multiplicity</b>	1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	--		
<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_CanSM_00346 :</b>		
<b>Name</b>	CanSMEnableBusOffDelay		
<b>Parent Container</b>	CanSMManagerNetwork		
<b>Description</b>	This parameter defines if the <User_GetBusOffDelay> shall be called for this network.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	EcucBooleanParamDef		
<b>Default value</b>	false		
<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_CanSM_00161 :</b>		
-----------------	---------------------------	--	--

<b>Name</b>	CanSMComMNetworkHandleRef		
<b>Parent Container</b>	CanSMMManagerNetwork		
<b>Description</b>	Unique handle to identify one certain CAN network. Reference to one of the network handles configured for the ComM.		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to [ ComMChannel ]		
<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: ComM		

<b>SWS Item</b>	<b>ECUC_CanSM_00137 :</b>		
<b>Name</b>	CanSMTransceiverId		
<b>Parent Container</b>	CanSMMManagerNetwork		
<b>Description</b>	ID of the CAN transceiver assigned to the configured network handle. Reference to one of the transceivers managed by the CanIf module.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ CanIfTrcvCfg ]		
<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: CanIf		

<b>Included Containers</b>		
<b>Container Name</b>	<b>Multiplicity</b>	<b>Scope / Dependency</b>
CanSMController	1..*	This container contains the controller IDs assigned to a CAN network.
CanSMDemEventParameterRefs	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.

### 10.2.5 CanSMController

<b>SWS Item</b>	<b>ECUC_CanSM_00338 :</b>		
<b>Container Name</b>	CanSMController		
<b>Parent Container</b>	CanSMMManagerNetwork		
<b>Description</b>	This container contains the controller IDs assigned to a CAN network.		
<b>Configuration Parameters</b>			

<b>SWS Item</b>	<b>ECUC_CanSM_00141 :</b>		
<b>Name</b>	CanSMControllerId		

<b>Parent Container</b>	CanSMController		
<b>Description</b>	Unique handle to identify one certain CAN controller. Reference to one of the CAN controllers managed by the CanIf module.		
<b>Multiplicity</b>	1		
<b>Type</b>	Symbolic name reference to [ CanIfCtrlCfg ]		
<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: CanIf		

**No Included Containers**

### 10.2.6 CanSMDemEventParameterRefs

<b>SWS Item</b>	<b>ECUC_CanSM_00127 :</b>		
<b>Container Name</b>	CanSMDemEventParameterRefs		
<b>Parent Container</b>	CanSMManagerNetwork		
<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_CanSM_00070 :</b>		
<b>Name</b>	CANSM_E_BUS_OFF		
<b>Parent Container</b>	CanSMDemEventParameterRefs		
<b>Description</b>	Reference to configured DEM event to report bus off errors for this CAN network.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		
<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: Dem		

<b>SWS Item</b>	<b>ECUC_CanSM_00352 :</b>		
<b>Name</b>	CANSM_E_MODE_REQUEST_TIMEOUT		
<b>Parent Container</b>	CanSMDemEventParameterRefs		
<b>Description</b>	Reference to configured DEM event to report bus off errors for this CAN network.		
<b>Multiplicity</b>	0..1		
<b>Type</b>	Symbolic name reference to [ DemEventParameter ]		

<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: Dem		

**No Included Containers**

### 10.3 Published Information

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



## 11 CanSM unspecific / not applicable requirements

**[SWS\_CanSM\_00652]** [ The following requirements are not applicable to this specification, because they are either general BSW requirements, which apply to all BSW modules and not only especially to the CanSM module or they are not applicable at all.] (SRS\_BSW\_00170, SRS\_BSW\_00375, SRS\_BSW\_00395, SRS\_BSW\_00416, SRS\_BSW\_00437, SRS\_BSW\_00168, SRS\_BSW\_00423, SRS\_BSW\_00426, SRS\_BSW\_00427, SRS\_BSW\_00428, SRS\_BSW\_00429, SRS\_BSW\_00432, SRS\_BSW\_00433, SRS\_BSW\_00336, SRS\_BSW\_00417, SRS\_BSW\_00161, SRS\_BSW\_00162, SRS\_BSW\_00005, SRS\_BSW\_00347, SRS\_BSW\_00314, SRS\_BSW\_00353, SRS\_BSW\_00361, SRS\_BSW\_00377, SRS\_BSW\_00308, SRS\_BSW\_00309, SRS\_BSW\_00360, SRS\_BSW\_00341, SRS\_BSW\_00439, SRS\_BSW\_00440, SRS\_BSW\_00004, SRS\_BSW\_00006, SRS\_BSW\_00007, SRS\_BSW\_00009, SRS\_BSW\_00010, SRS\_BSW\_00158, SRS\_BSW\_00159, SRS\_BSW\_00160, SRS\_BSW\_00164, SRS\_BSW\_00167, SRS\_BSW\_00172, SRS\_BSW\_00300, SRS\_BSW\_00301, SRS\_BSW\_00302, SRS\_BSW\_00305, SRS\_BSW\_00306, SRS\_BSW\_00307, SRS\_BSW\_00310, SRS\_BSW\_00312, SRS\_BSW\_00318, SRS\_BSW\_00321, SRS\_BSW\_00323, SRS\_BSW\_00325, SRS\_BSW\_00327, SRS\_BSW\_00328,, SRS\_BSW\_00330, SRS\_BSW\_00331, SRS\_BSW\_00334, SRS\_BSW\_00335, SRS\_BSW\_00339, SRS\_BSW\_00342, SRS\_BSW\_00343, SRS\_BSW\_00346, SRS\_BSW\_00348, SRS\_BSW\_00350, SRS\_BSW\_00357, SRS\_BSW\_00360, SRS\_BSW\_00369, SRS\_BSW\_00371, SRS\_BSW\_00373, SRS\_BSW\_00374, SRS\_BSW\_00378, SRS\_BSW\_00379, SRS\_BSW\_00380, SRS\_BSW\_00383, SRS\_BSW\_00384, SRS\_BSW\_00385, SRS\_BSW\_00386, SRS\_BSW\_00388, SRS\_BSW\_00389, SRS\_BSW\_00390, SRS\_BSW\_00392, SRS\_BSW\_00393, SRS\_BSW\_00394, SRS\_BSW\_00396, SRS\_BSW\_00397, SRS\_BSW\_00398, SRS\_BSW\_00399, SRS\_BSW\_00400, SRS\_BSW\_00401, SRS\_BSW\_00402, SRS\_BSW\_00408, SRS\_BSW\_00409, SRS\_BSW\_00410, SRS\_BSW\_00411, SRS\_BSW\_00413, SRS\_BSW\_00415, SRS\_BSW\_00419, SRS\_BSW\_00422, SRS\_BSW\_00438, SRS\_BSW\_00441, SRS\_BSW\_00442, SRS\_BSW\_00448, SRS\_BSW\_00449, SRS\_BSW\_00450, SRS\_BSW\_00451, SRS\_BSW\_00452, SRS\_BSW\_00453, , SRS\_BSW\_00454, SRS\_BSW\_00456, SRS\_BSW\_00457, SRS\_BSW\_00458, SRS\_BSW\_00459, SRS\_BSW\_00460, SRS\_BSW\_00461, SRS\_BSW\_00462, SRS\_BSW\_00463, SRS\_BSW\_00465, SRS\_BSW\_00466, SRS\_BSW\_00467, SRS\_BSW\_00469, SRS\_BSW\_00470, SRS\_BSW\_00471, SRS\_BSW\_00472, SRS\_Can\_01001, SRS\_Can\_01002, SRS\_Can\_01003, SRS\_Can\_01004, SRS\_Can\_01005, SRS\_Can\_01006, SRS\_Can\_01007, SRS\_Can\_01008, SRS\_Can\_01009, SRS\_Can\_01011, SRS\_Can\_01013, SRS\_Can\_01014, SRS\_Can\_01015, SRS\_Can\_01016, SRS\_Can\_01018, SRS\_Can\_01020, SRS\_Can\_01021, SRS\_Can\_01022, SRS\_Can\_01023, SRS\_Can\_01027, SRS\_Can\_01028, SRS\_Can\_01029, SRS\_Can\_01032, SRS\_Can\_01033, SRS\_Can\_01034, SRS\_Can\_01035, SRS\_Can\_01036, SRS\_Can\_01037, SRS\_Can\_01038, SRS\_Can\_01039, SRS\_Can\_01041, SRS\_Can\_01042, SRS\_Can\_01043, SRS\_Can\_01045, SRS\_Can\_01049, SRS\_Can\_01051, SRS\_Can\_01053, SRS\_Can\_01054, SRS\_Can\_01055, SRS\_Can\_01058, SRS\_Can\_01059, SRS\_Can\_01060, SRS\_Can\_01061, SRS\_Can\_01062, SRS\_Can\_01065, SRS\_Can\_01066, SRS\_Can\_01068, SRS\_Can\_01069, SRS\_Can\_01071, SRS\_Can\_01073, SRS\_Can\_01074, SRS\_Can\_01075, SRS\_Can\_01076, SRS\_Can\_01078, SRS\_Can\_01079, SRS\_Can\_01081,

SRS\_Can\_01082, SRS\_Can\_01086, SRS\_Can\_01090, SRS\_Can\_01091,  
SRS\_Can\_01092, SRS\_Can\_01095, SRS\_Can\_01096, SRS\_Can\_01097,  
SRS\_Can\_01098, SRS\_Can\_01099, SRS\_Can\_01100, SRS\_Can\_01101,  
SRS\_Can\_01103, SRS\_Can\_01107, SRS\_Can\_01108, SRS\_Can\_01109,  
SRS\_Can\_01110, SRS\_Can\_01111, SRS\_Can\_01112, SRS\_Can\_01114,  
SRS\_Can\_01115, SRS\_Can\_01116, SRS\_Can\_01117, SRS\_Can\_01121,  
SRS\_Can\_01122, SRS\_Can\_01125, SRS\_Can\_01126, SRS\_Can\_01129,  
SRS\_Can\_01130, SRS\_Can\_01131, SRS\_Can\_01132, SRS\_Can\_01134,  
SRS\_Can\_01135, SRS\_Can\_01136, SRS\_Can\_01138, SRS\_Can\_01139,  
SRS\_Can\_01140, SRS\_Can\_01141, SRS\_Can\_01143, SRS\_Can\_01147,  
SRS\_Can\_01148, SRS\_Can\_01149, SRS\_Can\_01150, SRS\_Can\_01151,  
SRS\_Can\_01153, SRS\_Can\_01154, SRS\_Can\_01155, SRS\_Can\_01156,  
SRS\_Can\_01157, SRS\_Can\_01159, SRS\_Can\_01160, SRS\_Can\_01161,  
SRS\_Can\_01162, SRS\_Can\_01163, SRS\_ModeMgm\_00049,  
SRS\_ModeMgm\_09001, SRS\_ModeMgm\_09009, SRS\_ModeMgm\_09017,  
SRS\_ModeMgm\_09028, SRS\_ModeMgm\_09071, SRS\_ModeMgm\_09072,  
SRS\_ModeMgm\_09078, SRS\_ModeMgm\_09080, SRS\_ModeMgm\_09081,  
SRS\_ModeMgm\_09083, SRS\_ModeMgm\_09084, SRS\_ModeMgm\_09085,  
SRS\_ModeMgm\_09087, SRS\_ModeMgm\_09089, SRS\_ModeMgm\_09090,  
SRS\_ModeMgm\_09097, SRS\_ModeMgm\_09098, SRS\_ModeMgm\_09100,  
SRS\_ModeMgm\_09101, SRS\_ModeMgm\_09102, SRS\_ModeMgm\_09104,  
SRS\_ModeMgm\_09106, SRS\_ModeMgm\_09107, SRS\_ModeMgm\_09109,  
SRS\_ModeMgm\_09110, SRS\_ModeMgm\_09112, SRS\_ModeMgm\_09113,  
SRS\_ModeMgm\_09114, SRS\_ModeMgm\_09115, SRS\_ModeMgm\_09116,  
SRS\_ModeMgm\_09118, SRS\_ModeMgm\_09119, SRS\_ModeMgm\_09120,  
SRS\_ModeMgm\_09122, SRS\_ModeMgm\_09125, SRS\_ModeMgm\_09126,  
SRS\_ModeMgm\_09127, SRS\_ModeMgm\_09128, SRS\_ModeMgm\_09132,  
SRS\_ModeMgm\_09133, SRS\_ModeMgm\_09136, SRS\_ModeMgm\_09141,  
SRS\_ModeMgm\_09143, SRS\_ModeMgm\_09145, SRS\_ModeMgm\_09146,  
SRS\_ModeMgm\_09147, SRS\_ModeMgm\_09149, SRS\_ModeMgm\_09155,  
SRS\_ModeMgm\_09156, SRS\_ModeMgm\_09157, SRS\_ModeMgm\_09158,  
SRS\_ModeMgm\_09159, SRS\_ModeMgm\_09160, SRS\_ModeMgm\_09161,  
SRS\_ModeMgm\_09162, SRS\_ModeMgm\_09163, SRS\_ModeMgm\_09164,  
SRS\_ModeMgm\_09165, SRS\_ModeMgm\_09166, SRS\_ModeMgm\_09168,  
SRS\_ModeMgm\_09169, SRS\_ModeMgm\_09172, SRS\_ModeMgm\_09173,  
SRS\_ModeMgm\_09174, SRS\_ModeMgm\_09175, SRS\_ModeMgm\_09176,  
SRS\_ModeMgm\_09177, SRS\_ModeMgm\_09178, SRS\_ModeMgm\_09179,  
SRS\_ModeMgm\_09180, SRS\_ModeMgm\_09182, SRS\_ModeMgm\_09183,  
SRS\_ModeMgm\_09184, SRS\_ModeMgm\_09185, SRS\_ModeMgm\_09186,  
SRS\_ModeMgm\_09187, SRS\_ModeMgm\_09188, SRS\_ModeMgm\_09189,  
SRS\_ModeMgm\_09190, SRS\_ModeMgm\_09194, SRS\_ModeMgm\_09199,  
SRS\_ModeMgm\_09207, SRS\_ModeMgm\_09220, SRS\_ModeMgm\_09221,  
SRS\_ModeMgm\_09222, SRS\_ModeMgm\_09223, SRS\_ModeMgm\_09225,  
SRS\_ModeMgm\_09226, SRS\_ModeMgm\_09228, SRS\_ModeMgm\_09229,  
SRS\_ModeMgm\_09230, SRS\_ModeMgm\_09231, SRS\_ModeMgm\_09232,  
SRS\_ModeMgm\_09233, SRS\_ModeMgm\_09234, SRS\_ModeMgm\_09235,  
SRS\_ModeMgm\_09236, SRS\_ModeMgm\_09237, SRS\_ModeMgm\_09238,  
SRS\_ModeMgm\_09239, SRS\_ModeMgm\_09240, SRS\_ModeMgm\_09241,  
SRS\_ModeMgm\_09242, SRS\_ModeMgm\_09243, SRS\_ModeMgm\_09244,  
SRS\_ModeMgm\_09245, SRS\_ModeMgm\_09246, SRS\_ModeMgm\_09247,

SRS\_ModeMgm\_09248, SRS\_ModeMgm\_09249, SRS\_ModeMgm\_09250,  
SRS\_ModeMgm\_09251, SRS\_ModeMgm\_09252, SRS\_ModeMgm\_09253,  
SRS\_ModeMgm\_09254, SRS\_ModeMgm\_09255, SRS\_ModeMgm\_09256,  
SRS\_ModeMgm\_09270, SRS\_ModeMgm\_09271, SRS\_ModeMgm\_09272,  
SRS\_ModeMgm\_09274, SRS\_ModeMgm\_09275, SRS\_ModeMgm\_09276,  
SRS\_ModeMgm\_09277)