Document Title	Specification of CAN State Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	253
Document Classification	Standard

Document Version	3.2.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	3.2.0	AUTOSAR Release Management	• Introduction of random delays • Re-Request of ComMode • Add WakeupValidation to avoid race conditions • Adapt Bus Off Recovery and NM state synchronization
31.10.2013	3.1.0	AUTOSAR Release Management	• Dependency to DCM module removed • Mileading timing row removed in CanSM_MainFunction • Editorial changes • Removed chapter(s) on change documentation
11.03.2013	3.0.0	AUTOSAR Administration	• Support Pretended Networking mode handling • Changed concept to setup baudrate • Initialization Sequence between ComM and CanSM • Do not send WUF as First Message on the Bus after BusOff • CanSm_TxTimeoutExeption in case of BusOff
24.11.2011	2.2.0	AUTOSAR Administration	• Added new handling to support partial networking • Changed handling for bus deinitialisation according to AR3.x behaviour • New API and handling to change the baudrate of a CAN network • Changed handling for bus-off recovery and related production error report • Comprehensive revision of all state

Document Change History			
Date	**Version**	**Changed by**	**Change Description**
			machine diagrams and SWS-ID-items • Changed classification of production errors and development errors • Solve conflicts of SWS-ID items with the conformance test specification
21.10.2010	2.1.0	AUTOSAR Administration	• Configurable Bus-Off revovery with CAN TX confirmation instead of time based recovery • Control of PDU channel modes completely shifted from CanIf to CanSM module
30.11.2009	2.0.0	AUTOSAR Administration	• VMM/AMM Concept related changes (PDU group control shifted to BswM) • Asynchronous handling of CAN network mode transitions (consideration of CAN Transceiver and CAN controller mode notifications) • 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) • Legal disclaimer revised
23.06.2008	1.0.1	AUTOSAR Administration	Legal disclaimer revised
13.11.2007	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

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

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

Table of Contents

1 Introduction and functional overview

This specification 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.

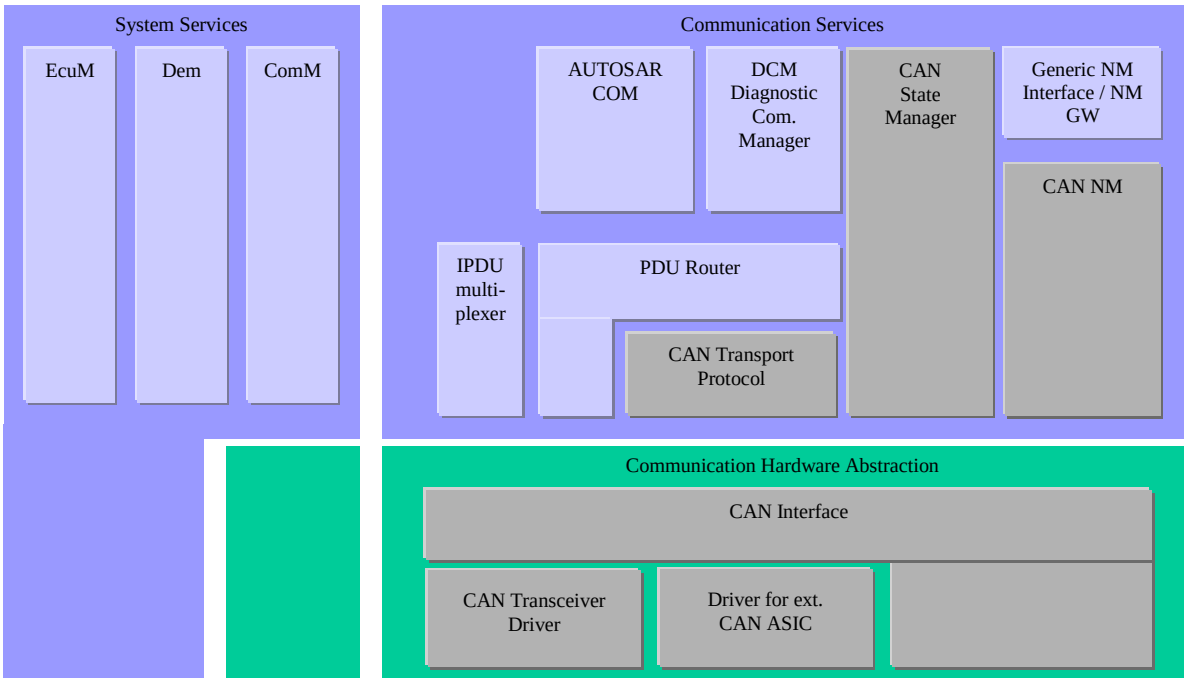

Figure 1-1: Layered Software Architecture from CanSM point of view

2 Acronyms and abbreviations

Abbreviation / Acronym:	Description:
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	Development 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_COMManager.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 Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.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 ore 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.

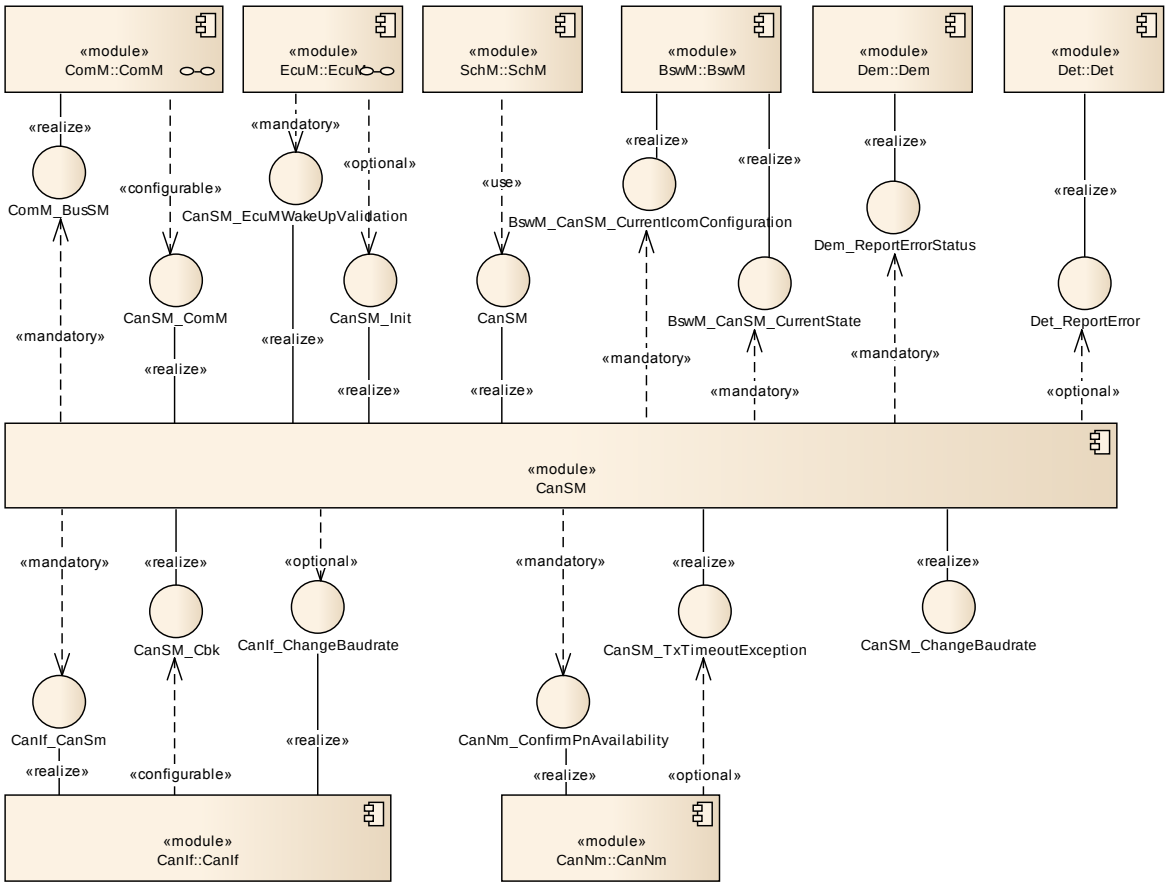

Figure 5-1: Module dependencies of the CanSM module

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 Development Error Tracer (DET)

The CanSM module reports development errors to the DET module, 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`, `CanSM_MainFunction` and `CanSM_Init`.⌋()

[SWS_CanSM_00238] ⌈The header file CanSM.h shall include the header file ComStack_Types.h.⌋()

Remark: The header file ComStack_Types.h includes the header file Std_Types.h

[SWS_CanSM_00174] ⌈The header file CanSM.h shall include the header file ComM.h.⌋()

Rationale: Some APIs of the CanSM use type definitions of the ComM module.

[SWS_CanSM_00009] ⌈The header file CanSM_ComM.h shall export the interfaces and the corresponding types, which are dedicated to the ComM module.⌋()

[SWS_CanSM_00010] ⌈The header file CanSM_Cfg.h shall contain references to the parameters of the c-source files CanSM_Lcfg.c and CanSM_PBcfg.c (see section 5.9.1 above) and shall contain pre-compile parameters, which are not declared as “const” parameter, but as defines.⌋(BSW00344, BSW0404, BSW00345, BSW00381, BSW00412)

[SWS_CanSM_00015] ⌈The CanSM module (CanSM.c) shall include the header file Det.h.⌋(BSW171)

Rationale: The functions declared in Det.h are used to report development errors.

[SWS_CanSM_00017] ⌈The CanSM module (CanSM.c) shall include the header file CanIf.h.⌋()

Rationale: The API of the CanIf module is needed for peripheral control.

[SWS_CanSM_00191] ⌈The CanSM module (CanSM.c) shall include the header file ComM_BusSM.h.⌋()

Rationale: The file ComM_BusSM.h provides the API of the ComM module, which is exclusively intended for the bus state managers.

[SWS_CanSM_00347] ⌈ The header file CanSM_BswM.h shall export the interfaces and the corresponding types, which are dedicated to the BswM module.⌋()

[SWS_CanSM_00348] ⌈The CanSM module (CanSM.c) shall include the header file CanSM_BswM.h.⌋()

[SWS_CanSM_00548]「The CanSM module (CanSM.c) shall include the header file CanNm_Cbk.h, if Partial Networking is enabled (ref. to ECUC_CanSM_00344).」()

[SWS_CanSM_00549]「The header file `CanSM_TxTimeoutException.h` shall provide the callback function `CanSM_TxTimeoutException` as optional interface to the CanNm module.」()

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
-	-	Deprecated:CANSM561
-	-	Deprecated:CANSM564
-	-	Deprecated:CANSM565
-	-	Deprecated:SWS_CanSM_00501
-	-	Deprecated:SWS_CanSM_00502
-	-	Deprecated:SWS_CanSM_00503
-	-	Deprecated:SWS_CanSM_00504
-	-	Deprecated:SWS_CanSM_00505
-	-	Deprecated:SWS_CanSM_00506
-	-	deprecated:SWS_CanSM_00524
-	-	deprecated:SWS_CanSM_00525
-	-	deprecated:SWS_CanSM_00526
-	-	deprecated:SWS_CanSM_00527
-	-	deprecated:SWS_CanSM_00529
-	-	Deprecated:SWS_CanSM_00530
-	-	deprecated:SWS_CanSM_00531
-	-	deprecated:SWS_CanSM_00532
-	-	deprecated:SWS_CanSM_00533
-	-	deprecated:SWS_CanSM_00534
-	-	deprecated:SWS_CanSM_00535
-	-	deprecated:SWS_CanSM_00536
-	-	deprecated:SWS_CanSM_00542
-	-	deprecated:SWS_CanSM_00543
-	-	Deprecated:SWS_CanSM_00562
-	-	Deprecated:SWS_CanSM_00563
-	-	Deprecated:SWS_CanSM_00566
-	-	Deprecated:SWS_CanSM_00569
-	-	Deprecated:SWS_CanSM_00571
-	-	Deprecated:SWS_CanSM_00573
-	-	Deprecated:SWS_CanSM_00574
-	-	SWS_CanSM_00008
-	-	SWS_CanSM_00009
-	-	SWS_CanSM_00017
-	-	SWS_CanSM_00167
-	-	SWS_CanSM_00174

-	-	SWS_CanSM_00182
-	-	SWS_CanSM_00183
-	-	SWS_CanSM_00184
-	-	SWS_CanSM_00186
-	-	SWS_CanSM_00187
-	-	SWS_CanSM_00188
-	-	SWS_CanSM_00189
-	-	SWS_CanSM_00190
-	-	SWS_CanSM_00191
-	-	SWS_CanSM_00235
-	-	SWS_CanSM_00238
-	-	SWS_CanSM_00250
-	-	SWS_CanSM_00251
-	-	SWS_CanSM_00252
-	-	SWS_CanSM_00266
-	-	SWS_CanSM_00278
-	-	SWS_CanSM_00282
-	-	SWS_CanSM_00284
-	-	SWS_CanSM_00347
-	-	SWS_CanSM_00348
-	-	SWS_CanSM_00360
-	-	SWS_CanSM_00369
-	-	SWS_CanSM_00370
-	-	SWS_CanSM_00371
-	-	SWS_CanSM_00372
-	-	SWS_CanSM_00374
-	-	SWS_CanSM_00375
-	-	SWS_CanSM_00376
-	-	SWS_CanSM_00377
-	-	SWS_CanSM_00395
-	-	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_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_00435
-	-	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_00463
-	-	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_00480
-	-	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_00495
-	-	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_00516
-	-	SWS_CanSM_00517
-	-	SWS_CanSM_00518
-	-	SWS_CanSM_00521
-	-	SWS_CanSM_00523
-	-	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_00536
-	-	SWS_CanSM_00538
-	-	SWS_CanSM_00539

-	-	SWS_CanSM_00540
-	-	SWS_CanSM_00541
-	-	SWS_CanSM_00542
-	-	SWS_CanSM_00543
-	-	SWS_CanSM_00546
-	-	SWS_CanSM_00548
-	-	SWS_CanSM_00549
-	-	SWS_CanSM_00550
-	-	SWS_CanSM_00554
-	-	SWS_CanSM_00555
-	-	SWS_CanSM_00556
-	-	SWS_CanSM_00557
-	-	SWS_CanSM_00558
-	-	SWS_CanSM_00560
-	-	SWS_CanSM_00561
-	-	SWS_CanSM_00569
-	-	SWS_CANSM_00575
-	-	SWS_CanSM_00576
-	-	SWS_CanSM_00577
-	-	SWS_CanSM_00578
-	-	SWS_CanSM_00579
-	-	SWS_CanSM_00580
-	-	SWS_CanSM_00581
-	-	SWS_CanSM_00582
-	-	SWS_CanSM_00583
-	-	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_00596
-	-	SWS_CanSM_00597
-	-	SWS_CanSM_00598
-	-	SWS_CanSM_00599

-	-	SWS_CanSM_00600
-	-	SWS_CanSM_00602
-	-	SWS_CanSM_00603
-	-	SWS_CanSM_00604
-	-	SWS_CanSM_00606
-	-	SWS_CanSM_00607
-	-	SWS_CanSM_00608
-	-	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_00635
-	-	SWS_CanSM_00637
BSW003	-	SWS_CanSM_00024
BSW00308	-	CANSM999
BSW00309	-	CANSM999
BSW00314	-	CANSM999
BSW00326	-	CANSM999
BSW00333	-	SWS_CanSM_00064
BSW00336	-	CANSM999
BSW00341	-	CANSM999

BSW00344	-	SWS_CanSM_00010
BSW00345	-	SWS_CanSM_00010
BSW00347	-	CANSM999
BSW00353	-	CANSM999
BSW00358	-	SWS_CanSM_00023
BSW00359	-	SWS_CanSM_00064
BSW00360	-	CANSM999
BSW00361	-	CANSM999
BSW00375	-	CANSM999
BSW00376	-	SWS_CanSM_00065
BSW00377	-	CANSM999
BSW00381	-	SWS_CanSM_00010
BSW00395	-	CANSM999
BSW00404	-	SWS_CanSM_00023
BSW00405	-	SWS_CanSM_00023
BSW00406	-	SWS_CanSM_00023, SWS_CanSM_00179
BSW00407	-	SWS_CanSM_00024
BSW00412	-	SWS_CanSM_00010
BSW00414	-	SWS_CanSM_00023
BSW00416	-	CANSM999
BSW00417	-	CANSM999
BSW00422	-	SWS_CanSM_00498, SWS_CanSM_00522, SWS_CanSM_00605
BSW00423	-	CANSM999
BSW00425	-	SWS_CanSM_00065
BSW00426	-	CANSM999
BSW00427	-	CANSM999
BSW00428	-	CANSM999
BSW00429	-	CANSM999
BSW00431	-	CANSM999
BSW00432	-	CANSM999
BSW00433	-	CANSM999
BSW00434	-	CANSM999
BSW00435	-	CANSM999
BSW00437	-	CANSM999
BSW00439	-	CANSM999
BSW00440	-	CANSM999
BSW005	-	CANSM999
BSW01142	-	SWS_CanSM_00062, SWS_CanSM_00063
BSW01144	-	SWS_CanSM_00424

BSW01146	-	SWS_CanSM_00064
BSW0404	-	SWS_CanSM_00010
BSW0405	-	SWS_CanSM_00023
BSW0424	-	SWS_CanSM_00065
BSW09080	-	SWS_CanSM_00062, SWS_CanSM_00063
BSW09081	-	SWS_CanSM_00062
BSW09083	-	SWS_CanSM_00062
BSW09084	-	SWS_CanSM_00063
BSW101	-	SWS_CanSM_00023
BSW161	-	CANSM999
BSW162	-	CANSM999
BSW168	-	CANSM999
BSW170	-	CANSM999
BSW171	-	SWS_CanSM_00015
SRS_Can_01145	The CAN State Manager shall control the assigned CAN Devices	SWS_CanSM_00609, SWS_CanSM_00610

According to [3] (General BSW Requirements):

Requirement	***Satisfied by***
[BSW00344] Reference to link-time configuration	Chapter 5.9, SWS_CanSM_00010
[BSW0404] Reference to post build time configuration	Chapter 5.9, SWS_CanSM_00010
[BSW0405] Reference to multiple configuration sets	SWS_CanSM_00023, chapter 8.2.1
[BSW00345] Pre-compile time configuration	Chapter 5.9, SWS_CanSM_00010, ECUC_CanSM_00123, ECUC_CanSM_00126, ECUC_CanSM_00127
[BSW159] Tool based configuration	Changed to not applicable during SW improvement (CANSM155 deleted)
[BSW167] Static configuration checking	SWS_CanSM_00025
[BSW171] Configurability of optional functionality	SWS_CanSM_00015, ECUC_CanSM_00133
[BSW170] Data for reconfiguration of SW-components	Not applicable (requirement on SWC-module)
[BSW00380] Separate C-Files for configuration parameters	Chapter 5.9
[BSW00419] Separate C-Files for pre-compile time configuration parameters	Chapter 5.9
[BSW00381] Separate configuration header file for pre-compile time parameters	SWS_CanSM_00010
[BSW00412] Separate configuration	SWS_CanSM_00010

header file for configuration parameters	
[BSW00383] List dependencies of configuration files	ECUC_CanSM_00161, ECUC_CanSM_00137, ECUC_CanSM_00141
[BSW00384] List dependencies to other modules	Chapter 5
[BSW00387] Specify the configuration class of callback function	Chapter 0
[BSW00388] Introduce containers	ECUC_CanSM_00123, ECUC_CanSM_00126, ECUC_CanSM_00127
[BSW00389] Containers shall have names	Chapter 10.2
[BSW00390] Parameter content shall be unique within the module	Chapter 10.2
[BSW00391] Parameter shall have unique names	Chapter 10.2
[BSW00392] Parameters shall have a type	Chapter 10.2
[BSW00393] Parameters shall have a range	Chapter 10.2
[BSW00394] Specify the scope of the parameters	Chapter 10.2
[BSW00395] List the required parameters (per parameter)	Not applicable
[BSW00396] Configuration classes	Chapter 10.2
[BSW00397] Pre–compile–time parameters	Chapter 10.2
[BSW00398] Link–time parameters	Chapter 10.2
[BSW00399] Loadable Post–build time parameters	Chapter 10.2
[BSW00400] Selectable Post–build time parameters	Chapter 10.2.1
[BSW00438] Post Build Configuration Data Structure	chapter (TODO)
[BSW00402] Published information	Chapter 10.3
[BSW00375] Notification of wake-up reason	Not applicable (no wake up interrupt)
[BSW101] Initialization interface	SWS_CanSM_00023
[BSW00416] Sequence of Initialization	Not applicable (CanSM module cannot influence the sequence for initialization)
[BSW00406] Check module initialization	SWS_CanSM_00023 SWS_CanSM_00179
[BSW00437] NoInit–Area in RAM	Not applicable (not in scope of this spec)
[BSW168] Diagnostic interface	Not applicable (requirement on SWC-module)
[BSW00407] Function to read out published parameters	SWS_CanSM_00024

[BSW00423] Usage of SW–C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (not in scope of this spec)
[BSW00424] BSW main processing function task allocation	SWS_CanSM_00065
[BSW00425] Trigger conditions for schedulable objects	SWS_CanSM_00065
[BSW00426] Exclusive areas in BSW modules	Not applicable (not in scope of this spec)
[BSW00427] ISR description for BSW modules	Not applicable (not in scope of this spec)
[BSW00428] Execution order dependencies of main processing functions	Not applicable (not in scope of this spec)
[BSW00429] Restricted BSW OS functionality access	Not applicable (not in scope of this spec)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (not in scope of this spec)
[BSW00432] Modules should have separate main processing functions for read/receive and write/transmit data path	Not applicable (not in scope of this spec)
[BSW00433] Calling of main processing functions	Not applicable (not in scope of this spec)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (not in scope of this spec)
[BSW00336] Shutdown interface	Not applicable (no deinitialization function)
[BSW00337] Classification of errors	Chapter 7.3
[BSW00338] Detection and Reporting of development errors	Chapter 7.5, SWS_CanSM_00028
[BSW00369] Do not return development error codes via API	Chapter 7.9
[BSW00339] Reporting of production relevant errors and exceptions	SWS_CanSM_00074
[BSW00422] Pre–de–bouncing of production relevant error status	SWS_CanSM_00498, CANSM520, SWS_CanSM_00522
[BSW00417] Reporting of Error Events by Non–Basic Software	Not applicable (not in scope of this spec)
[BSW00323] API parameter checking	SWS_CanSM_00071
[BSW004] Version check	SWS_CanSM_00025
[BSW00409] Header files for production code error IDs	Chapter 7.3
[BSW00385] List possible error notifications	chapter 7.3
[BSW00386] Configuration for detecting an error	Chapter 7.5, SWS_CanSM_00071, SWS_CanSM_00028
[BSW161] Microcontroller abstraction	Not applicable (not in scope of this spec)
[BSW162] ECU layout abstraction	Not applicable

	(not in scope of this spec)
[BSW005] No hard coded horizontal interfaces within MCAL	Not applicable (not in scope of this spec)
[BSW00415] User dependent include files	Chapter 5.9.2
[BSW164] Implementation of interrupt service routines	Chapter 7.9
[BSW00325] Runtime of interrupt service routines	Chapter 7.9
[BSW00326] Transition from ISRs to OS tasks	Not applicable (not in scope of this spec)
[BSW00342] Usage of source code and object code	Chapter 10.2
[BSW00343] Specification and configuration of time	Chapter 10.2
[BSW160] Human–readable configuration data	Changed to not applicable during SW improvement (CANSM155 deleted)
[BSW007] HIS MISRA C	Chapter 7.9
[BSW00300] Module naming convention	Chapter 7.9
[BSW00413] Accessing instances of BSW modules	Chapter 7.9
[BSW00347] Naming separation of different instances of BSW drivers	Not applicable (not in scope of this spec)
[BSW00305] Self–defined data types naming convention	Chapter 8.2
[BSW00307] Global variables naming convention	Chapter 7.9
[BSW00310] API naming convention	Chapter 8.3
[BSW00373] Main processing function naming convention	Chapter 8.5.1
[BSW00327] Error values naming convention	Chapter 7.3
[BSW00335] Status values naming convention	Chapter 8.2
[BSW00350] Development error detection keyword	Chapter 7.5
[BSW00408] Configuration parameter naming convention	Chapter 10.2
[BSW00410] Compiler switches shall have defined values	Chapter 10.2
[BSW00411] Get version info keyword	Chapter 8.3.6.1 Chapter 10.2
[BSW00346] Basic set of module files	Chapter 5.9
[BSW158] Separation of configuration from implementation	Chapter 5.9
[BSW00314] Separation of interrupt frames and service routines	Not applicable (not in scope of this spec)
[BSW00370] Separation of callback interface from API	Chapter 5.9
[BSW00435] Header File Structure for the	Not applicable

Basic Software Scheduler	(not in scope of this spec)
[BSW00436] Module Header File Structure for the Basic Software Memory Mapping	SWS_CanSM_00016
[BSW00348] Standard type header	Chapter 5.9
[BSW00353] Platform specific type header	Not applicable (not in scope of this spec)
[BSW00361] Compiler specific language extension header	Not applicable (not in scope of this spec)
[BSW00301] Limit imported information	Chapter 5.9
[BSW00302] Limit exported information	Chapter 5.9
[BSW00328] Avoid duplication of code	Chapter 7.9
[BSW00312] Shared code shall be reentrant	Chapter 7.9
[BSW006] Platform independency	Chapter 7.9
[BSW00357] Standard API return type [	Chapter 8.3
[BSW00377] Module specific API return types	Not applicable (not used)
[BSW00304] AUTOSAR integer data types	Chapter 7.9
[BSW00355] Do not redefine AUTOSAR integer data types	Chapter 7.9
[BSW00378] AUTOSAR boolean type	Chapter 7.9
[BSW00306] Avoid direct use of compiler and platform specific keywords [	Chapter 7.9
[BSW00308] Definition of global data	Not applicable (not used)
[BSW00309] Global data with read–only constraint	Not applicable (not used)
[BSW00371] Do not pass function pointers via API	Chapter 8.3
[BSW00358] Return type of init() functions	SWS_CanSM_00023
[BSW00414] Parameter of init function	SWS_CanSM_00023
[BSW00376] Return type and parameters of main processing functions	SWS_CanSM_00065
[BSW00359] Return type of callback functions	SWS_CanSM_00064
[BSW00360] Parameters of callback functions	Not applicable (assignment between bus-off and impacted controller id is necessary, which is transferred as parameter)
[BSW00329] Avoidance of generic interfaces	Chapter 7.9
[BSW00330] Usage of macros / inline functions instead of functions	Chapter 7.9
[BSW00331] Separation of error and status values	Chapter 7.3, Chapter 8.2,
[BSW009] Module User Documentation	Chapter 7.9
[BSW00401] Documentation of multiple	Chapter 10.2

instances of configuration parameters	
[BSW172] Compatibility and documentation of scheduling strategy	Chapter 7.9
[BSW010] Memory resource documentation	Chapter 7.9
[BSW00333] Documentation of callback function context	SWS_CanSM_00064
[BSW00374] Module vendor identification	CANSM125
[BSW00379] Module identification	CANSM125
[BSW003] Version identification	CANSM125, SWS_CanSM_00024
[BSW00318] Format of module version numbers	CANSM125
[BSW00321] Enumeration of module version numbers	Chapter 7.9
[BSW00341] Microcontroller compatibility documentation	Not applicable (not in scope of this spec)
[BSW00334] Provision of XML file	Chapter 7.9
[BSW00439] Declaration of interrupt handlers and ISRs	Not applicable (CanSM not part of MCAL)
[BSW00405] Reference to multiple configuration sets	SWS_CanSM_00023
[BSW00440] Function prototype for callback functions of AUTOSAR Services	Not applicable (not in scope of this spec)
[BSW00441] Enumeration literals and #define naming convention	Chapter of CanSM_StateType
[BSW00404] Reference to post build time configuration	SWS_CanSM_00023

The CAN SRS ([7]) specifies the CAN specific parent requirements for the CanSM, which are listed in the following table:

Requirement	***Satisfied by***
[BSW01014] Network configuration abstraction	ECUC_CanSM_00126
BSW01142] Control flow abstraction of CAN networks	SWS_CanSM_00062, SWS_CanSM_00063, SWS_CanSM_00635 chapter 7.2
[BSW01143] BusOff recovery time	ECUC_CanSM_00128, ECUC_CanSM_00129
[BSW01144] Power-On Initialization	SWS_CanSM_00424
[BSW01145] Management of CAN devices	chapter 7.2
[BSW01146] Bus-off recovery and error handling	Figure 7-8 SWS_CanSM_00064, ECUC_CanSM_00070, CANSM343

The CanSM provides services to the ComM. Because of that, the CanSM also has to consider some requirements of the Mode Management SRS [9], which specifies the upper level requirements for the ComM. These requirements are listed in following table:

Requirement	***Satisfied by***
[BSW09080] Physical channel independency	SWS_CanSM_00062, SWS_CanSM_00063, ECUC_CanSM_00126
[BSW09081] API for requesting communication	SWS_CanSM_00062
[BSW09083] Support of different communication modes	SWS_CanSM_00062
[BSW09084] API for querying the current communication mode	SWS_CanSM_00063
[BSW09085] Indication of communication mode changes	chapter 7, chapter 8.6.1

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

[SWS_CanSM_00266] 「The CanSM module shall store the latest notified current network mode with `ComM_BusSM_ModeIndication` (chapter 8.6.1) for each configured CAN network internally (ref. to ECUC_CanSM_00126).」()

[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`.」()

[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).」()

[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).」()

[CANSM385] 「If the CanSM module state machine was triggered with `T_REPEAT_MAX` (ref. to SWS_CanSM_00463, SWS_CanSM_00480,

SWS_CanSM_00495, SWS_CanSM_00523, SWS_CanSM_00536), the CanSM module shall call the function `Det_ReportError` with the ErrorId parameter `CANSM_E_MODE_REQUEST_TIMEOUT` (ref. to chapter 7.3).」

[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 8.6.1) with the related CAN network as `channel` to confirm the PN availability to the CanNm module.」()

[SWS_CanSM_00375] 「The CanSM module shall deny any network mode request, if the time since the last detected bus-off is lower than `CanSMBorTimeL1` (ref. to ECUC_CanSM_00128) and the bus-off counter is lower than `CanSMBorCounterL1ToL2` (ref. to ECUC_CanSM_00131).」()

Rationale: Block communication mode requests during bus-off recovery

[SWS_CanSM_00376] 「The CanSM module shall deny any network mode request, if the time since the last detected bus-off is lower than `CanSMBorTimeL2` and the bus-off counter is greater or equal than `CanSMBorCounterL1ToL2` (ref. to ECUC_CanSM_00131).」()

Rationale: Block communication mode requests during bus-off recovery

[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).」()

[**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).」()

7.2 State machine for each CAN network

The following diagram specifies the behavioral state machine of the CanSM module, which shall be implemented for each configured CAN network (ref. to ECUC_CanSM_00126).

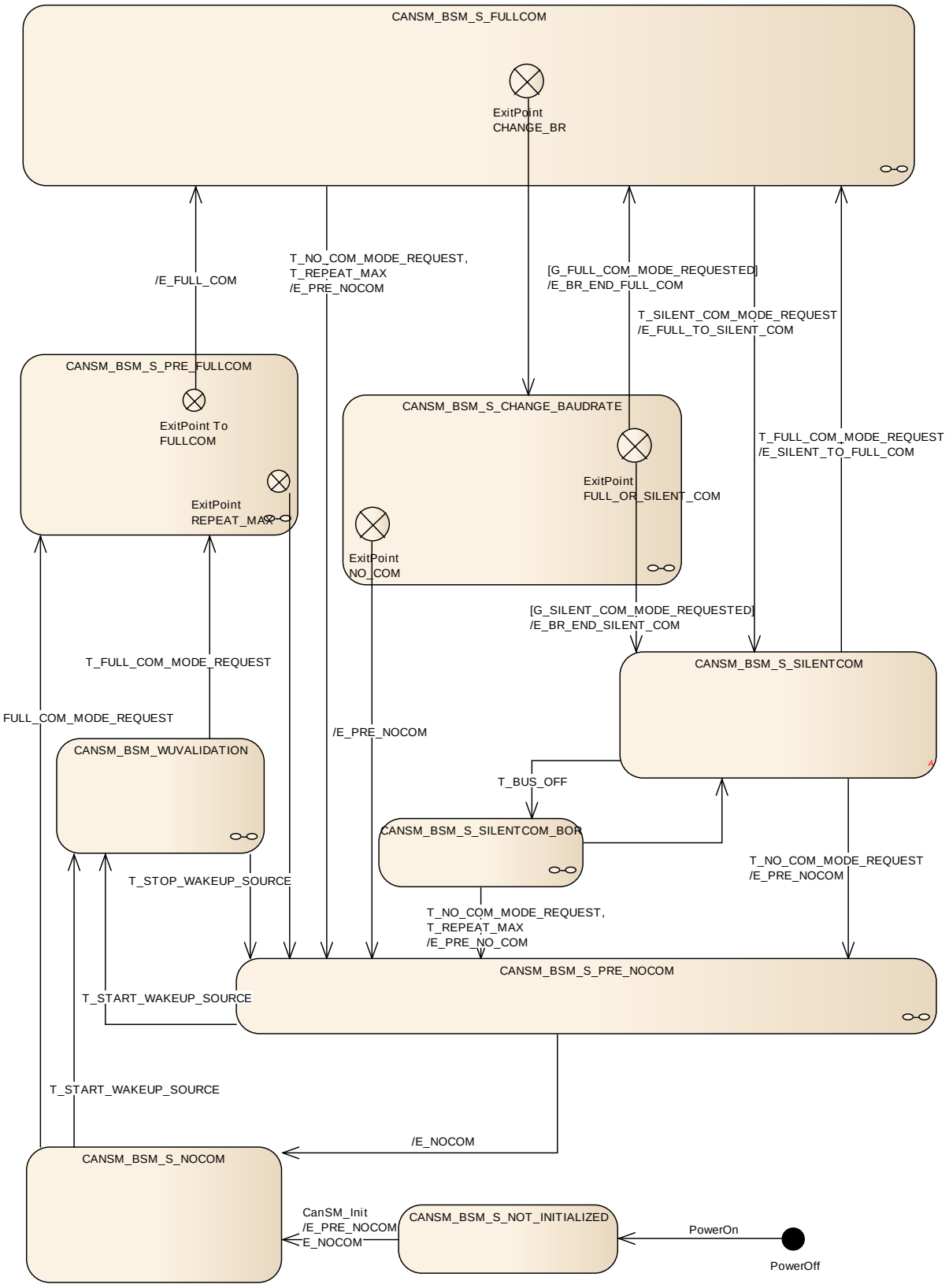

Figure 7-1: CANSM_BSM, state machine diagram for one CAN network

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`.⌋(BSW01144)

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`.⌋()

7.2.3 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`.⌋()

7.2.4 Trigger: T_STOP_WAKEUP_SOURCE

[**SWS_CanSM_00608**]⌈ If the API request `CanSM_StopWakeUpSource` (ref. to SWS_CanSM_00610) returns `E_OK` (ref. toSWS_CanSM_00622), it shall trigger the state machine (ref. to Figure 7-1) with `T_STOP_WAKEUP_SOURCE`.⌋()

7.2.5 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).⌋()

7.2.6 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).」()

7.2.7 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.」()

7.2.8 Trigger: T_REPEAT_MAX

[SWS_CanSM_00523] 「If the state machine CANSM_BSM (ref. to Figure 7-1) has repeated in one of it's sub state machines the CanIf API to start the CAN controller(s) of the CAN network (e. g. : ref. to SWS_CanSM_00509) more often than configured (ref. to ECUC_CanSM_00335) without getting the return value E_OK and without getting the supposed mode indication (e. g. : ref. to SWS_CanSM_00511), this shall trigger the state machine CANSM_BSM with T_REPEAT_MAX.」()

7.2.9 Guarding condition: G_FULL_COM_MODE_REQUESTED

[SWS_CanSM_00427] 「The guarding condition G_FULL_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_FULL_COMMUNICATION.」()

7.2.10 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.」()

7.2.11 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.`⌋()

7.2.12 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` and shall call the API `ComM_BusSM_ModeIndication` with the parameters `Channel := CanSMComMNetworkHandleRef` (ref. to ECUC_CanSM_00161) and `ComMode := COMM_NO_COMMUNICATION.`⌋()

7.2.13 Effect: E_FULL_COM

[SWS_CanSM_00435] ⌈The effect `E_FULL_COM` of the CanSM_BSM state machine (ref. to Figure 7-1) shall call at 1st place for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANSM_BSWM_FULL_COMMUNICATION.`⌋()

[SWS_CanSM_00539] ⌈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), then the effect `E_FULL_COM` of the CanSM_BSM state machine (ref. to Figure 7-1) shall call at 2nd 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.`⌋()

[SWS_CanSM_00540] ⌈The effect `E_FULL_COM` of the CanSM_BSM state machine (ref. to Figure 7-1) shall call at 3rd place for the corresponding CAN network the API `ComM_BusSM_ModeIndication` with the parameters `Channel := CanSMComMNetworkHandleRef` (ref. to ECUC_CanSM_00161) and `ComMode := COMM_FULL_COMMUNICATION.`⌋()

7.2.14 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 1st place for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANSM_BSWM_SILENT_COMMUNICATION.`⌋()

[SWS_CanSM_00541] ⌈The effect `E_FULL_TO_SILENT_COM` of the CanSM_BSM state machine (ref. to Figure 7-1) shall call at 2nd 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`⌋()

[SWS_CanSM_00538] ⌈The effect `E_FULL_TO_SILENT_COM` of the CanSM_BSM state machine (ref. to Figure 7-1) shall call at 4th 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.`⌋()

7.2.15 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_FULLCOM` (ref. to SWS_CanSM_00435).⌋()

7.2.16 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 SWS_CanSM_00434).⌋()

7.2.17 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_FULLCOM` (ref. to SWS_CanSM_00435).⌋()

7.2.18 Sub state machine CANSM_BSM_WUVALIDATION

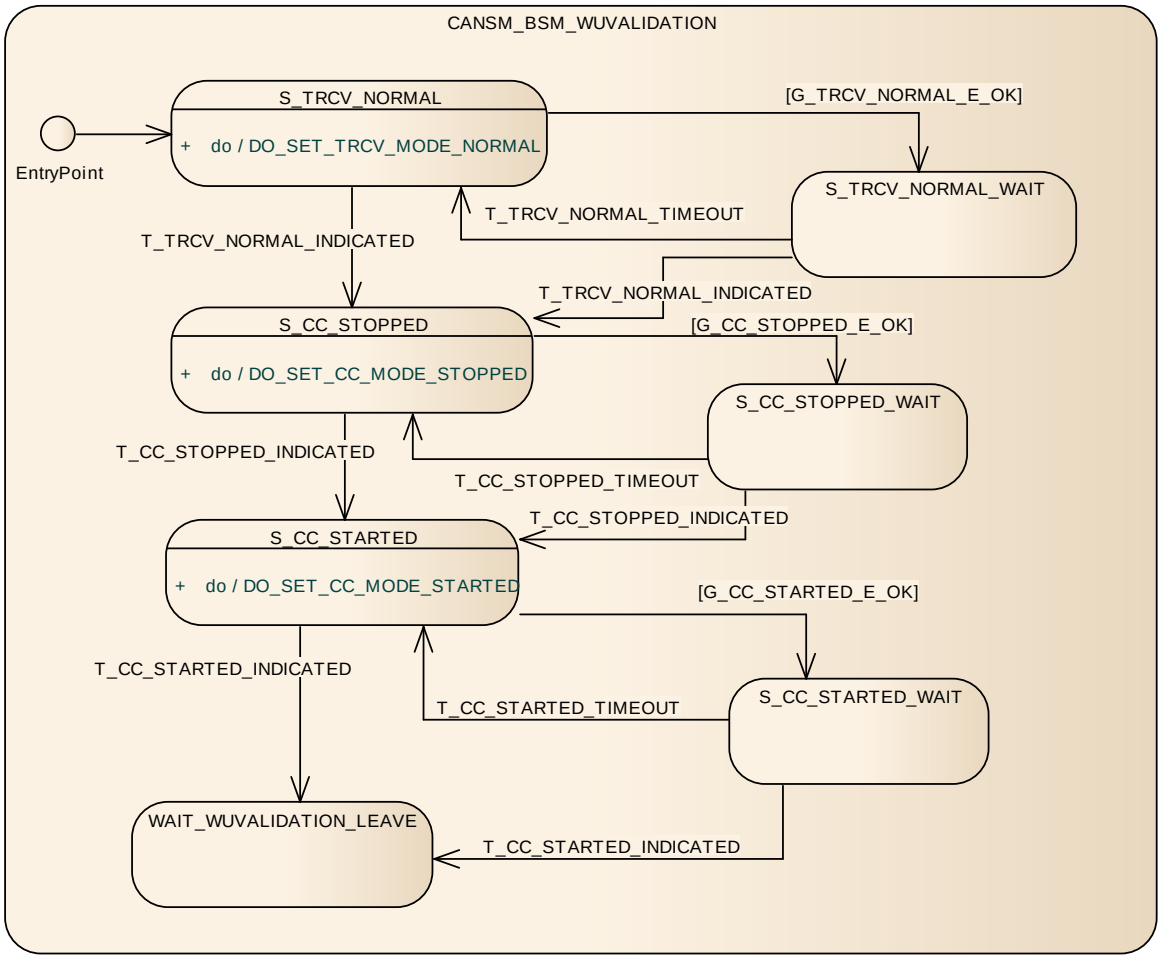

Figure 7-2: CANSM_BSM_WUVALIDATION, sub state machine of CANSM_BSM

7.2.18.1 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 8.6.1) with `TransceiverMode` equal to `CANTRCV_TRCVMODE_NORMAL`.⌋()

7.2.18.2 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`.⌋()

7.2.18.3 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.⌋()

7.2.18.4 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.⌋()

7.2.18.5 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 8.6.1) with ControllerMode equal to CANIF_CS_STOPPED.⌋()

7.2.18.6 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.⌋()

7.2.18.7 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.⌋()

7.2.18.8 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. toSWS_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.⌋()

7.2.18.9 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 8.6.1) with ControllerMode equal to CANIF_CS_STARTED.⌋()

7.2.18.10 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.⌋()

7.2.18.11 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.⌋()

7.2.18.12 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. toSWS_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.⌋()

7.2.19 Sub state machine: CANSM_BSM_S_PRE_NOCOM

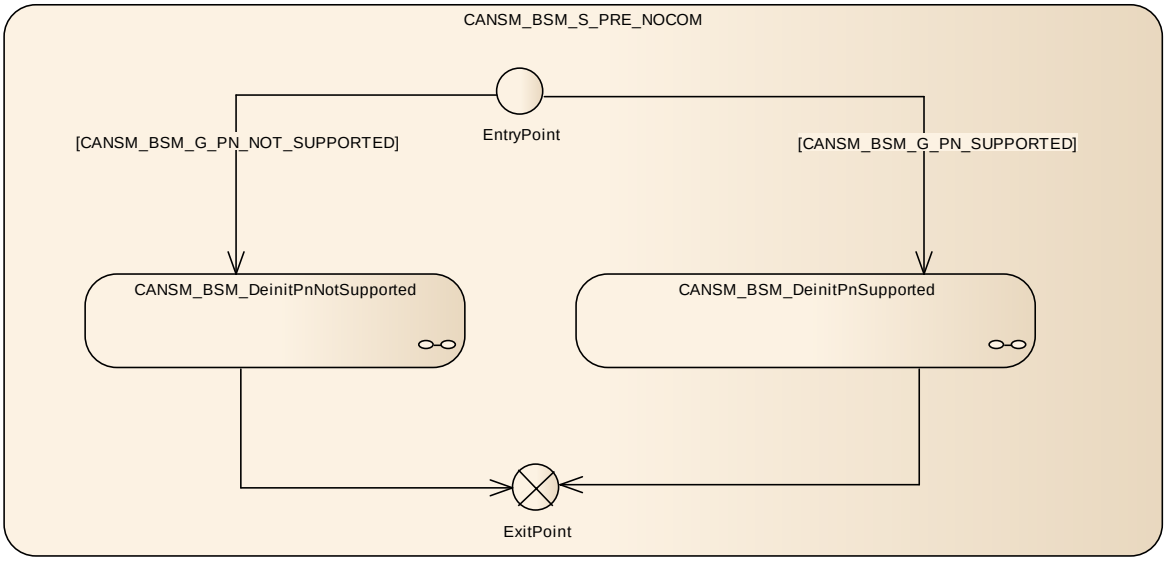

Figure 7-3: CANSM_BSM_S_PRE_NOCOM, sub state machine of CANSM_BSM

7.2.19.1 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.⌋()

7.2.19.2 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).⌋()

7.2.19.3 Sub state machine: CANSM_BSM_DeInitPnSupported

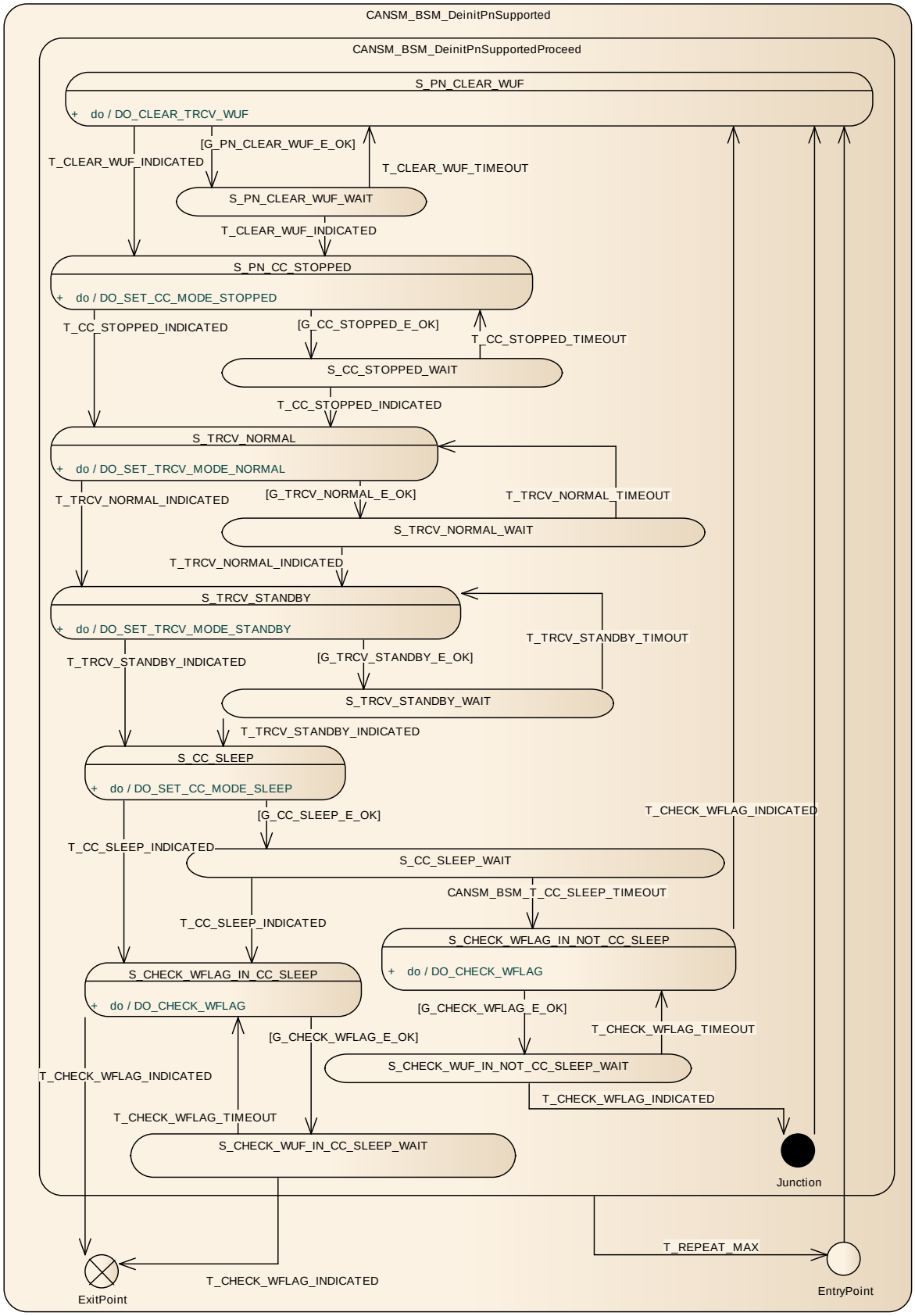

Figure 7-4: CANSM_BSM_DeinitPnSupported, sub state machine of CANSM_BSM_S_PRE_NOCOM

7.2.19.3.1 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 8.6.1) and use the configured Transceiver (ref. to ECUC_CanSM_00137) as API function parameter.⌋()

7.2.19.3.2 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`.⌋()

7.2.19.3.3 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_ClearTrcvWufFlagIndication` matches to the configured CAN Transceiver (ref. to ECUC_CanSM_00137) of the CAN network.⌋()

7.2.19.3.4 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.`⌋()

7.2.19.3.5 State operation to do in: S_PN_CC_STOPPED

[SWS_CanSM_00441] ⌈As long the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to Figure 7-4) is in the state `S_PN_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 8.6.1) with `ControllerMode` equal to `CANIF_CS_STOPPED`.⌋()

7.2.19.3.6 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`.⌋()

7.2.19.3.7 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`.⌋()

7.2.19.3.8 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`.⌋()

7.2.19.3.9 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 8.6.1) with `TransceiverMode` equal to `CANTRCV_TRCVMODE_NORMAL`.⌋()

7.2.19.3.10 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`.⌋()

7.2.19.3.11 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.」()

7.2.19.3.12 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.」()

7.2.19.3.13 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 8.6.1) with TransceiverMode equal to CANTRCV_TRCVMODE_STANDBY.」()

7.2.19.3.14 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.」()

7.2.19.3.15 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.」()

7.2.19.3.16 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.」()

7.2.19.3.17 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 8.6.1) with ControllerMode equal to CANIF_CS_SLEEP.⌋()

7.2.19.3.18 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.⌋()

7.2.19.3.19 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.⌋()

7.2.19.3.20 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.⌋()

7.2.19.3.21 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 8.6.1) and use the configured CAN Transceiver of the related Network (ref. to ECUC_CanSM_00137) as Transceiver parameter.⌋()

7.2.19.3.22 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`.⌋()

7.2.19.3.23 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.⌋()

7.2.19.3.24 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`.⌋()

7.2.19.3.25 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 8.6.1) and use the configured CAN Transceiver of the related Network (ref. to ECUC_CanSM_00137) as Transceiver parameter.⌋()

7.2.19.3.26 Trigger: T_REPEAT_MAX

[SWS_CanSM_00463] ⌈If the sub state machine `CANSM_BSM_DeinitPnSupported` (ref. to Figure 7-4) has repeated any of the CanIf API calls (ref. to SWS_CanSM_00438, SWS_CanSM_00441, SWS_CanSM_00446, SWS_CanSM_00450, SWS_CanSM_00453, SWS_CanSM_00458, SWS_CanSM_00462) more often than configured (ref. to ECUC_CanSM_00335) without getting the return value `E_OK` and without getting the supposed mode indication callbacks (ref. to SWS_CanSM_00444, SWS_CanSM_00448, SWS_CanSM_00452, SWS_CanSM_00456,

SWS_CanSM_00460), this shall trigger the sub state machine `CANSM_BSM_DeinitPnSupported` with `T_REPEAT_MAX.`⌋()

7.2.19.4 Sub state machine: CANSM_BSM_DeInitPnNotSupported

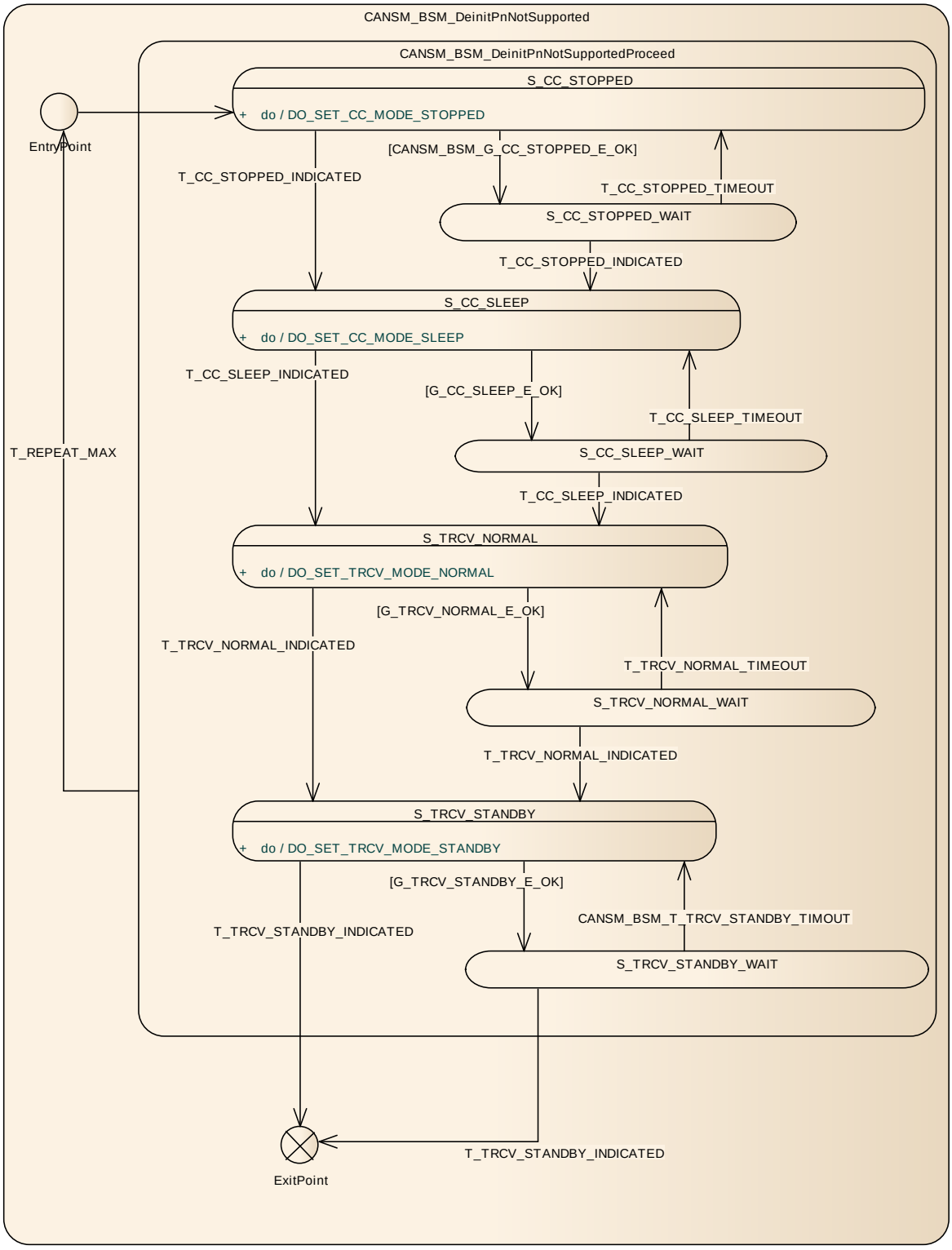

Figure 7-5: CANSM_BSM_DeinitPnNotSupported, sub state machine of CANSM_BSM_S_PRE_NOCOM

7.2.19.4.1 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 8.6.1) with ControllerMode equal to CANIF_CS_STOPPED.⌋()

7.2.19.4.2 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.⌋()

7.2.19.4.3 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.⌋()

7.2.19.4.4 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.⌋()

7.2.19.4.5 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 8.6.1) with ControllerMode equal to CANIF_CS_SLEEP.⌋()

7.2.19.4.6 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`.⌋()

7.2.19.4.7 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`.⌋()

7.2.19.4.8 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`.⌋()

7.2.19.4.9 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 8.6.1) with `TransceiverMode` equal to `CANTRCV_TRCVMODE_NORMAL`.⌋()

7.2.19.4.10 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`.⌋()

7.2.19.4.11 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.」()

[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.」()

7.2.19.4.12 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.」()

7.2.19.4.13 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 8.6.1) with TransceiverMode equal to CANTRCV_TRCVMODE_STANDBY.」()

7.2.19.4.14 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.」()

7.2.19.4.15 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.」()

[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.⌋()

7.2.19.4.16 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 CANSM_BSM_T_TRCV_STANDBY_TIMEOUT.⌋()

7.2.19.4.17 Trigger: T_REPEAT_MAX

[SWS_CanSM_00480] ⌈If the sub state machine CANSM_BSM_DeinitPnNotSupported (ref. to Figure 7-5) has repeated any of the CanIf API calls (ref. to SWS_CanSM_00464, SWS_CanSM_00468, SWS_CanSM_00472, SWS_CanSM_00476) more often than configured (ref. to ECUC_CanSM_00335) without getting the return value E_OK and without getting the supposed mode indication callbacks (ref. to SWS_CanSM_00466, SWS_CanSM_00470, SWS_CanSM_00474, SWS_CanSM_00478), this shall trigger the sub state machine CANSM_BSM_DeinitPnNotSupported with T_REPEAT_MAX.⌋()

7.2.20 Sub state machine: CANSM_BSM_S_SILENTCOM_BOR

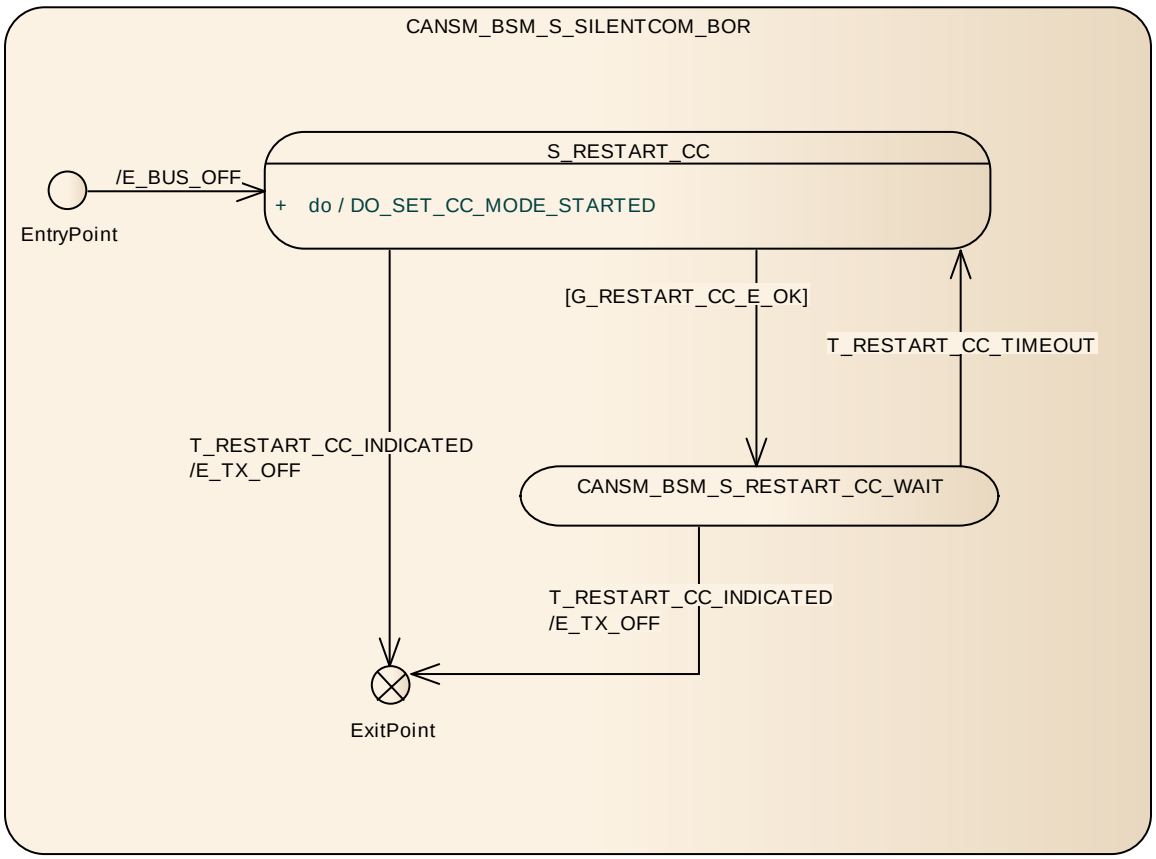

Figure 7-6: CANSM_BSM_S_SILENTCOM_BOR, sub state machine of CANSM_BSM

7.2.20.1 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 invocate `Dem_ReportErrorStatus` (ref. to chapter 8.6.1) with the parameters `EventId := CANSM_E_BUS_OFF` (ref. to ECUC_CanSM_00070) and `EventStatus := DEM_EVENT_STATUS_PRE_FAILED`.⌋(BSW00422)

7.2.20.2 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 8.6.1) with `ControllerMode` equal to `CANIF_CS_STARTED`.⌋()

7.2.20.3 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.⌋()

7.2.20.4 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.⌋()

7.2.20.5 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.⌋()

7.2.20.6 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).

7.2.21 Sub state machine: CANSM_BSM_S_PRE_FULLCOM

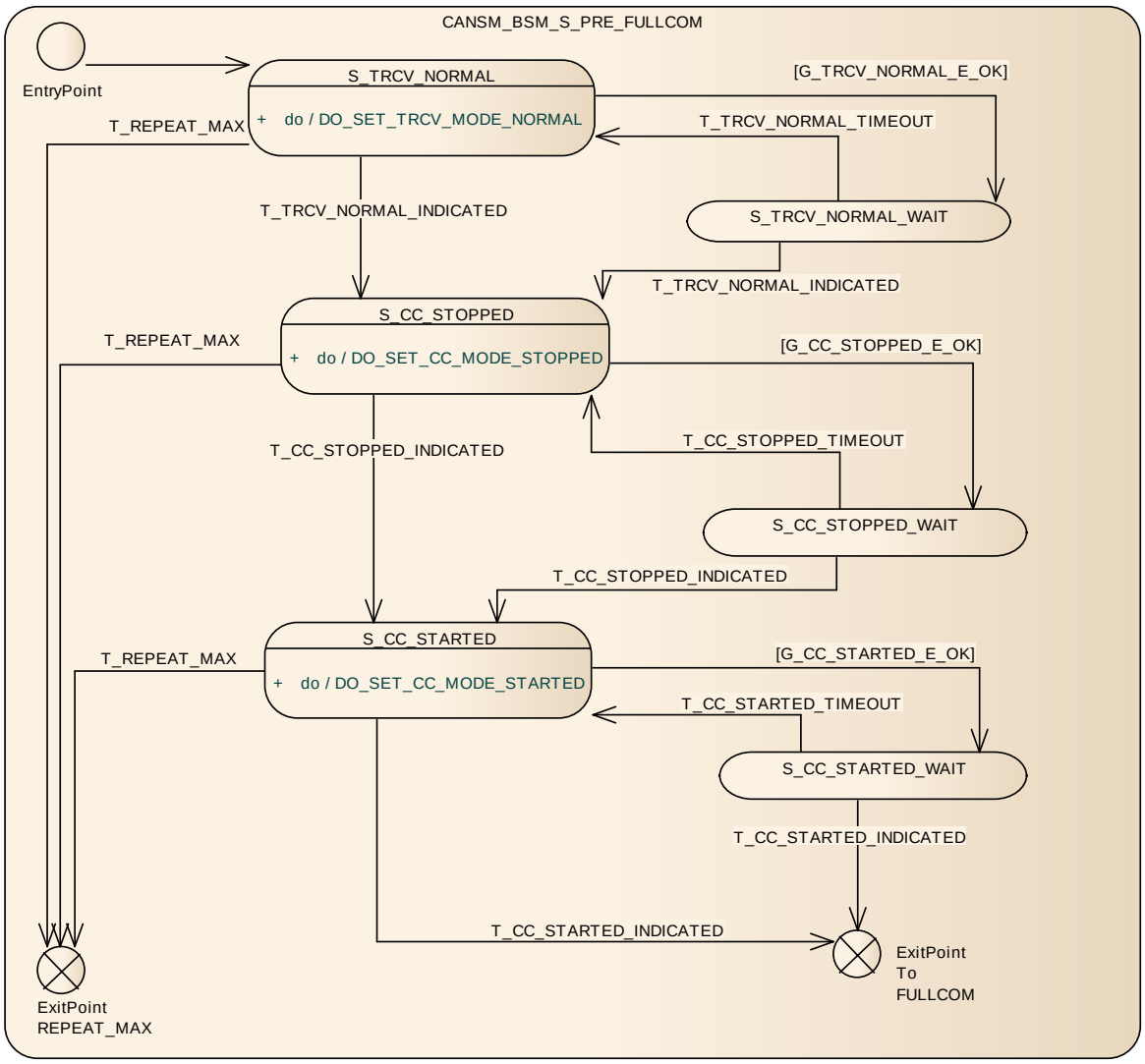

Figure 7-7: CANSM_BSM_S_PRE_FULLCOM, sub state machine of CANSM_BSM

7.2.21.1 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 8.6.1) with `TransceiverMode` equal to `CANTRCV_TRCVMODE_NORMAL`.⌋()

7.2.21.2 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.⌋()

7.2.21.3 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.⌋()

[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.⌋()

7.2.21.4 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.⌋()

7.2.21.5 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 8.6.1) with ControllerMode equal to CANIF_CS_STOPPED.⌋()

7.2.21.6 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.⌋()

7.2.21.7 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.⌋()

7.2.21.8 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.⌋()

7.2.21.9 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 8.6.1) with ControllerMode equal to CANIF_CS_STARTED.⌋()

7.2.21.10 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.⌋()

7.2.21.11 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.⌋()

7.2.21.12 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.⌋()

7.2.21.13 Trigger: T_REPEAT_MAX

[SWS_CanSM_00495] ⌈If the sub state machine CANSM_BSM_S_PRE_FULLCOM (ref. to Figure 7-7) has repeated any of the CanIf API calls (ref. to SWS_CanSM_00483, SWS_CanSM_00487, SWS_CanSM_00491) more often than configured (ref. to ECUC_CanSM_00335) without getting the return value E_OK and without getting the supposed mode indication callbacks (ref. to SWS_CanSM_00485, SWS_CanSM_00489, SWS_CanSM_00493), this shall trigger the sub state machine CANSM_BSM_S_PRE_FULLCOM with T_REPEAT_MAX.⌋()

7.2.22 Sub state machine CANSM_BSM_S_FULLCOM

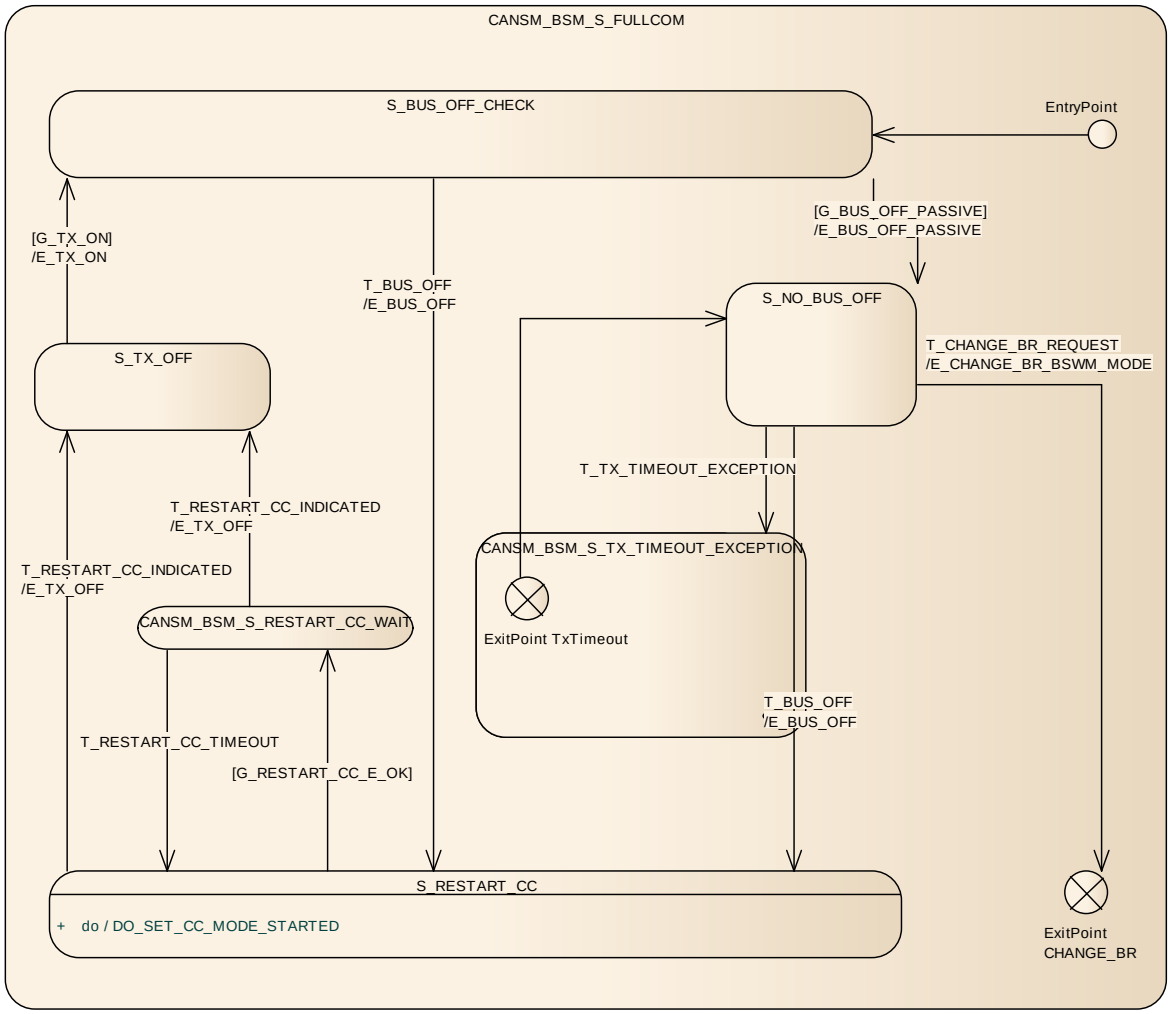

Figure 7-8: CANSM_BSM_S_FULLCOM, sub state machine of CANSM_BSM

7.2.22.1 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).⌋()

[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 8.6.1) returns `CANIF_TX_RX_NOTIFICATION` for all configured CAN controllers of the CAN network (ref. to ECUC_CanSM_00141).⌋()

7.2.22.2 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 invocate `Dem_ReportErrorStatus` (ref. to chapter 8.6.1) with the parameters `EventId := CANSM_E_BUS_OFF` (ref. to ECUC_CanSM_00070) and `EventStatus := DEM_EVENT_STATUS_PASSED`.⌋(BSW00422)

7.2.22.3 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 7-8) 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).⌋()

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

[SWS_CanSM_00554] ⌈The API request `CanSM_RequestComMode` (ref. to SWS_CanSM_00635) with the parameter `ComM_Mode` equal to `COMM_NO_COMMUNICATION` shall trigger the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) 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).⌋()

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`)".

7.2.22.4 Trigger: T_CHANGE_BR_REQUEST

[SWS_CanSM_00507] ⌈The API function `CanSM_SetBaudrate` shall trigger the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) for the requested CAN network with `T_CHANGE_BR_REQUEST`, if the CanSM module has accepted the `CanSM_SetBaudrate` (ref. to SWS_CanSM_00561) request with return of `E_OK`.⌋()

7.2.22.5 Effect: E_CHANGE_BR_BSWM_MODE

[SWS_CanSM_00528] ⌈The effect `E_CHANGE_BR_BSWM_MODE` of the sub state machine `CANSM_BSM_S_FULLCOM` (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.」()

7.2.22.6 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_FULLCOM (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.」()

7.2.22.7 Effect: E_BUS_OFF

[SWS_CanSM_00508] 「The effect E_BUS_OFF of the sub state machine CANSM_BSM_S_FULLCOM (ref. to Figure 7-8) shall call at 1st place for the corresponding CAN network the API BswM_CanSM_CurrentState with the parameters Network := CanSMComMNetworkHandleRef and CurrentState := CANSM_BSWM_BUS_OFF.」()

[SWS_CanSM_00521] 「The effect E_BUS_OFF of the sub state machine CANSM_BSM_S_FULLCOM (ref. to Figure 7-8) shall call at 2nd 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.」()

[SWS_CanSM_00522] 「The effect E_BUS_OFF of the sub state machine CANSM_BSM_S_FULLCOM (ref. to Figure 7-8) shall invocate Dem_ReportErrorStatus (ref. to chapter 8.6.1) with the parameters EventId := CANSM_E_BUS_OFF (ref. to ECUC_CanSM_00070) and EventStatus := DEM_EVENT_STATUS_PRE_FAILED.」(BSW00422)

7.2.22.8 State operation to do in: S_RESTART_CC

[SWS_CanSM_00509] 「As long the sub state machine CANSM_BSM_S_FULLCOM (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 8.6.1) with ControllerMode equal to CANIF_CS_STARTED.」()

7.2.22.9 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.⌋()

7.2.22.10 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.⌋()

7.2.22.11 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.⌋()

7.2.22.12 Effect: E_TX_OFF

The effect E_TX_OFF shall do nothing.

7.2.22.13 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), 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).⌋()

[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), 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).⌋()

[SWS_CanSM_00636] ⌈If `CanSMEnableBusOffDelay` is `TRUE`, then the guarding conditions of SWS_CANSM_00514 and SWS_CANSM_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>`.

7.2.22.14 Effect: E_TX_ON

[SWS_CanSM_00516] ⌈The effect `E_TX_ON` of the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) shall call at 1st place for the configured CAN controllers of the CAN network (ref. to ECUC_CanSM_00141) the API function `CanIf_SetPduMode` (ref. to chapter 8.6.1) with the parameters `ControllerId := CanSMControllerId` (ref. to ECUC_CanSM_00141) and `PduModeRequest := CANIF_ONLINE.`⌋()

[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 2nd place for the corresponding CAN network the API `BswM_CanSM_CurrentState` with the parameters `Network := CanSMComMNetworkHandleRef` and `CurrentState := CANSM_BSWM_FULL_COMMUNICATION.`⌋()

[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 3rd place the API `ComM_BusSM_ModeIndication` with the parameters `Channel := CanSMComMNetworkHandleRef` (ref. to ECUC_CanSM_00161) and `ComMode := COMM_FULL_COMMUNICATION.`⌋()

7.2.22.15 Trigger: T_TX_TIMEOUT_EXCEPTION

[**SWS_CanSM_00584**] ⌈The callback function `CanSM_TxTimeoutException` (ref. to SWS_CANSM_00410) shall trigger the sub state machine `CANSM_BSM_S_FULLCOM` (ref. to Figure 7-8) with `T_TX_TIMEOUT_EXCEPTION.`⌋()

7.2.22.16 Sub state machine: CANSM_BSM_S_TX_TIMEOUT_EXCEPTION

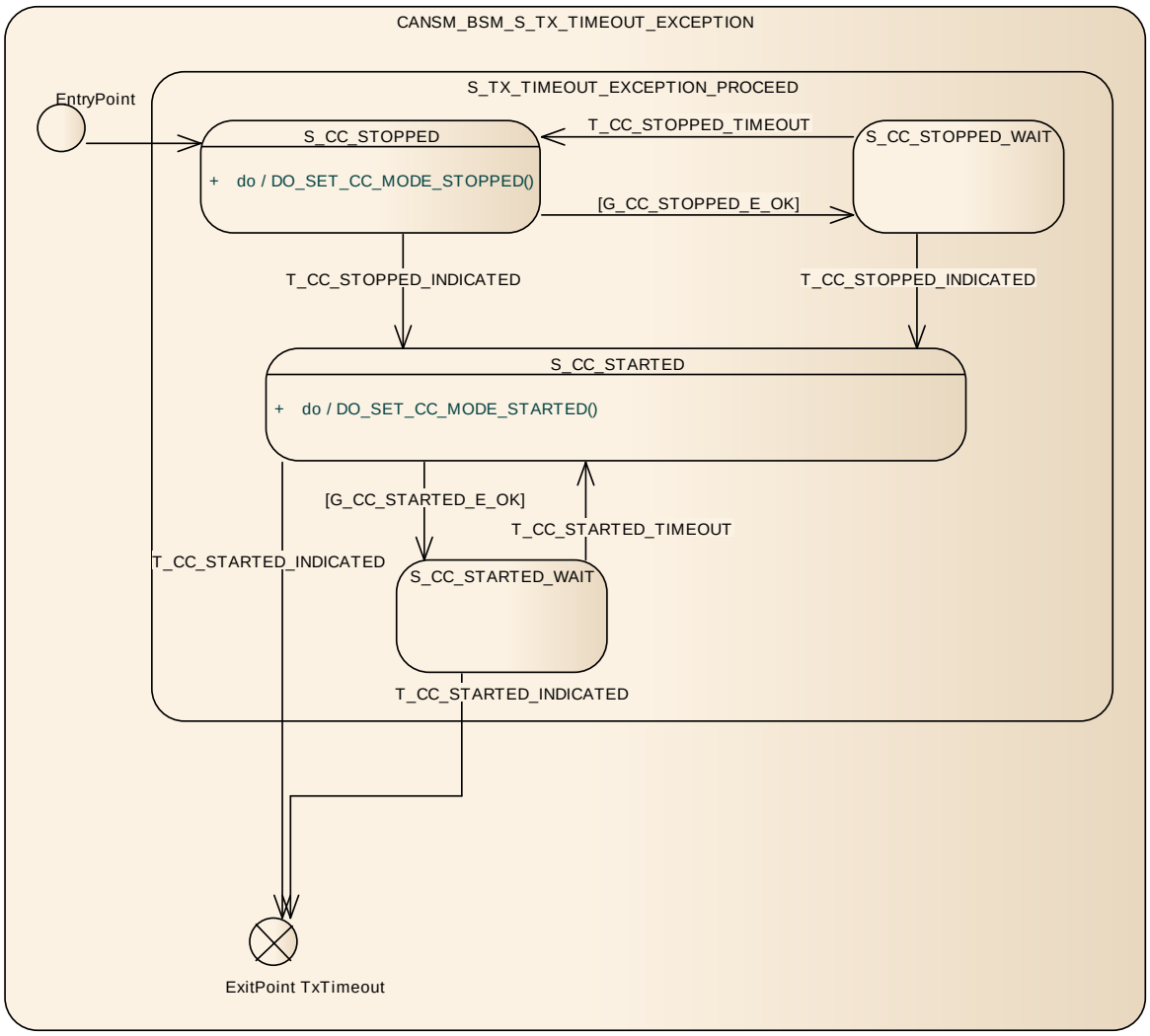

Figure 7-9: CANSM_BSM_S_TX_TIMEOUT_EXCEPTION, sub state machine of CANSM_BSM_S_FULLCOM

7.2.22.16.1 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.`⌋()

7.2.22.16.2 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.`⌋()

7.2.22.16.3 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 8.6.1) with ControllerMode equal to CANIF_CS_STOPPED.⌋()

7.2.22.16.4 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.⌋()

7.2.22.16.5 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.⌋()

7.2.22.16.6 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.⌋()⌋

7.2.22.16.7 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 8.6.1) with ControllerMode equal to CANIF_CS_STARTED.⌋()

7.2.22.16.8 Trigger: T_CC_STARTED_INDICATED

[**SWS_CanSM_00583**] ⌈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 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`.⌋()

7.2.22.16.9 Trigger: T_REPEAT_MAX

[**SWS_CANSM_00575**] ⌈ If the sub state machine `CANSM_BSM_S_TX_TIMEOUT_EXCEPTION` (ref. to Figure 7-9) has repeated the CanIf API to restart the CAN controllers(s) of the CAN network more often than configured (ref. to ECUC_CanSM_00335) without getting the supposed mode indication, this shall trigger the sub state machine `CANSM_BSM_S_TX_TIMEOUT_EXCEPTION` with `T_REPEAT_MAX`.⌋()

7.2.23 Sub state machine: CANSM_BSM_S_CHANGE_BAUDRATE

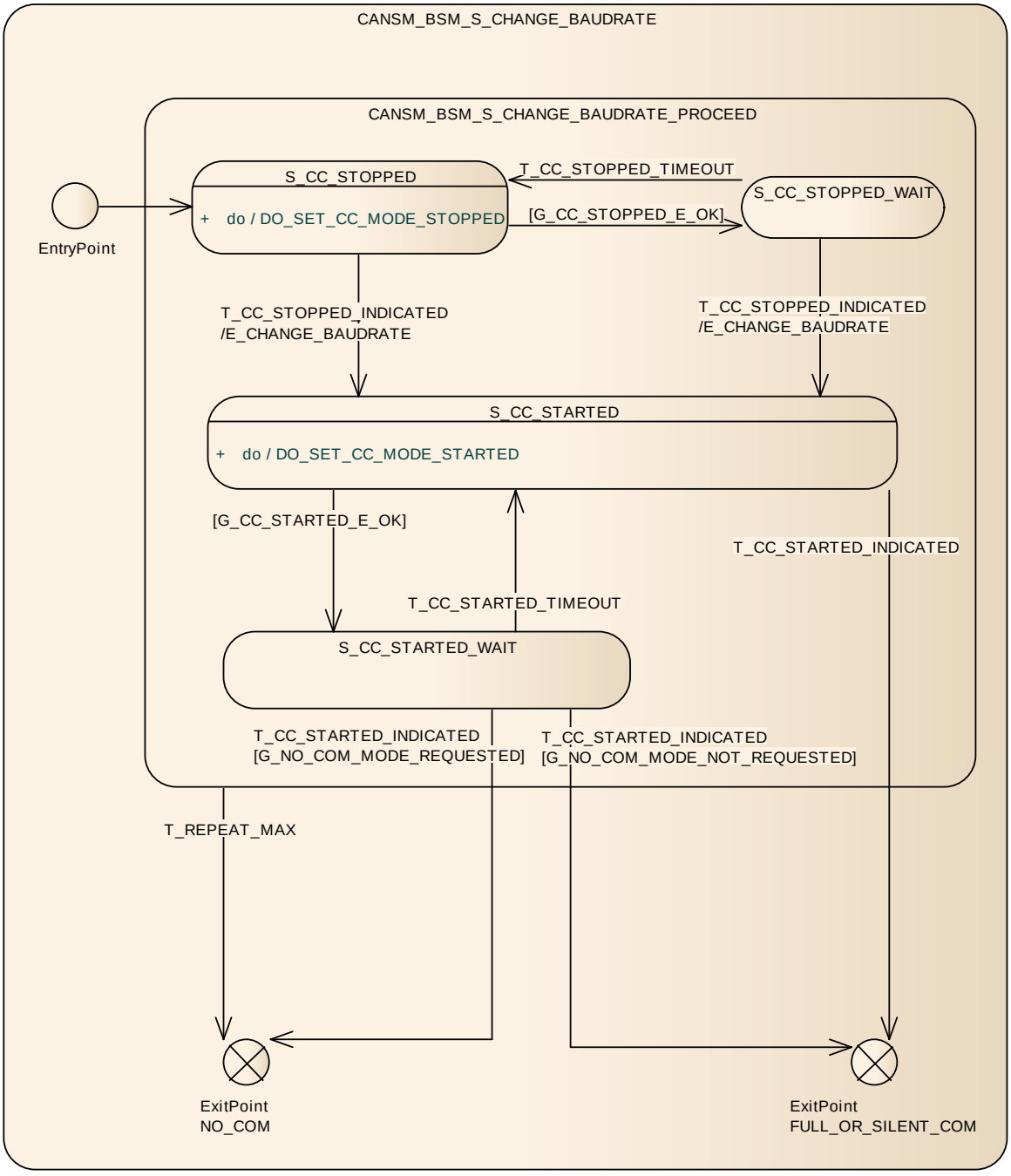

Figure 7-10: CANSM_BSM_S_CHANGE_BAUDRATE, sub state machine of CANSM_BSM

7.2.23.1 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 8.6.1) with ControllerMode equal to CANIF_CS_STOPPED.」()

7.2.23.2 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.」()

7.2.23.3 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.」()

7.2.23.4 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.」()

7.2.23.5 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 1st 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.」()

[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 2nd place for all configured CAN controllers of the CAN network (ref. to ECUC_CanSM_00141) the API request CanIf_SetBaudrate (ref. to chapter 8.6.2) with the respective ControllerId parameter and shall use as BaudRateConfigID parameter the remembered BaudRateConfigID from the call CanSM_SetBaudrate ()」()

7.2.23.6 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 8.6.1) with ControllerMode equal to CANIF_CS_STARTED.⌋()

7.2.23.7 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.⌋()

7.2.23.8 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.⌋()

7.2.23.9 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. toSWS_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.⌋()

7.2.23.10 Trigger: T_REPEAT_MAX

[SWS_CanSM_00536] ⌈If the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE (ref. to Figure 7-10) has repeated the referenced CanIf APIs (ref. to SWS_CanSM_00524, SWS_CanSM_00532) for the CAN controllers of the corresponding CAN network more often than configured (ref. to ECUC_CanSM_00335) without getting the return value E_OK and without getting the supposed mode indications (ref. to SWS_CanSM_00526, SWS_CanSM_00534), this shall trigger the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE with T_REPEAT_MAX.⌋()

7.2.23.11 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.⌋()

7.2.23.12 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.⌋()

7.2.24 Deprecated Sub state machine: CANSM_BSM_S_CHANGE_BAUDRATE

Hint
The following SWS ID items will be removed in future. They have been still kept in the document, because it is not decided completely, when they can be removed.

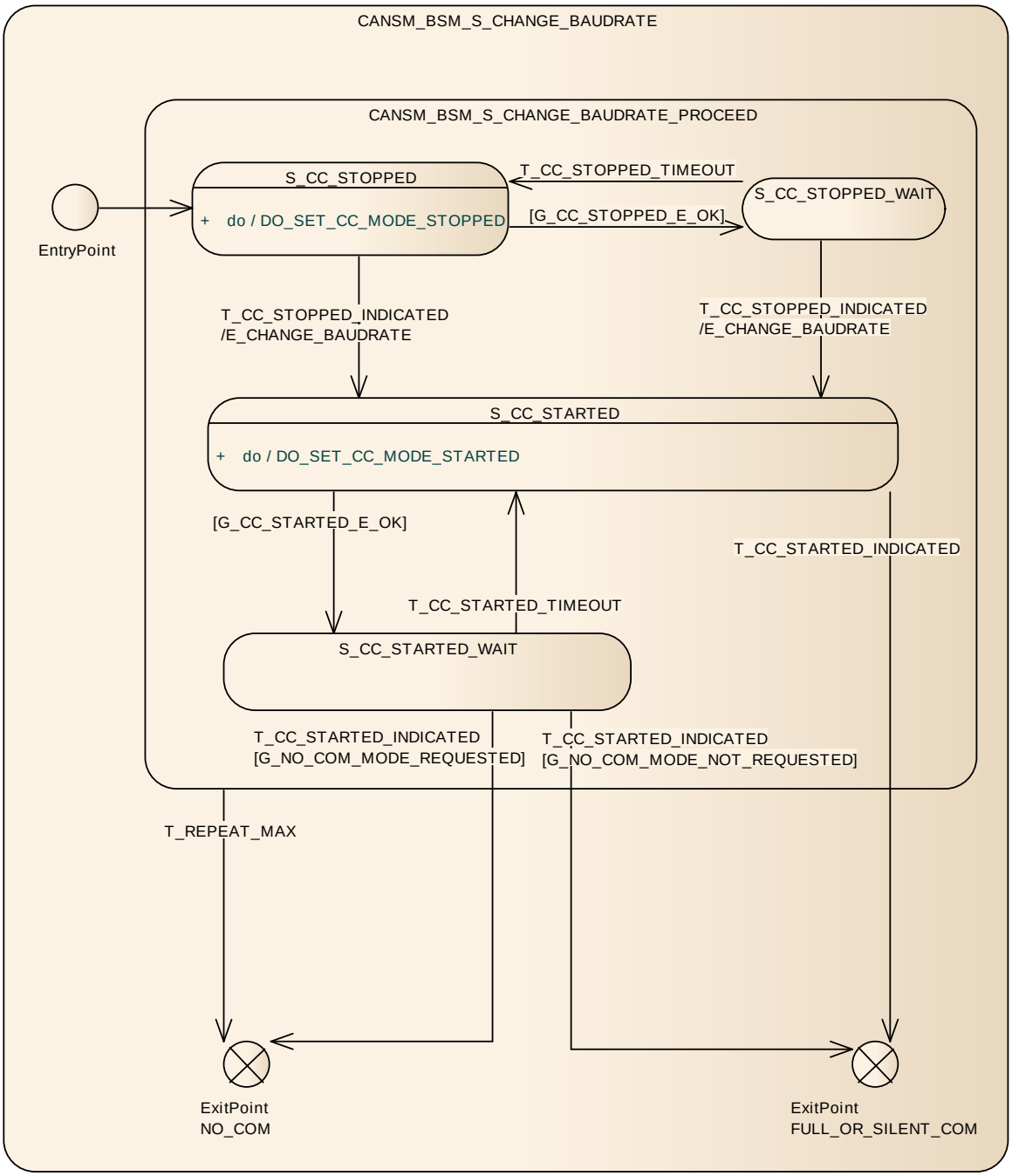

Figure 7-11: Deprecated CANSM_BSM_S_CHANGE_BAUDRATE, sub state machine of CANSM_BSM

7.2.24.1 State operation to do in: S_CC_STOPPED

[deprecated: SWS_CanSM_00524] ⌈As long the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE 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 8.6.1) with ControllerMode equal to CANIF_CS_STOPPED.⌋()

7.2.24.2 Guarding condition: G_CC_STOPPED_OK

[deprecated:SWS_CanSM_00525] ⌈The guarding condition G_CC_STOPPED_OK of the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE shall be passed, if all API calls of SWS_CanSM_00524 have returned E_OK.⌋()

7.2.24.3 Trigger: T_CC_STOPPED_INDICATED

[deprecated: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 of the CAN network with T_CC_STOPPED_INDICATED.⌋()

7.2.24.4 Trigger: T_CC_STOPPED_TIMEOUT

[deprecated: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) of the respective network with T_CC_STOPPED_TIMEOUT.⌋()

7.2.24.5 Effect: E_CHANGE_BAUDRATE

[deprecated:SWS_CanSM_00529] ⌈The effect E_CHANGE_BAUDRATE of the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE shall call at 1st 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.⌋()

[deprecated:SWS_CanSM_00531] ⌈The effect E_CHANGE_BAUDRATE of the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE shall call at 2nd place for all configured CAN controllers of the CAN network (ref. to ECUC_CanSM_00141) the API request CanIf_ChangeBaudrate (ref. to chapter 8.6.2) with the respective

ControllerId parameter and shall use as baudrate parameter the checked and remembered baud rate (ref. to SWS_CanSM_00572 and SWS_CanSM_00503).」()

7.2.24.6 State operation to do in: S_CC_STARTED

[deprecated:SWS_CanSM_00532] 「As long the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE 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 8.6.1) with ControllerMode equal to CANIF_CS_STARTED.」()

7.2.24.7 Guarding condition: G_CC_STARTED_OK

[deprecated:SWS_CanSM_00533] 「The guarding condition G_CC_STARTED_OK of the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE shall be passed, if all API calls of SWS_CanSM_00532 have returned E_OK.」()

7.2.24.8 Trigger: T_CC_STARTED_INDICATED

[deprecated: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) of the CAN network with T_CC_STARTED_INDICATED.」()

7.2.24.9 Trigger: T_CC_STARTED_TIMEOUT

[deprecated: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 of the respective network with T_CC_STARTED_TIMEOUT.」()

7.2.24.10 Trigger: T_REPEAT_MAX

[deprecated:SWS_CanSM_00536] 「If the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE has repeated the referenced CanIf APIs (ref. to SWS_CanSM_00524, SWS_CanSM_00532) for the CAN controllers of the corresponding CAN network more often than configured (ref. to ECUC_CanSM_00335) without getting the return value E_OK and without getting the supposed mode indications (ref. to SWS_CanSM_00526, SWS_CanSM_00534), this

shall trigger the sub state machine CANSM_BSM_S_CHANGE_BAUDRATE with T_REPEAT_MAX.」()

7.2.24.11 Guarding condition: G_NO_COM_MODE_REQUESTED

[deprecated:SWS_CanSM_00542] 「The sub state machine CANSM_BSM_S_CHANGE_BAUDRATE shall pass the guarding condition G_NO_COM_MODE_REQUESTED, if the latest accepted communication mode request with CanSM_RequestComMode (ref. to SWS_CanSM_00062) for the respective network handle of the state machine has been with the parameter ComM_Mode equal to COMM_NO_COMMUNICATION.」()

7.2.24.12 Guarding condition: G_NO_COM_MODE_NOT_REQUESTED

[deprecated:SWS_CanSM_00543] 「The sub state machine CANSM_BSM_S_CHANGE_BAUDRATE 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_00062) 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.」()

7.3 Production errors

7.3.1 CANSM_E_BUS_OFF

Error Name:	CANSM_E_BUS_OFF (ref. to ECUC_CanSM_00070)	
Short Description:	Bus-off detection	
Long Description:	The bus-off recovery state machine of a CAN network has detected a certain amount of sequential bus-offs without successful recovery	
Recommended DTC:	Assigned by DEM	
Detection Criteria:	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)
Secondary Parameters:	None	
Time Required:	PRE_FAILED immediately (in error interrupt context), FAILED depending on debounce configuration of DEM	
Monitor Frequency	Continuous	
MIL illumniation:	Assigned by DEM	

7.4 Error classification

This chapter lists and classifies all errors that can be detected by this software module. Each error is classified to relevance (development / production) and the related error code (unique label for the error). For development errors this table also specifies the unique values, which correspond to the error codes.

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service used without module initialization	Development	CANSM_E_UNINIT	0x01
API service called with wrong pointer	Development	CANSM_E_PARAM_POINTER	0x02
API service called with wrong parameter	Development	CANSM_E_INVALID_NETWORK_HANDLE	0x03
API service called with wrong parameter	Development	CANSM_E_PARAM_CONTROLLER	0x04
API service called with wrong parameter	Development	CANSM_E_PARAM_TRANSCEIVER	0x05
Network mode request during not finished bus-off recovery	Development	CANSM_E_BUSOFF_RECOVERY_ACTIVE	0x06
Network mode request during pending indication	Development	CANSM_E_WAIT_MODE_INDICATION	0x07
Mode request for a network failed more often as allowed by configuration	Development	CANSM_E_MODE_REQUEST_TIMEOUT	0x0A
Invalid BaudrateConfig for at least one of the CAN Controllers of the requested CAN Network (related to the deprecated APIs CanSM_CheckBaudrate and CanSM_ChangeBaudatre)	Development	CANSM_E_PARAM_INVALID_BAUDRATE	0x09

7.5 Pretended Networking function

7.5.1 Activation

[**SWS_CanSM_00588**]⌈ To activate Pretended Networking the CanSM module shall request an ICOM configuration by calling CanIf_SetIcomConfiguration.⌋()

[**SWS_CanSM_00589**]⌈ The CanSM shall inform the BswM about the activation status by calling BswM_CanSM_CurrentIcomConfiguration.⌋()

7.5.2 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.⌋()

[**SWS_CanSM_00591**]⌈ The CanSM shall inform BswM about the deactivation status by calling BswM_CanSM_CurrentIcomConfiguration.⌋()

7.6 Error detection

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

7.7 Error notification

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

7.8 Interface for AUTOSAR debug and trace

For details refer to the chapter 7.1.17 “Debugging support” in *SWS_BSWGeneral.*

7.9 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 files are listed:

[CANSM243]

Module	*Imported Type*
CanIf	CanIf_ControllerModeType
	CanIf_NotifStatusType
	CanIf_PduModeType
Can_GeneralTypes	CanTrcv_TrcvModeType
ComM	ComM_ModeType
ComStack_Types	IcomConfigIdType
	IcomSwitch_ErrorType
	NetworkHandleType
Dem	Dem_EventIdType
	Dem_EventStatusType
Std_Types	Std_ReturnType
	Std_VersionInfoType

8.2 Type definitions

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

8.2.1 CanSM_StateType

[SWS_CanSM_00596]⌈

Name:	`CanSM_StateType`	
Type:	`Enumeration`	
Range:	`CANSM_INITED`	--
	`CANSM_UNINITED`	--
Description:	Defines the values of the internal states of the CanSM module	

⌋()

8.2.2 CanSM_ConfigType

[SWS_CanSM_00597]⌈

Name:	`CanSM_ConfigType`	
Type:	`Structure`	
Range:	`--`	--
Description:	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.	

⌋()

8.2.3 CanSM_BswMCurrentStateType

[SWS_CanSM_00598]⌈

Name:	CanSM_BswMCurrentStateType	
Type:	Enumeration	
Range:	CANSM_BSWM_NO_COMMUNICATION	--
	CANSM_BSWM_SILENT_COMMUNICATION	--
	CANSM_BSWM_FULL_COMMUNICATION	--
	CANSM_BSWM_BUS_OFF	--
	CANSM_BSWM_CHANGE_BAUDRATE	--
Description:	Can specific communication modes / states notified to the BswM module	

⌋()

8.3 Function definitions

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

8.3.1 CanSM_Init

[SWS_CanSM_00023] ⌈

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

⌋(BSW0405, BSW101, BSW00406, BSW00358, BSW00414, BSW00405, BSW00404)

[SWS_CanSM_00179] ⌈Only for configuration variant 3: The function `CanSM_Init` shall report the development error `CANSM_E_PARAM_POINTER` to the DET, if the user of this function hands over a NULL-pointer as `ConfigPtr`.⌋(BSW00406)

8.3.2 CanSM_RequestComMode

[SWS_CanSM_00062] ⌈

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

⌋(BSW01142, BSW09080, BSW09081, BSW09083)

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).⌋()

[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).⌋()

[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).⌋()

[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.」()

[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).」()

[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.」()

[SWS_CanSM_00395] 「If the CanSM module has to deny the request CanSM_RequestComMode, because of a pending mode indication (ref. to CANSM388), then this function shall call the function Det_ReportError with the ErrorId parameter CANSM_E_WAIT_MODE_INDICATION (ref. to chapter 7.3).」()

8.3.3 CanSM_GetCurrentComMode

[SWS_CanSM_00063] 「

Service name:	CanSM_GetCurrentComMode	
Syntax:	`Std_ReturnType CanSM_GetCurrentComMode( NetworkHandleType network, ComM_ModeType* ComM_ModePtr )`	
Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	network	Network handle, whose current communication mode shall be put out
Parameters (inout):	None	
Parameters (out):	ComM_ModePtr	Pointer, where to put out the current communication mode
Return value:	Std_ReturnType	E_OK: Service accepted E_NOT_OK: Service denied
Description:	This service shall put out the current communication mode of a CAN network.	

」(BSW01142, BSW09080, BSW09084)

[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).」()

[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).⌋()

[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).⌋()

[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.⌋()

[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.⌋()

[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`.⌋()

[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`.⌋()

8.3.4 CanSM_StartWakeupSource

[SWS_CanSM_00609]⌈

Service name:	CanSM_StartWakeupSource	
Syntax:	`Std_ReturnType CanSM_StartWakeupSource( NetworkHandleType network )`	
Service ID[hex]:	0x11	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	network	Affected CAN network
Parameters (inout):	None	
Parameters (out):	None	

Return value:	Std_ReturnType	E_OK: Request accepted E_NOT_OK: Request denied
Description:	This function shall be called by EcuM when a wakeup source shall be started.	

⌋(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).⌋()

[**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).⌋()

[**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).⌋()

[**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).⌋()

[**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).⌋()

8.3.5 CanSM_StopWakeupSource

[**SWS_CanSM_00610**]⌈

Service name:	CanSM_StopWakeupSource
Syntax:	`Std_ReturnType CanSM_StopWakeupSource( NetworkHandleType network`

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

」(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).」()

[**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).」()

[**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).」()

[**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).」()

[**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).」()

8.3.6 Optional

8.3.6.1 CanSM_GetVersionInfo

[SWS_CanSM_00024] ⌈

Service name:	CanSM_GetVersionInfo	
Syntax:	`void CanSM_GetVersionInfo( Std_VersionInfoType* VersionInfo )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	VersionInfo	Pointer to where to store the version information of this module.
Return value:	None	
Description:	This service puts out the version information of this module (module ID, vendor ID, vendor specific version numbers related to BSW00407)	

⌋(BSW00407, BSW003)

[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.⌋()

8.3.6.2 CanSM_SetBaudrate

[SWS_CanSM_00561] ⌈

Service name:	CanSM_SetBaudrate	
Syntax:	`Std_ReturnType CanSM_SetBaudrate( NetworkHandleType Network, uint16 BaudRateConfigID )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different Networks. Non reentrant for the same Network.	
Parameters (in):	Network	Handle of the addressed CAN network for the baud rate change
	BaudRateConfigID	references a baud rate configuration by ID (see CanControllerBaudRateConfigID)
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Service request accepted, setting of (new) baud rate started E_NOT_OK: Service request not accepted
Description:	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.	

⌋()

[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`. ⌋()

[SWS_CanSM_00M570] 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`. ⌋()

[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).⌋()

[SWS_CanSM_00504] ⌈The function `CanSM_ChangeBaudrate` 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.⌋()

[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`.⌋()

[SWS_CanSM_00530] ⌈The CanSM module shall deny the `CanSM_SetBaudrate` API request, if the CanSM module is not initialized.⌋()

[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).⌋()

[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 start the asynchronous process to change the baud rate of the CAN network's CAN Controllers.⌋()

8.3.6.3 CanSM_SetIcomConfiguration

[SWS_CanSM_00586]⌈

Service name:	CanSM_SetIcomConfiguration
Syntax:	`Std_ReturnType CanSM_SetIcomConfiguration( NetworkHandleType Network,`

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

」()

[**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`.」()

[**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 7.5) and return E_OK or E_NOT_OK depending on the return value of the requested CanIf API.」()

[**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`.」()

8.3.7 Obsolete / deprecated

The following API functions are deprecated and will be removed in the future AUTOSAR releases.

8.3.7.1 CanSM_CheckBaudrate

[Deprecated: SWS_CanSM_00501] ⌈

Service name:	CanSM_CheckBaudrate	
Syntax:	`Std_ReturnType CanSM_CheckBaudrate( NetworkHandleType network, const uint16 Baudrate )`	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	network	Handle of the addressed CAN network to check if a baudrate is supported
	Baudrate	Baudrate to check in kbps
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Baudrate supported by all configured CAN controllers of the network E_NOT_OK: Baudrate not supported / invalid network
Description:	This service shall check, if a certain baudrate is supported by the configured CAN controllers of a certain CAN network. Please note that this API is deprecated and is kept only for backward compatibility reasons. In the next major release this API will be deleted.	

⌋()

[Deprecated: CANSM564]: ⌈The CanSM module shall provide the API function `CanSM_CheckBaudrate`, if the `CanSmChangeBaudrateApi` parameter (ref. to ECUC_CanSM_00342) is configured with the value `TRUE`. ⌋()

[Deprecated: CANSM565]: ⌈The CanSM module shall not provide the API function `CanSM_CheckBaudrate`, if the `CanSmChangeBaudrateApi` parameter (ref. to ECUC_CanSM_00342) is configured with the value `FALSE`. ⌋()

[Deprecated: SWS_CanSM_00562] ⌈The CanSM module shall deny the `CanSM_CheckBaudrate` API request, if the `NetworkHandle` parameter does not match to the configured Network handles of the CanSM module (ref. to ECUC_CanSM_00161). ⌋()

[Deprecated: SWS_CanSM_00571] ⌈The function `CanSM_CheckBaudrate` 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.⌋()

[Deprecated: SWS_CanSM_00563] ⌈If the `NetworkHandle` parameter in the `CanSM_CheckBaudrate` request matches to one of the configured Network handles (ref. to ECUC_CanSM_00161) and the requested baud rate is supported (ref. to SWS_CanSM_00567), then the function shall return `E_OK`. ⌋()

[**Deprecated: SWS_CanSM_00566**] ⌈If the `NetworkHandle` parameter in the `CanSM_CheckBaudrate` request matches to one of the configured Network handles (ref. to ECUC_CanSM_00161) and the requested baud rate is not supported (ref. to SWS_CanSM_00568), then the function shall return `E_NOT_OK`. ⌋()

8.3.7.2 CanSM_ChangeBaudrate

[Deprecated: CANSM561] ⌈

Service name:	CanSM_ChangeBaudrate	
Syntax:	`Std_ReturnType CanSM_ChangeBaudrate(NetworkHandleType network, const uint16 Baudrate)`	
Service ID[hex]:	0x0e	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	network	Handle of the addressed CAN network for the baudrate change
	Baudrate	Requested Baudrate in kbps
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Service request accepted E_NOT_OK: Service request not accepted
Description:	This service shall start an asynchronous process to change the baudrate for the configured CAN controllers of a certain CAN network	

⌋()

[Deprecated: SWS_CanSM_00569] ⌈The CanSM module shall provide the API function `CanSM_ChangeBaudrate`, if the `CanSmChangeBaudrateApi` parameter (ref. to ECUC_CanSM_00342) is configured with the value `TRUE`. ⌋()

[Deprecated: CANSM570] The CanSM module shall not provide the API function `CanSM_ChangeBaudrate`, if the `CanSmChangeBaudrateApi` parameter (ref. to ECUC_CanSM_00342) is configured with the value `FALSE`. ⌋()

[Deprecated: SWS_CanSM_00502] ⌈The CanSM module shall deny the `CanSM_ChangeBaudrate` API request, if the `NetworkHandle` parameter does not match to the configured Network handles of the CanSM module (ref. to ECUC_CanSM_00161).⌋()

[Deprecated: SWS_CanSM_00504] ⌈The function `CanSM_ChangeBaudrate` 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.⌋()

[Deprecated: SWS_CanSM_00505] ⌈The function `CanSM_ChangeBaudrate` shall deny its request, if the requested CAN network is not in the communication mode `COMM_FULL_COMMUNICATION`.⌋()

[Deprecated: SWS_CanSM_00530] ⌈The CanSM module shall deny the `CanSM_ChangeBaudrate` API request, if the CanSM module is not initialized.⌋()

[Deprecated: SWS_CanSM_00506] ⌈If the function `CanSM_ChangeBaudrate` 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).⌋()

[Deprecated: SWS_CanSM_00573] ⌈If the requested baud rate is not equal to the remembered baud rate of the last `CanSM_CheckBaudrate` call (ref. to SWS_CanSM_00572) for the corresponding CAN network or if the remembered result of the last `CanSM_CheckBaudrate` call for the corresponding CAN network has been `E_NOT_OK`, then the `CanSM_ChangeBaudrate` call shall return `E_NOT_OK`. ⌋()

[Deprecated: SWS_CanSM_00574] ⌈If the requested baud rate is not equal to the remembered baud rate of the last `CanSM_CheckBaudrate` call (ref. to SWS_CanSM_00572) for the corresponding CAN network or if the remembered result of the last `CanSM_CheckBaudrate` call for the corresponding CAN network has been `E_NOT_OK`, then the `CanSM_ChangeBaudrate` call the function `Det_ReportError` with `ErrorId` parameter `CANSM_E_PARAM_INVALID_BAUDRATE` (ref. to chapter 7.3). ⌋()

[Deprecated: SWS_CanSM_00503] ⌈If no condition is present to deny the `CanSM_ChangeBaudrate` request according to SWS_CanSM_00502, SWS_CanSM_00505, SWS_CanSM_00530 and SWS_CanSM_00573, then the CanSM module shall return `E_OK` and start the asynchronous process to change the

baud rate of the CAN network's CAN Controllers to the checked and requested baud rate (ref. to SWS_CanSM_00507).」()

8.4 Call-back notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the file `CanSM_Cbk.h`

8.4.1 CanSM_ControllerBusOff

[SWS_CanSM_00064] 「

Service name:	CanSM_ControllerBusOff	
Syntax:	`void CanSM_ControllerBusOff( uint8 ControllerId )`	
Service ID[hex]:	0x04	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (only for different CanControllers)	
Parameters (in):	ControllerId	CAN controller, which detected a bus-off event
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	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.	

」(BSW00359, BSW00333, BSW01146)

[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`.」()

[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`.」()

[SWS_CanSM_00377] 「If the CanSM module has to deny the request `CanSM_RequestComMode`, because of a not finished bus-off recovery (ref. to SWS_CanSM_00375 and SWS_CanSM_00376), then this function shall call the function `Det_ReportError` with the `ErrorId` parameter `CANSM_E_BUSOFF_RECOVERY_ACTIVE` (ref. to chapter 7.3).」()

[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).⌋()

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.

8.4.2 CanSM_ControllerModeIndication

[SWS_CanSM_00396] ⌈

Service name:	CanSM_ControllerModeIndication	
Syntax:	`void CanSM_ControllerModeIndication( uint8 ControllerId, CanIf_ControllerModeType ControllerMode )`	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant (only for different CAN controllers)	
Parameters (in):	ControllerId	CAN controller, whose mode has changed
	ControllerMode	Notified CAN controller mode
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This callback shall notify the CanSM module about a CAN controller mode change.	

⌋()

[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`.⌋()

[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`.⌋()

8.4.3 CanSM_TransceiverModeIndication

[SWS_CanSM_00399] ⌈

Service name:	CanSM_TransceiverModeIndication	
Syntax:	`void CanSM_TransceiverModeIndication( uint8 TransceiverId, CanTrcv_TrcvModeType TransceiverMode )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different CAN Transceivers	
Parameters (in):	TransceiverId	CAN transceiver, whose mode has changed
	TransceiverMode	Notified CAN transceiver mode
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This callback shall notify the CanSM module about a CAN transceiver mode change.	

⌋()

[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`.⌋()

[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`.⌋()

8.4.4 CanSM_TxTimeoutException

[SWS_CanSM_00410] ⌈

Service name:	CanSM_TxTimeoutException	
Syntax:	`void CanSM_TxTimeoutException( NetworkHandleType Channel )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Affected CAN network
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	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.

」()

[SWS_CanSM_00411] 「The function `CanSM_TxTimeoutException` shall report `CANSM_E_UNINIT` to the DET, if the CanSM is not initialized yet.」()

[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.」()

Remarks: Reentrancy is necessary for different Channels.

8.4.5 CanSM_ClearTrcvWufFlagIndication

[SWS_CanSM_00413] 「

Service name:	CanSM_ClearTrcvWufFlagIndication	
Syntax:	`void CanSM_ClearTrcvWufFlagIndication( uint8 Transceiver )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different CAN Transceivers	
Parameters (in):	Transceiver	Requested Transceiver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This callback function shall indicate the CanIf_ClearTrcvWufFlag API process end for the notified CAN Transceiver.	

」()

[SWS_CanSM_00414] 「The function `CanSM_ClearTrcvWufFlagIndication` shall report `CANSM_E_UNINIT` to the DET, if the CanSM is not initialized yet.」()

[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`.」()

8.4.6 CanSM_CheckTransceiverWakeFlagIndication

[SWS_CanSM_00416] ⌈

Service name:	CanSM_CheckTransceiverWakeFlagIndication	
Syntax:	`void CanSM_CheckTransceiverWakeFlagIndication( uint8 Transceiver )`	
Service ID[hex]:	0x0a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant for different CAN Transceivers	
Parameters (in):	Transceiver	Requested Transceiver
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This callback function indicates the CheckTransceiverWakeFlag API process end for the notified CAN Transceiver.	

⌋()

[SWS_CanSM_00417] ⌈The function `CanSM_CheckTransceiverWakeFlagIndicatio`n shall report `CANSM_E_UNINIT` to the DET, if the CanSM module is not initialized yet.⌋()

[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`.⌋()

8.4.7 CanSM_ConfirmPnAvailability

[SWS_CanSM_00419] ⌈

Service name:	CanSM_ConfirmPnAvailability	
Syntax:	`void CanSM_ConfirmPnAvailability( uint8 TransceiverId )`	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	TransceiverId	CAN transceiver, which was checked for PN availability
Parameters (inout):	None	
Parameters (out):	None	

Return value:	None
Description:	This callback function indicates that the transceiver is running in PN communication mode.

⌋()

[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).⌋()

[SWS_CanSM_00420] ⌈
The function `CanSM_ConfirmPnAvailability` shall report `CANSM_E_UNINIT` to the DET, if the CanSM module is not initialized yet.⌋()

[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`.⌋()

8.4.8 CanSM_CurrentIcomConfiguration

[**SWS_CanSM_00587**] ⌈

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

⌋()

[**SWS_CanSM_00595**] ⌈ If the `CANSM_ICOM_SUPPORT` parameter (ref. to ECUC_CanSM_00345) is configured with the value `TRUE`, then the callback function `CanSM_CurrentIcomConfiguration` shall notify the BswM about the status of activation or deactivation of Pretended Networking (ref. to chapter 7.5) for the CAN Network, which contains the notified ControllerId in its configuration. It shall transfer the `ConfigurationId` and `Error` parameter to the BswM therefore. ⌋()

8.5 Scheduled functions

For details refer to the chapter 8.5 “Scheduled functions” in *SWS_BSWGeneral.*

8.5.1 CanSM_MainFunction

[SWS_CanSM_00065] ⌈

Service name:	CanSM_MainFunction
Syntax:	`void CanSM_MainFunction( void )`
Service ID[hex]:	0x05
Description:	Scheduled function of the CanSM

⌋(BSW0424, BSW00425, BSW00376)

[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. ⌋()

8.6 Expected Interfaces

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

8.6.1 Mandatory Interfaces

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

API function	***Description***
BswM_CanSM_CurrentIcomConfiguration	Function to inform BswM about the switch of Icom Configuration.
BswM_CanSM_CurrentState	Function called by CanSM to indicate its current state.
CanIf_CheckTrcvWakeFlag	Requests the CanIf module to check the Wake flag of the designated CAN transceiver.
CanIf_ClearTrcvWufFlag	Requests the CanIf module to clear the WUF flag of the designated CAN transceiver.
CanIf_GetTxConfirmationState	This service reports, if any TX confirmation has been done for the whole CAN controller since the last CAN controller start.

CanIf_SetControllerMode	This service calls the corresponding CAN Driver service for changing of the CAN controller mode.
CanIf_SetPduMode	This service sets the requested mode at the L-PDUs of a predefined logical PDU channel.
CanIf_SetTrcvMode	This service changes the operation mode of the tansceiver TransceiverId, via calling the corresponding CAN Transceiver Driver service.
CanNm_ConfirmPnAvailability	Enables the PN filter functionality on the indicated NM channel. Availability: The API is only available if CanNmPnEnabled is TRUE.
ComM_BusSM_ModeIndication	Indication of the actual bus mode by the corresponding Bus State Manager. ComM shall propagate the indicated state to the users with means of the RTE and BswM.
Dem_ReportErrorStatus	Queues the reported events from the BSW modules (API is only used by BSW modules). The interface has an asynchronous behavior, because the processing of the event is done within the Dem main function. OBD Events Suppression shall be ignored for this computation.

8.6.1.1 Remark: Usage of CanIf_SetPduMode

Although the CanIf module provides more requestable PDU modes, the CanSM module only uses the parameters `CANIF_ONLINE` 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. Regarding CANIF_TX_OFFLINE_ACTIVE, this state can be set either by integration code or by a (customized) CanSM (not specified yet in AUTOSAR).

8.6.2 Optional Interfaces

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

API function	***Description***
CanIf_ChangeBaudrate	This service shall change the baudrate of the CAN controller. Please note that this API is deprecated and is kept only for backward compatibility reasons. CanIf_SetBaudrate API shall be used instead to change the baud rate configuration. In the next major release this API will be deleted.
CanIf_CheckBaudrate	This service shall check, if a certain CAN controller supports a requested baudrate Please note that this API is deprecated and is kept only for backward compatibility reasons. In the next major release this API will be deleted.
CanIf_SetBaudrate	This service shall set the baud rate configuration of the CAN controller. Depending on necessary baud rate modifications the controller might have to reset.
Det_ReportError	Service to report development errors.

8.6.3 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.

8.6.3.1 <User_GetBusOffDelay>

[**SWS_CanSM_00637**] ⌈

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

⌋()

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

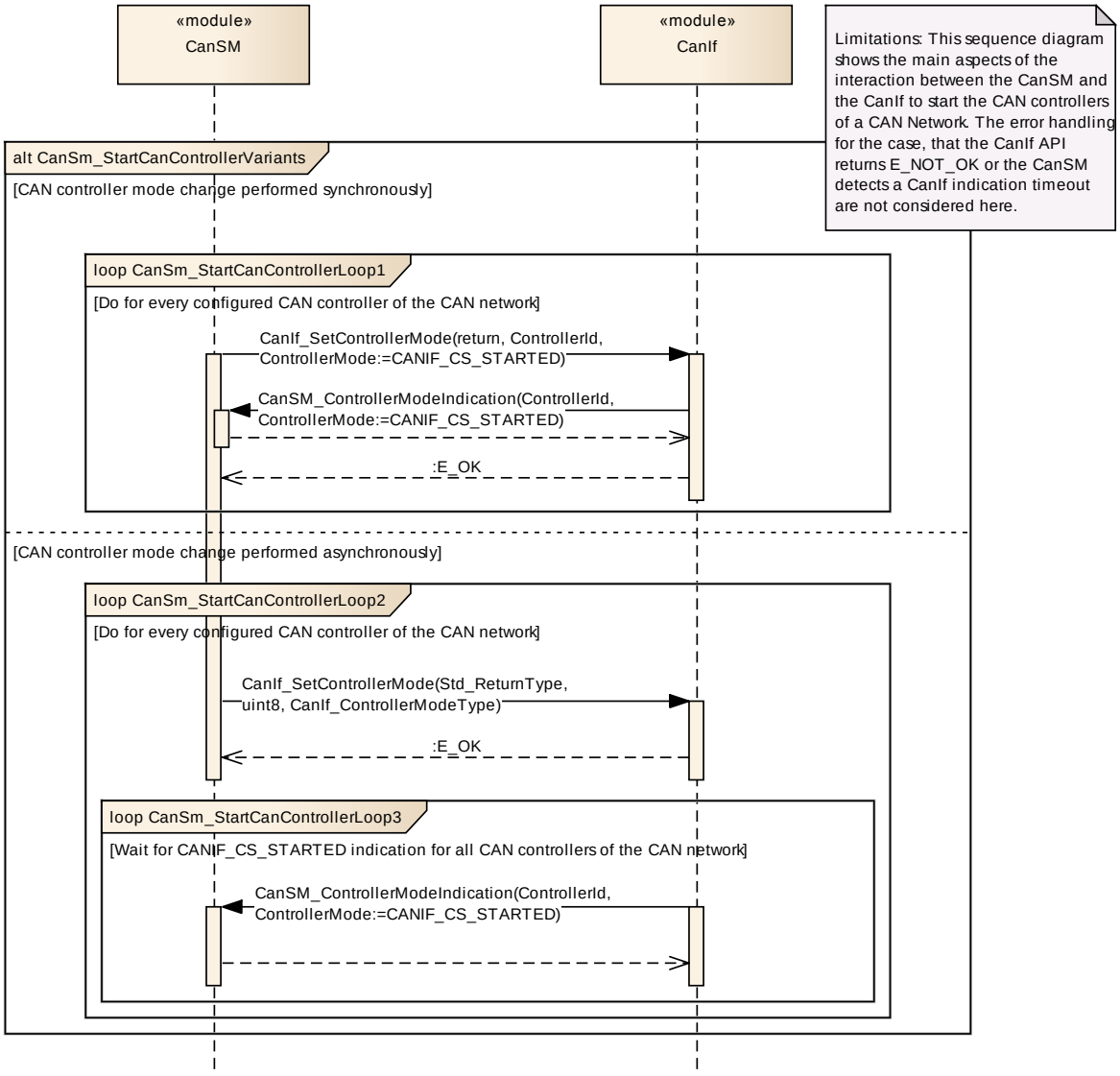

figure 9-1: Sequence diagram CanSm_StartCanController

9.2 Sequence diagram CanSm_StopCanController

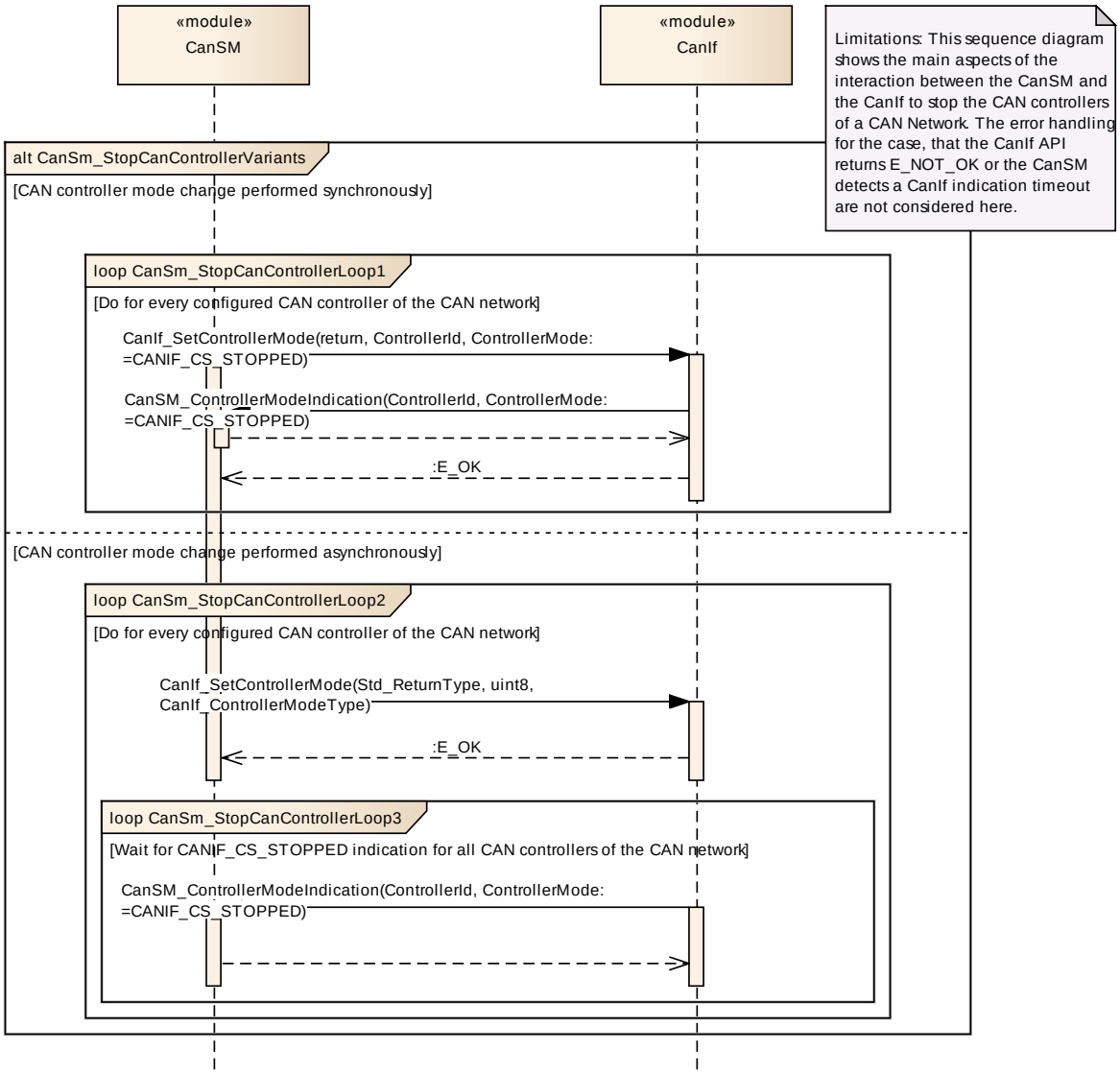

figure 9-2: Sequence diagram CanSm_StopCanController

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 Variants

[SWS_CanSM_00250] ⌈VARIANT-PRE-COMPILE: Only pre-compile parameters⌋()

[SWS_CanSM_00251] ⌈VARIANT-LINK-TIME: Mix of pre-compile and link time parameters⌋()

[SWS_CanSM_00252] ⌈VARIANT-POST-BUILD: Mix of pre compile-, link time and post build time parameters⌋()

10.2.2 CanSM

Module Name	*CanSM*
Module Description	Configuration of the CanSM module

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
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

10.2.3 CanSMConfiguration

SWS Item	**ECUC_CanSM_00123 :**
Container Name	CanSMConfiguration [Multi Config Container]
Description	This container contains the global parameters of the CanSM and sub containers, which are for the CAN network specific configuration.
Configuration Parameters	

SWS Item	**ECUC_CanSM_00335 :**		
Name	CanSMModeRequestRepetitionMax {CANSM_MODEREQ_MAX}		
Description	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.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00336 :**		
Name	CanSMModeRequestRepetitionTime {CANSM_MODEREQ_REPEAT_TIME}		
Description	Specifies in which time duration the CanSM module shall repeat mode change requests by using the API of the CanIf module.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CanSMManagerNetwork	1..*	This container contains the CAN network specific parameters of each CAN network

10.2.4 CanSMManagerNetwork

SWS Item	**ECUC_CanSM_00126 :**
Container Name	CanSMManagerNetwork
Description	This container contains the CAN network specific parameters of each CAN network
Configuration Parameters	

SWS Item	**ECUC_CanSM_00131 :**
Name	CanSMBorCounterL1ToL2 {CANSM_BOR_COUNTER_L1_TO_L2}
Description	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).		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00128 :**		
Name	CanSMBorTimeL1 {CANSM_BOR_TIME_L1}		
Description	This time parameter defines in seconds the duration of the bus-off recovery time in level 1 (short recovery time).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00129 :**		
Name	CanSMBorTimeL2 {CANSM_BOR_TIME_L2}		
Description	This time parameter defines in seconds the duration of the bus-off recovery time in level 2 (long recovery time).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00130 :**		
Name	CanSMBorTimeTxEnsured {CANSM_BOR_TIME_TX_ENSURED}		
Description	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).		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CANSM_BOR_TX_CONFIRMATION_POLLING disabled		

SWS Item	**ECUC_CanSM_00339 :**		
Name	CanSMBorTxConfirmationPolling {CANSM_BOR_TX_CONFIRMATION_POLLING}		
Description	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.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00346 :**		
Name	CanSMEnableBusOffDelay {CANSM_ENABLE_BUS_OFF_DELAY}		
Description	This parameter defines if the <User_GetBusOffDelay> shall be called for this network.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00161 :**		
Name	CanSMComMNetworkHandleRef {CANSM_NETWORK_HANDLE}		
Description	Unique handle to identify one certain CAN network. Reference to one of the network handles configured for the ComM.		
Multiplicity	1		
Type	Symbolic name reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: ComM		

SWS Item	**ECUC_CanSM_00137 :**		
Name	CanSMTransceiverId {CANSM_TRANSCEIVER_ID}		
Description	ID of the CAN transceiver assigned to the configured network handle. Reference to one of the transceivers managed by the CanIf module.		
Multiplicity	0..1		
Type	Symbolic name reference to [CanIfTrcvCfg]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIf		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
CanSMController	1..*	This container contains the controller IDs assigned to a CAN network.
CanSMDemEventParameterRef	0..1	Container for the references to DemEventParameter

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

10.2.5 CanSMDemEventParameterRefs

SWS Item	**ECUC_CanSM_00127 :**
Container Name	CanSMDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**ECUC_CanSM_00070 :**		
Name	CANSM_E_BUS_OFF {CANSM_E_BUS_OFF}		
Description	Reference to configured DEM event to report bus off errors for this CAN network.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: Dem		

No Included Containers

10.2.6 CanSMController

SWS Item	**ECUC_CanSM_00338 :**
Container Name	CanSMController
Description	This container contains the controller IDs assigned to a CAN network.
Configuration Parameters	

SWS Item	**ECUC_CanSM_00141 :**		
Name	CanSMControllerId {CANSM_CONTROLLER_ID}		
Description	Unique handle to identify one certain CAN controller. Reference to one of the CAN controllers managed by the CanIf module.		
Multiplicity	1		
Type	Symbolic name reference to [CanIfCtrlCfg]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIf		

No Included Containers

10.2.7 CanSMGeneral

SWS Item	**ECUC_CanSM_00314 :**
Container Name	CanSMGeneral
Description	Container for general pre-compile parameters of the CanSM module
Configuration Parameters	

SWS Item	**ECUC_CanSM_00342 : (Obsolete)**		
Name	CanSMChangeBaudrateApi {CANSM_CHANGE_BAUDRATE_API}		
Description	The support of the Can_ChangeBaudrate API is optional. If this parameter is set to true the Can_ChangeBaudrate API shall be supported. Otherwise the API is not supported. Please note that the Can_ChangeBaudrate API and this parameter are deprecated and will be removed in future. **Tags:** atp.Status=obsolete atp.StatusRevisionBegin=4.1.1		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_CanSM_00133 :**		
Name	CanSMDevErrorDetect {CANSM_DEV_ERROR_DETECT}		
Description	Enables and disables the development error detection and notification mechanism.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00347 :**		
Name	CanSMGetBusOffDelayFunction {CANSM_GET_BUS_OFF_DELAY_FUNCTION}		
Description	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.		
Multiplicity	0..1		
Type	EcucFunctionNameDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	

Scope / Dependency	scope: local

SWS Item	**ECUC_CanSM_00348 :**		
Name	CanSMGetBusOffDelayHeader {CANSM_GET_BUS_OFF_DELAY_HEADER}		
Description	This parameter configures the header file containing the prototype of the <User_GetBusOffDelay> callout function.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency			

SWS Item	**ECUC_CanSM_00345 :**		
Name	CanSMIcomSupport {CANSM_ICOM_SUPPORT}		
Description	Selects support of Pretended Network features in CanSM. True: Enabled False: Disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_CanSM_00312 :**		
Name	CanSMMainFunctionTimePeriod {CANSM_MAIN_FUNCTION_TIME_PERIOD}		
Description	This parameter defines the cycle time of the function CanSM_MainFunction in seconds		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0.001 .. 65.535		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_CanSM_00344 :**		
Name	CanSMPncSupport {CANSM_PNC_SUPPORT}		
Description	Enables or disables support of partial networking. False: Partial Networking is disabled True: Partial Networking is enabled		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: This parameter shall be available only if ComMPncSupport is		

	enabled in ComM

SWS Item	**ECUC_CanSM_00343 :**		
Name	CanSMSetBaudrateApi {CANSM_SET_BAUDRATE_API}		
Description	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.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_CanSM_00311 :**		
Name	CanSMVersionInfoApi {CANSM_VERSION_INFO_API}		
Description	Activate/Deactivate the version information API (CanSM_GetVersionInfo). true: version information API activated false: version information API deactivated		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.8 CanSMDemEventParameterRefs

SWS Item	**ECUC_CanSM_00127 :**
Container Name	CanSMDemEventParameterRefs
Description	Container for the references to DemEventParameter elements which shall be invoked using the API Dem_ReportErrorStatus API in case the corresponding error occurs. The EventId is taken from the referenced DemEventParameter's DemEventId value. The standardized errors are provided in the container and can be extended by vendor specific error references.
Configuration Parameters	

SWS Item	**ECUC_CanSM_00070 :**		
Name	CANSM_E_BUS_OFF {CANSM_E_BUS_OFF}		
Description	Reference to configured DEM event to report bus off errors for this CAN network.		
Multiplicity	0..1		
Type	Symbolic name reference to [DemEventParameter]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: Dem		

No Included Containers

10.2.9 CanSMController

SWS Item	**ECUC_CanSM_00338 :**
Container Name	CanSMController
Description	This container contains the controller IDs assigned to a CAN network.
Configuration Parameters	

SWS Item	**ECUC_CanSM_00141 :**		
Name	CanSMControllerId {CANSM_CONTROLLER_ID}		
Description	Unique handle to identify one certain CAN controller. Reference to one of the CAN controllers managed by the CanIf module.		
Multiplicity	1		
Type	Symbolic name reference to [CanIfCtrlCfg]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	X	VARIANT-LINK-TIME
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local dependency: CanIf		

No Included Containers

10.3 Published Information

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

11 Not applicable requirements

[CANSM999] 「 These requirements are not applicable to this specification. 」 (BSW170, BSW00375, BSW00395, BSW00416, BSW00437, BSW168, BSW00423, BSW00426, BSW00427, BSW00428, BSW00429, BSW00431, BSW00432, BSW00433, BSW00434, BSW00336, BSW00417, BSW161, BSW162, BSW005, BSW00326, BSW00347, BSW00314, BSW00435, BSW00353, BSW00361, BSW00377, BSW00308, BSW00309, BSW00360, BSW00341, BSW00439, BSW00440)