Document Title	Specification of Communication Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	079
Document Classification	Standard
Document Version	4.3.0
Document Status	Final
Part of Release	4.1
Revision	3

Document Change History

Date	Version	Changed by	Change Description
31.03.2014	4.3.0	Release Management	• Max. number of supported PNCs by ComM now 56 • ComM supports VariantPostBuild instead of VariantPostBuildSelectable • Restrictions for PNCs with ComMChannels of ComMNmVariant “PASSIVE”
31.10.2013	4.2.0	AUTOSAR Release Management	• Introduced modeling of Service Interfaces in Chapt. 8 • Repair the reset after forcing NO_COM Feature • Editorial changes • Removed chapter(s) on change documentation
26.02.2013	4.1.0	AUTOSAR Administration	• ComM allows configuration of arbitrary bus names for Bus SMs • Nm Variant Passive not configurable on individual channels anymore • Assignment of ComMPncId to Nm UserData bits specified
23.11.2011	4.0.0	AUTOSAR Administration	• Partial Network Cluster Management • Improved/Corrected illustration of start-up sequences (chap 9) • Forbid assigning ComM users to channels with NmVariant=PASSIVE • Removed re-request of unchanged communication mode in case of mismatch with BusStateManager (ComM901) • Removed remains of DEM error reporting

Document Change History			
03.11.2010	3.1.0	AUTOSAR Administration	• Table for interaction between ComM and NM added • Production error COMM_E_NET_START_IND_CHANNEL removed • Lower range of configuration parameter “ComMMainFunctionPeriod” modified
07.12.2009	3.0.0	AUTOSAR Administration	• Changed interaction between ComM and ECU State Manager (EcuM) • Changed interaction between ComM and Diagnostic Communication Manager (DCM) • Added dependencies to new modules Basic Software Mode Manager (BswM) and Ethernet State Manager • Legal disclaimer revised
23.06.2008	2.0.1	AUTOSAR Administration	Legal disclaimer revised
29.11.2007	2.0.0	AUTOSAR Administration	• Bus specific error handling (e.g. bus off handling) removed • Control of the actual bus states removed • PDU group handling removed • Initialization of Communication stack removed • Document meta information extended • Small layout adaptations made
31.01.2007	1.1.0	AUTOSAR Administration	Changed features • Restart (silent com. -> full com.) now possible even if mode limitation is active • Channel state machine changed • Sequence diagrams changed New services to upper layers • Mode indication API to RTE changed New calls to other modules • Usage of channel specific API (EcuM and ComM) to indicate that a communication channel has been woken up and has gone to sleep • API for NM control canged (Nm_PassiveStartUp, Nm_NetworkRequest, Nm_NetworkRelease) • Legal disclaimer revised • Release Notes added • “Advice for users” revised • “Revision Information” added

Document Change History			
08.05.2006	1.0.0	AUTOSAR Administration	Initial Release

Disclaimer

Advice for users

AUTOSAR Specification Documents 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 Specification Documents for illustration purposes only, and they themselves are not part of the AUTOSAR Standard. Neither their presence in such Specification Documents, 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

List of Figures

1 Introduction and functional overview

The Communication Manager Module (COM Manager, ComM) is a component of the Basic Software (BSW). It is a Resource Manager, which encapsulates the control of the underlying communication services. The ComM module controls basic software modules relating to communication and not software components or runnable entities. The ComM module collects the bus communication access requests from communication requestors (see definition of term “User” in Chapter 2) and coordinates the bus communication access requests.

The purpose of the ComM module is:

1. Simplifying the usage of the bus communication stack for the user. This includes a simplified network management handling.

2. Coordinating the availability of the bus communication stack (allow sending and receiving of signals) of multiple independent software components on one ECU.

Comment: A user should not have any knowledge about the hardware (e.g. on which channel to communicate). A user simply requests a "Communication Mode" and ComM module switches the communication capability of the corresponding channel on/off.

3. Offer an API to disable sending of signals to prevent the ECU from (actively) waking up the communication bus.

Comment: On CAN every message wakes up the bus, on FlexRay it is only possible to wake up the bus with a so called wake-up pattern.

4. Controlling of more than one communication bus channel of an ECU by implementing a channel state machine for every channel.

Comment: The ComM module requests a Communication Mode from the corresponding Bus State Manager module. The actual bus states are controlled by the corresponding Bus State Manager module.

5. Offering the possibility to force an ECU that keeps the bus awake to the 'No Communication' mode (see Section 7.3.2.2 for details).

6. Simplifying the resource management by allocating all resources necessary for the requested Communication Mode.

Comment: E.g. check if communication is allowed when a user requests 'Full Communication' mode, and prevent the ECU from shutdown during communication.

2 Acronyms and definitions

Abbreviation / Acronym:	Description:
BSW	Basic Software
BswM	Basic Software Mode Manager
ComM	Communication Manager
DCM	Diagnostic Communication Manager
Det	Development Error Tracer
EcuM	ECU State Manager module
I-PDU	Information Protocol Data Unit
NM	Network Management
PDU	Protocol Data Unit
SW-C	Software Component
VMM	Vehicle Message Matrix

Term:	Description:
DCM_ActiveDiagnostic indication	The DCM module indicates an active diagnostic session. DCM need "full communication" = `COMM_FULL_COMMUNICATION` for diagnostic purpose
Active wake-up	Wake-up caused by the hosting ECU e.g. by a sensor.
Application signal scheduling	Sending of application signals according to the VMM. Scheduling of CAN application signals is performed by the Communication Module, scheduling of LIN application I-PDUs (a PDU containing signals) is performed by the LIN interface and scheduling of FlexRay application PDUs is performed by the FlexRay Interface module.
Bus sleep	No activity required on the communication bus (e.g. CAN bus sleep).
Bus communication messages	Bus communication messages are all messages that are sent on the communication bus. This can be either a diagnostic message or an application message.
COM Inhibition status	Defines whether full communication, silent communication or wake-up is allowed or not.
Communication Channel	The medium used to convey information from a sender (or transmitter) to a receiver.
Communication Mode	Mode determining which kind of communication are allowed: "full communication" = `COMM_FULL_COMMUNICATION` "no communication" = `COMM_NO_COMMUNICATION` "silent communication" = `COMM_SILENT_COMMUNICATION` *Note:* `COMM_SILENT_COMMUNICATION` *can not be requested by a user. Internal mode for synchronizing network at shutdown*
Diagnostic PDU scheduling	Sending of diagnostic PDUs. Scheduling of CAN diagnostic PDUs is performed by the diagnostic module, scheduling of LIN diagnostic PDUs is performed by the diagnostic module and the LIN interface and scheduling of FlexRay diagnostic PDUs is performed by the diagnostic module and the FlexRay Interface module.
ECU shut down	See ECU State Manager specification [6].
Fan-out	Same message/indication are sent to multiple destinations/receivers
Independent software component	A separately developed software component performing a coherent set of functions with a minimum amount of interfaces to other software applications on an ECU. This can be e.g. a basic software component or an application software component.
Passive wake-up	Wake-up by another ECU and propagated (e.g. by bus or wake-up-line) to the ECU currently in focus.
System User	An administration functionality (a specific "user", which is generated within the internal context of the ComM) for making a default request and for overriding the user requests.

User	Concept for requestors of the ECU State Manager module and of the Communication Manager Module. A user may be the BswM, a runnable entity, a SW-C or a group of SW-Cs, which act as a single unit towards the ECU State Manager module and the Communication Manager Module.
User Request	A User can request different Communication Modes from ComM

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] Requirements on Mode Management
AUTOSAR_SRS_ModeManagement.pdf

[5] Specification of ECU Configuration
AUTOSAR_TPS_ECUConfiguration.pdf

[6] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

[7] Specification of NVRAM Manager
AUTOSAR_SWS_NVRAMManager.pdf

[8] Specification of RTE Software
AUTOSAR_SWS_RTE.pdf

[9] Specification of Generic Network Management Interface
AUTOSAR_SWS_NetworkManagementInterface.pdf

[10] Specification of Communication
AUTOSAR_SWS_COM.pdf

[11] Specification of Diagnostic Communication Manager
AUTOSAR_SWS_DiagnosticCommunicationManager.pdf

[12] Specification of LIN Interface
AUTOSAR_SWS_LINInterface.pdf

[13] Specification of FlexRay Interface
AUTOSAR_SWS_FlexRayInterface.pdf

[14] Specification of Development Error Tracer
AUTOSAR_SWS_DevelopmentErrorTracer.pdf

[16] Specification of CAN Transceiver Driver
AUTOSAR_SWS_CANTransceiverDriver.pdf

[17] Specification of CAN Interface
AUTOSAR_SWS_CANInterface.pdf

[18] Specification of FlexRay Transceiver Driver
AUTOSAR_SWS_FlexRayTransceiver.pdf

[19] Specification of PDU Router
AUTOSAR_SWS_PDURouter.pdf

[20] Requirements on IPDU Multiplexer
AUTOSAR_SWS_IPDUM.pdf

[21] Specification of System Services Mode Management
AUTOSAR_SystemServices_ModeManagement.pdf

[22] Specification of C Implementation Rules
AUTOSAR_Tr_CImplementationRules.pdf

[23] Specification of LIN State Manager
AUTOSAR_SWS_LINStateManager.pdf

[24] Specification of CAN State Manager
AUTOSAR_SWS_CANStateManager.pdf

[25] Specification of FlexRay State Manager
AUTOSAR_SWS_FlexRayStateManager.pdf

[26] Basic Software Module Description Template,
AUTOSAR_TPS_BSWModuleDescriptionTemplate.pdf

[27] Glossary,
AUTOSAR_TR_Glossary.pdf

[28] Specification of Ethernet State Manager
AUTOSAR_SWS_EthernetStateManager.pdf

[29] Specification of Basic Software Mode Manager
AUTOSAR_SWS_BSWModeManager.pdf

[30] Specification of ECU State Manager Fixed
AUTOSAR_SWS_ECUStateManagerFixed.pdf

[31] General Specification of Basic Software Modules
AUTOSAR_SWS_BSWGeneral.pdf

3.2 Related standards and norms

Not applicable.

3.3 Related specification

AUTOSAR provides a General Specification on Basic Software modules [31] (SWS BSW General), which is also valid for COM Manager.

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

4 Constraints and assumptions

4.1 Limitations

No limitations.

4.2 Applicability to car domains

No restrictions.

5 Dependencies to other modules

A context view which shows the Communication Manager Module and the dependencies to other modules is shown in Figure 1:

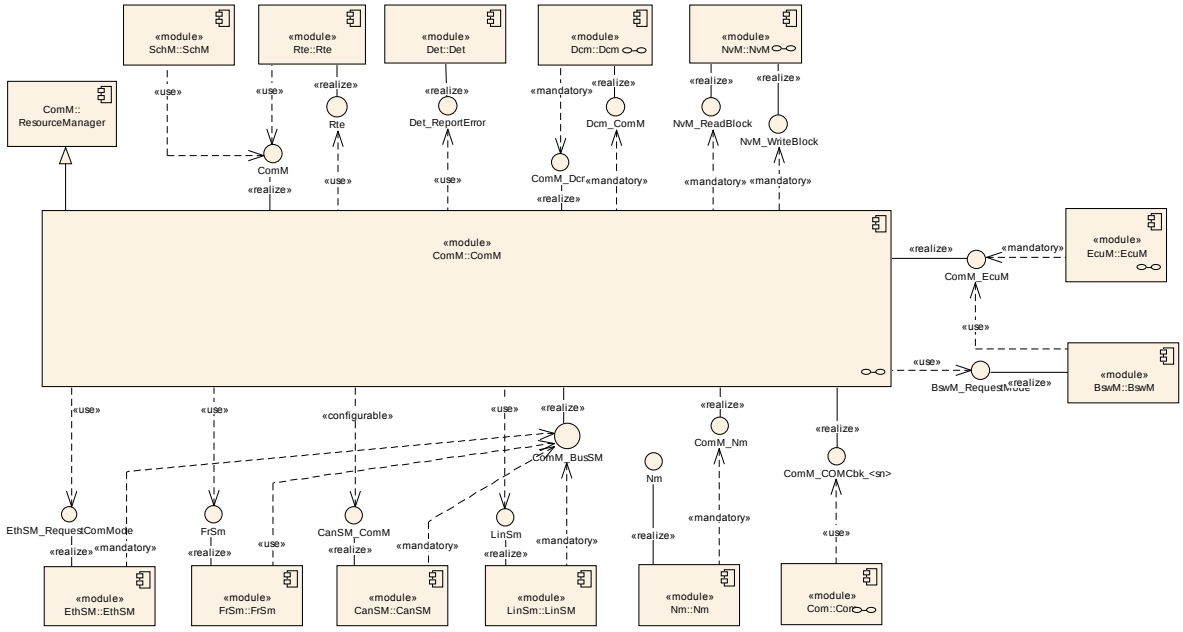

Figure 1: Communication Manager Module context view

The Communication Manager Module requests the communication capabilities, requested from the users, from the Bus State Manager modules.

5.1 File structure

5.1.1 Header file structure

[SWS_ComM_00506] ⌈The ComM module shall, depending on the ComM configuration, include the header files of the modules providing interfaces to the ComM module (see Figure 1):

ComM Schedule Manager:	`SchM_ComM.h`
RTE generated header file:	`Rte_ComM.h`
Development Error Tracer:	`Det.h`
Diagnostic Communication Manager:	`Dcm_Cbk.h`
NVRAM Manager:	`NvM.h`
ECU State Manager:	`EcuM.h`
Network Management Interface:	`Nm.h`
LIN State Manager:	`LinSM.h`
CAN State Manager:	`CanSM_ComM.h`
FlexRay State Manager:	`FrSM.h`

Ethernet State Manager:	`EthSM.h`
Basis Software Manager:	`BswM.h`
Communication:	`Com.h`

」(SRS_BSW_00436)

[SWS_ComM_00956] 「The module header file `ComM.h` shall include `Rte_ComM_Type.h` to include the types which are common used by BSW Modules and Software Components.
This file shall only contain types, which are not already defined in `Rte_ComM_Type.h`.」()

[SWS_ComM_00463] 「The ComM module shall provide in addition to `ComM_Lcfg.c` and `ComM_PBcfg.c` at least the following files:

ComM header file:	`ComM.h`
ComM callback declarations:	`ComM_Nm.h, ComM_EcuMBswM.h, ComM_Dcm.h, ComM_BusSM.h, ComM_Com.h`
ComM configuration file:	`ComM_Cfg.h`
ComM source file:	`ComM.c`」(SRS_BSW_00346, SRS_BSW_00381, SRS_BSW_00412, SRS_BSW_00415, SRS_BSW_00435)

Rationale for SWS_ComM_00463: Source code and configuration are strictly separated. User defined configurations will not imply the change of the original source code.

5.2 AUTOSAR Runtime Environment (RTE)

Every user can request a Communication Mode. The RTE propagates the user request to the ComM module and the Communication Mode indications from the ComM to the users (for details refer to [8]).

5.3 ECU State Manager (EcuM)

Two different variants of EcuM can be used, called EcuM-Fixed and EcuM-Flex. For details about the difference between to two variants, refer to EcuM-Flex [6] and EcuM-Fixed [30].

The EcuM-Fixed is responsible for initialization of ComM. Both EcuMs are also responsible to validate wake-up events and send an indication to ComM if a wake-up is validated.

If EcuM-Fixed is used, EcuM-Fixed will indicate to ComM if communication is allowed to start or not. Then EcuM-Fixed must check with ComM if the ECU can be shutdown or not, i.e. if communication is in progress or not.

If EcuM-Flex is used, the above functionality (communication allowed and shutdown of ECU) is handled by EcuM-Flex together with BswM.

5.4 Basic Software Mode Manager (BswM)

The BswM realizes two functionalities Mode Arbitration and Mode Control to allow the application of an Application Mode Management and a Vehicle Mode Management.

The BswM propagates user requests to the ComM module, if configured in the action lists of BswM to be able to request ComM modes via BswM.

The BswM controls the PDU Groups in the AUTOSAR Communication Module (COM), if the call of Com_IpduGroupControl is configured in the action list.

[**SWS_ComM_00976**] 「ComM indicates all channel main state changes and all PNC state changes to the BswM.」()

If EcuM-Flex is used, BswM will indicate to ComM if communication is allowed or not.

5.5 NVRAM Manager

The ComM module uses the NVRAM Manager to store and read non-volatile data. For details on initial values of the NVRAM data refer to Chapter 10.

Comment: The NVRAM Manager must be initialized after a power up or reset of the ECU. It must be initialized before ComM, as when ComM is initialized, ComM assumes that NVRAM is ready to be used, and that it can read back non-volatile configuration data. When ComM is de-initialized, it writes non-volatile data to NVRAM.

5.6 Diagnostic Communication Manager (DCM)

The DCM performs the scheduling of diagnostic PDUs. The DCM acts as a user by requesting Communication Mode `COMM_FULL_COMMUNICATION` via a "DCM_ActiveDiagnostic" indication if diagnostics shall be performed. The DCM does not provide an API to start/stop sending and receiving but guarantees that the communication capabilities are according to the ComM module Communication Modes.

5.7 LIN State Manager

The LIN State Manager controls the actual states of the LIN bus that correspond to a Communication Mode of the ComM module. The ComM module requests a Communication Mode from the LIN State Manager and the LIN State Manager maps the Communication Mode to a bus state.

5.8 CAN State Manager

The CAN State Manager controls the actual states of the CAN bus that correspond to a Communication Mode of the ComM module. The ComM module requests a Communication Mode from the CAN State Manager and the CAN State Manager maps the Communication Mode to a bus state.

5.9 FlexRay State Manager

The FlexRay State Manager controls the actual states of the FlexRay bus that correspond to a Communication Mode of the ComM module. The ComM module requests a Communication Mode from the FlexRay State Manager and the FlexRay State Manager maps the Communication Mode to a bus state.

5.10 Ethernet State Manager

The Ethernet State Manager controls the actual states of the Ethernet bus that correspond to a Communication Mode of the ComM module. The ComM module requests a Communication Mode from the Ethernet State Manager and the Ethernet State Manager maps the Communication Mode to a bus state.

5.11 Network Management (NM)

The ComM module uses the NM to synchronize the control of communication capabilities across the network (synchronous start-up and shutdown).

5.12 Development Error Tracer (DET)

The DET provides services to store development errors (see Section 7.8).

5.13 Communication (COM)

[SWS_ComM_00975] 「The AUTOSAR Communication module (COM) shall be used to distribute the status information about PNCs using COM signals.」()

6 Requirements traceability

Requirement	Description	Satisfied by
-	-	SWS_ComM_00066
-	-	SWS_ComM_00069
-	-	SWS_ComM_00071
-	-	SWS_ComM_00073
-	-	SWS_ComM_00084
-	-	SWS_ComM_00085
-	-	SWS_ComM_00092
-	-	SWS_ComM_00103
-	-	SWS_ComM_00133
-	-	SWS_ComM_00140
-	-	SWS_ComM_00141
-	-	SWS_ComM_00142
-	-	SWS_ComM_00143
-	-	SWS_ComM_00151
-	-	SWS_ComM_00157
-	-	SWS_ComM_00160
-	-	SWS_ComM_00161
-	-	SWS_ComM_00162
-	-	SWS_ComM_00182
-	-	SWS_ComM_00191
-	-	SWS_ComM_00215
-	-	SWS_ComM_00218
-	-	SWS_ComM_00219
-	-	SWS_ComM_00261
-	-	SWS_ComM_00266
-	-	SWS_ComM_00275
-	-	SWS_ComM_00288
-	-	SWS_ComM_00295
-	-	SWS_ComM_00296
-	-	SWS_ComM_00301
-	-	SWS_ComM_00313
-	-	SWS_ComM_00322
-	-	SWS_ComM_00355
-	-	SWS_ComM_00374
-	-	SWS_ComM_00383

-	-	SWS_ComM_00390
-	-	SWS_ComM_00391
-	-	SWS_ComM_00392
-	-	SWS_ComM_00402
-	-	SWS_ComM_00470
-	-	SWS_ComM_00472
-	-	SWS_ComM_00485
-	-	SWS_ComM_00488
-	-	SWS_ComM_00500
-	-	SWS_ComM_00501
-	-	SWS_ComM_00502
-	-	SWS_ComM_00552
-	-	SWS_ComM_00582
-	-	SWS_ComM_00583
-	-	SWS_ComM_00599
-	-	SWS_ComM_00602
-	-	SWS_ComM_00610
-	-	SWS_ComM_00619
-	-	SWS_ComM_00625
-	-	SWS_ComM_00637
-	-	SWS_ComM_00662
-	-	SWS_ComM_00663
-	-	SWS_ComM_00665
-	-	SWS_ComM_00667
-	-	SWS_ComM_00668
-	-	SWS_ComM_00669
-	-	SWS_ComM_00670
-	-	SWS_ComM_00671
-	-	SWS_ComM_00672
-	-	SWS_ComM_00673
-	-	SWS_ComM_00674
-	-	SWS_ComM_00675
-	-	SWS_ComM_00690
-	-	SWS_ComM_00694
-	-	SWS_ComM_00733
-	-	SWS_ComM_00734
-	-	SWS_ComM_00736
-	-	SWS_ComM_00740
-	-	SWS_ComM_00741

-	-	SWS_ComM_00742
-	-	SWS_ComM_00743
-	-	SWS_ComM_00744
-	-	SWS_ComM_00745
-	-	SWS_ComM_00747
-	-	SWS_ComM_00752
-	-	SWS_ComM_00778
-	-	SWS_ComM_00792
-	-	SWS_ComM_00793
-	-	SWS_ComM_00794
-	-	SWS_ComM_00795
-	-	SWS_ComM_00796
-	-	SWS_ComM_00798
-	-	SWS_ComM_00799
-	-	SWS_ComM_00800
-	-	SWS_ComM_00801
-	-	SWS_ComM_00802
-	-	SWS_ComM_00803
-	-	SWS_ComM_00805
-	-	SWS_ComM_00806
-	-	SWS_ComM_00808
-	-	SWS_ComM_00810
-	-	SWS_ComM_00812
-	-	SWS_ComM_00814
-	-	SWS_ComM_00816
-	-	SWS_ComM_00818
-	-	SWS_ComM_00819
-	-	SWS_ComM_00825
-	-	SWS_ComM_00826
-	-	SWS_ComM_00827
-	-	SWS_ComM_00828
-	-	SWS_ComM_00829
-	-	SWS_ComM_00830
-	-	SWS_ComM_00839
-	-	SWS_ComM_00840
-	-	SWS_ComM_00841
-	-	SWS_ComM_00842
-	-	SWS_ComM_00847
-	-	SWS_ComM_00848

-	-	SWS_ComM_00861
-	-	SWS_ComM_00864
-	-	SWS_ComM_00865
-	-	SWS_ComM_00866
-	-	SWS_ComM_00871
-	-	SWS_ComM_00872
-	-	SWS_ComM_00873
-	-	SWS_ComM_00874
-	-	SWS_ComM_00875
-	-	SWS_ComM_00876
-	-	SWS_ComM_00877
-	-	SWS_ComM_00878
-	-	SWS_ComM_00879
-	-	SWS_ComM_00880
-	-	SWS_ComM_00881
-	-	SWS_ComM_00882
-	-	SWS_ComM_00883
-	-	SWS_ComM_00884
-	-	SWS_ComM_00885
-	-	SWS_ComM_00886
-	-	SWS_ComM_00888
-	-	SWS_ComM_00889
-	-	SWS_ComM_00890
-	-	SWS_ComM_00891
-	-	SWS_ComM_00892
-	-	SWS_ComM_00895
-	-	SWS_ComM_00896
-	-	SWS_ComM_00897
-	-	SWS_ComM_00898
-	-	SWS_ComM_00899
-	-	SWS_ComM_00902
-	-	SWS_ComM_00903
-	-	SWS_ComM_00904
-	-	SWS_ComM_00906
-	-	SWS_ComM_00907
-	-	SWS_ComM_00908
-	-	SWS_ComM_00909
-	-	SWS_ComM_00910
-	-	SWS_ComM_00911

-	-	SWS_ComM_00912
-	-	SWS_ComM_00913
-	-	SWS_ComM_00915
-	-	SWS_ComM_00916
-	-	SWS_ComM_00919
-	-	SWS_ComM_00920
-	-	SWS_ComM_00924
-	-	SWS_ComM_00925
-	-	SWS_ComM_00926
-	-	SWS_ComM_00927
-	-	SWS_ComM_00929
-	-	SWS_ComM_00930
-	-	SWS_ComM_00931
-	-	SWS_ComM_00932
-	-	SWS_ComM_00933
-	-	SWS_ComM_00934
-	-	SWS_ComM_00936
-	-	SWS_ComM_00937
-	-	SWS_ComM_00938
-	-	SWS_ComM_00940
-	-	SWS_ComM_00942
-	-	SWS_ComM_00943
-	-	SWS_ComM_00944
-	-	SWS_ComM_00945
-	-	SWS_ComM_00946
-	-	SWS_ComM_00947
-	-	SWS_ComM_00948
-	-	SWS_ComM_00950
-	-	SWS_ComM_00951
-	-	SWS_ComM_00952
-	-	SWS_ComM_00953
-	-	SWS_ComM_00955
-	-	SWS_ComM_00956
-	-	SWS_ComM_00957
-	-	SWS_ComM_00958
-	-	SWS_ComM_00959
-	-	SWS_ComM_00960
-	-	SWS_ComM_00962
-	-	SWS_ComM_00963

-	-	SWS_ComM_00964
-	-	SWS_ComM_00966
-	-	SWS_ComM_00971
-	-	SWS_ComM_00972
-	-	SWS_ComM_00975
-	-	SWS_ComM_00976
-	-	SWS_ComM_00978
-	-	SWS_ComM_00979
-	-	SWS_ComM_00980
-	-	SWS_ComM_00981
-	-	SWS_ComM_00982
-	-	SWS_ComM_00984
-	-	SWS_ComM_00986
-	-	SWS_ComM_00987
-	-	SWS_ComM_00988
-	-	SWS_ComM_00990
-	-	SWS_ComM_00991
-	-	SWS_ComM_00992
-	-	SWS_ComM_00993
-	-	SWS_ComM_00994
-	-	SWS_ComM_00995
-	-	SWS_ComM_00996
-	-	SWS_ComM_00998
-	-	SWS_ComM_00999
-	-	SWS_ComM_01000
-	-	SWS_ComM_01001
-	-	SWS_ComM_01002
-	-	SWS_ComM_01003
-	-	SWS_ComM_01004
-	-	SWS_ComM_01005
-	-	SWS_ComM_01006
-	-	SWS_ComM_01007
-	-	SWS_ComM_01008
-	-	SWS_ComM_01009
-	-	SWS_ComM_01010
-	-	SWS_ComM_01011
-	-	SWS_ComM_01012
BSW00431	-	SWS_ComM_00499
BSW00434	-	SWS_ComM_00499

SRS_BSW_00004	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	SWS_ComM_00418
SRS_BSW_00005	Modules of the æC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces	SWS_ComM_00499
SRS_BSW_00009	All Basic SW Modules shall be documented according to a common standard.	SWS_ComM_00499
SRS_BSW_00010	The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.	SWS_ComM_00499
SRS_BSW_00101	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	SWS_ComM_00146
SRS_BSW_00158	All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation	SWS_ComM_00464
SRS_BSW_00161	The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers	SWS_ComM_00499
SRS_BSW_00162	The AUTOSAR Basic Software shall provide a hardware abstraction layer	SWS_ComM_00499
SRS_BSW_00164	The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules	SWS_ComM_00499
SRS_BSW_00167	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	SWS_ComM_00419
SRS_BSW_00168	SW components shall be tested by a function defined in a common API in the Basis-SW	SWS_ComM_00499
SRS_BSW_00170	The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands	SWS_ComM_00499
SRS_BSW_00314	All internal driver modules shall separate the interrupt frame definition from the service routine	SWS_ComM_00499
SRS_BSW_00323	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	SWS_ComM_00234
SRS_BSW_00325	The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short	SWS_ComM_00499
SRS_BSW_00326	-	SWS_ComM_00499
SRS_BSW_00327	Error values naming convention	SWS_ComM_00234
SRS_BSW_00331	All Basic Software Modules shall strictly separate error and status information	SWS_ComM_00649

SRS_BSW_00336	Basic SW module shall be able to shutdown	SWS_ComM_00147
SRS_BSW_00337	Classification of development errors	SWS_ComM_00234
SRS_BSW_00341	Module documentation shall contains all needed informations	SWS_ComM_00499
SRS_BSW_00342	It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed	SWS_ComM_00459
SRS_BSW_00343	The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit	SWS_ComM_00499
SRS_BSW_00344	BSW Modules shall support link-time configuration	SWS_ComM_00499
SRS_BSW_00345	BSW Modules shall support pre-compile configuration	SWS_ComM_00456
SRS_BSW_00346	All AUTOSAR Basic Software Modules shall provide at least a basic set of module files	SWS_ComM_00463
SRS_BSW_00348	All AUTOSAR standard types and constants shall be placed and organized in a standard type header file	SWS_ComM_00820
SRS_BSW_00353	All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header	SWS_ComM_00499
SRS_BSW_00357	For success/failure of an API call a standard return type shall be defined	SWS_ComM_00820
SRS_BSW_00358	The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void	SWS_ComM_00146
SRS_BSW_00361	All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header	SWS_ComM_00499
SRS_BSW_00369	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	SWS_ComM_00649
SRS_BSW_00373	The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention	SWS_ComM_00429
SRS_BSW_00375	Basic Software Modules shall report wake-up reasons	SWS_ComM_00499
SRS_BSW_00376	-	SWS_ComM_00429
SRS_BSW_00377	A Basic Software Module can return a module specific types	SWS_ComM_00649
SRS_BSW_00378	AUTOSAR shall provide a boolean type	SWS_ComM_00499
SRS_BSW_00381	The pre-compile time parameters shall be placed into a separate configuration header file	SWS_ComM_00463

SRS_BSW_00385	List possible error notifications	SWS_ComM_00234
SRS_BSW_00386	The BSW shall specify the configuration for detecting an error	SWS_ComM_00234
SRS_BSW_00387	The Basic Software Module specifications shall specify how the callback function is to be implemented	SWS_ComM_00620
SRS_BSW_00388	Containers shall be used to group configuration parameters that are defined for the same object	SWS_ComM_00549
SRS_BSW_00398	The link-time configuration is achieved on object code basis in the stage after compiling and before linking	SWS_ComM_00499
SRS_BSW_00404	BSW Modules shall support post-build configuration	SWS_ComM_00499
SRS_BSW_00405	BSW Modules shall support multiple configuration sets	SWS_ComM_00499
SRS_BSW_00406	A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called	SWS_ComM_00242, SWS_ComM_00612, SWS_ComM_00858
SRS_BSW_00407	Each BSW module shall provide a function to read out the version information of a dedicated module implementation	SWS_ComM_00370
SRS_BSW_00412	References to c-configuration parameters shall be placed into a separate h-file	SWS_ComM_00463
SRS_BSW_00413	An index-based accessing of the instances of BSW modules shall be done	SWS_ComM_00499
SRS_BSW_00414	The init function may have parameters	SWS_ComM_00146
SRS_BSW_00415	Interfaces which are provided exclusively for one module shall be separated into a dedicated header file	SWS_ComM_00463
SRS_BSW_00416	The sequence of modules to be initialized shall be configurable	SWS_ComM_00499
SRS_BSW_00417	Software which is not part of the SW-C shall report error events only after the DEM is fully operational.	SWS_ComM_00499
SRS_BSW_00422	Pre-de-bouncing of error status information is done within the DEM	SWS_ComM_00499
SRS_BSW_00423	BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template	SWS_ComM_00499
SRS_BSW_00424	BSW module main processing functions shall not be allowed to enter a wait state	SWS_ComM_00499
SRS_BSW_00425	The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects	SWS_ComM_00499
SRS_BSW_00426	BSW Modules shall ensure data consistency of data which is shared between BSW modules	SWS_ComM_00499

SRS_BSW_00427	ISR functions shall be defined and documented in the BSW module description template	SWS_ComM_00499
SRS_BSW_00428	A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence	SWS_ComM_00499
SRS_BSW_00429	BSW modules shall be only allowed to use OS objects and/or related OS services	SWS_ComM_00499
SRS_BSW_00432	Modules should have separate main processing functions for read/receive and write/transmit data path	SWS_ComM_00499
SRS_BSW_00433	Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler	SWS_ComM_00499
SRS_BSW_00435	-	SWS_ComM_00463
SRS_BSW_00436	-	SWS_ComM_00506
SRS_BSW_00437	Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup	SWS_ComM_00499
SRS_BSW_00438	Configuration data shall be defined in a structure	SWS_ComM_00499
SRS_BSW_00439	Enable BSW modules to handle interrupts	SWS_ComM_00499
SRS_BSW_00441	Naming convention for type, macro and function	SWS_ComM_00649, SWS_ComM_00863
SRS_ModeMgm_00049	The Communication Manager shall initiate the wake-up and keep awake physical channels	SWS_ComM_00869, SWS_ComM_00870
SRS_ModeMgm_09071	It shall be possible to limit communication modes independently for each physical channel	SWS_ComM_00303
SRS_ModeMgm_09078	The Communication Manager shall coordinate multiple communication requests	SWS_ComM_00686
SRS_ModeMgm_09080	Each physical channel shall be controlled by an independent communication mode	SWS_ComM_00051
SRS_ModeMgm_09081	The Communication Manager shall provide an API allowing collecting communication requests	SWS_ComM_00110
SRS_ModeMgm_09083	The Communication Manager shall support two communication modes for each physical channel	SWS_ComM_00845, SWS_ComM_00846, SWS_ComM_00867, SWS_ComM_00868
SRS_ModeMgm_09084	The Communication Manager shall provide an API which allows application to query the current communication mode	SWS_ComM_00083
SRS_ModeMgm_09085	The Communication Manager shall provide an indication of communication mode changes	SWS_ComM_00091

SRS_ModeMgm_09087	The Minimum duration of communication request after wakeup shall be configurable	SWS_ComM_00893, SWS_ComM_00894
SRS_ModeMgm_09089	The Communication Manager shall be able to prevent waking up physical channels	SWS_ComM_00302
SRS_ModeMgm_09090	Relationship between users and physical channels shall be configurable at pre compile time	SWS_ComM_00159
SRS_ModeMgm_09133	It shall be possible to assign physical channels to the Communication Manager	SWS_ComM_00327
SRS_ModeMgm_09149	The Communication Manager shall provide an API for querying the requested communication mode	SWS_ComM_00079
SRS_ModeMgm_09155	The Communication Manager shall provide a counter for inhibited communication requests	SWS_ComM_00138
SRS_ModeMgm_09156	It shall be provided an API to retrieve the number of inhibited "Full Communication" mode requests	SWS_ComM_00108, SWS_ComM_00224
SRS_ModeMgm_09157	It shall be possible to revoke a communication mode limitation, independently for each physical channel	SWS_ComM_00124, SWS_ComM_00156, SWS_ComM_00163
SRS_ModeMgm_09168	The Communication Manager shall support users that are connected to no physical channel	SWS_ComM_00664
SRS_ModeMgm_09172	It shall be possible to evaluate the current communication mode	SWS_ComM_00176

Document: AUTOSAR General Requirements on Basic Software Modules [3].

Requirement	Satisfied by
[SRS_BSW_00003] Version identification	SWS_ComM_00280
[SRS_BSW_00004] Version check	SWS_ComM_00418
[SRS_BSW_00005] No hard coded horizontal interfaces within MCAL	Not applicable (requirement on implementation, not on specification)
[SRS_BSW_00006] Platform independency	SWS_ComM_00462
[SRS_BSW_00007] HIS MISRA C	SWS_ComM_00462
[SRS_BSW_00009] Module User Documentation	Not applicable (requirement on documentation, not on specification)
[SRS_BSW_00010] Memory resource documentation	Not applicable (requirement on documentation, not on specification)
[SRS_BSW_00101] Initialization interface	SWS_ComM_00146

Requirement	Satisfied by
[SRS_BSW_00158] Separation of configuration from implementation	SWS_ComM_00464
[SRS_BSW_00159] Tool-based configuration	SWS_ComM_00457
[SRS_BSW_00160] Human-readable configuration data	SWS_ComM_00460
[SRS_BSW_00161] Microcontroller abstraction	Not applicable (requirement on software architecture, not for a single module)
[SRS_BSW_00162] ECU layout abstraction	Not applicable (requirement on software architecture, not for a single module)
[SRS_BSW_00164] Implementation of interrupt service routines	Not applicable (no interrupt service routines shall be implemented in ComM)
[SRS_BSW_00167] Static configuration checking	SWS_ComM_00419
[SRS_BSW_00168] Diagnostic Interface of SW components	Not applicable (the module does not support a special diagnostic interface)
[SRS_BSW_00170] Data for reconfiguration of AUTOSAR SW-Components	Not applicable (requirement for SW-Cs)
[SRS_BSW_00171] Configurability of optional functionality	ECUC_ComM_00555 ComM558_Conf ECUC_ComM_00559ComM561
[SRS_BSW_00300] Module naming convention	SWS_ComM_00462
[SRS_BSW_00301] Limit imported information	SWS_ComM_00462
[SRS_BSW_00302] Limit exported information	SWS_ComM_00462
[SRS_BSW_00304] AUTOSAR integer data types	SWS_ComM_00466
[SRS_BSW_00305] Self-defined data types naming convention	SWS_ComM_00462
[SRS_BSW_00306] Avoid direct use of compiler and platform specific keywords	SWS_ComM_00462

Requirement	Satisfied by
[SRS_BSW_00307] Global variables naming convention	SWS_ComM_00462
[SRS_BSW_00308] Definition of global data	SWS_ComM_00462
[SRS_BSW_00309] Global data with read-only constraint	SWS_ComM_00462
[SRS_BSW_00310] API naming convention	SWS_ComM_00462
[SRS_BSW_00312] Shared code shall be reentrant	SWS_ComM_00462
[SRS_BSW_00314] Separation of interrupt frames and service routines	Not applicable (this module does not implement any interrupt service routines)
[SRS_BSW_00318] Format of module version numbers	SWS_ComM_00280
[SRS_BSW_00321] Enumeration of module version numbers	SWS_ComM_00469
[SRS_BSW_00323] API parameter checking	SWS_ComM_00234
[SRS_BSW_00325] Runtime of interrupt service routines	Not applicable (this module does not implement any interrupt service routines)
[SRS_BSW_00326] Transition from ISRs to OS tasks	Not applicable (this module does not implement any interrupt service routines)
[SRS_BSW_00327] Error values naming convention	SWS_ComM_00234
[SRS_BSW_00328] Avoid duplication of code	SWS_ComM_00462
[SRS_BSW_00329] Avoidance of generic interfaces	SWS_ComM_00462
[SRS_BSW_00330] Usage of macros / inline functions instead of functions	SWS_ComM_00462
[SRS_BSW_00331] Separation of error and status values	SWS_ComM_00649, section 8.2.1
[SRS_BSW_00333] Documentation of callback function context	section 8.4
[SRS_BSW_00334] Provision of XML file	SWS_ComM_00460
[SRS_BSW_00335] Status values naming convention	section 8.2.1

Requirement	Satisfied by
[SRS_BSW_00336] Shutdown interface	SWS_ComM_00147
[SRS_BSW_00337] Classification of errors	SWS_ComM_00234
[SRS_BSW_00338] Reporting of development errors	SWS_ComM_00270
[SRS_BSW_00339] Reporting of production relevant error status	ComM515
[SRS_BSW_00341] Microcontroller compatibility documentation	Not applicable (requirement on documentation, not on specification)
[SRS_BSW_00342] Usage of source code and object code	SWS_ComM_00459
[SRS_BSW_00343] Specification and configuration of time	Not applicable (timing constraints for alive-supervision are defined in number of executions)
[SRS_BSW_00344] Reference to link-time configuration	Not applicable (this module does not provide link-time parameters)
[SRS_BSW_00345] Pre-compile-time configuration	SWS_ComM_00456
[SRS_BSW_00346] Basic set of module files	SWS_ComM_00463
[SRS_BSW_00347] Naming separation of different instances of BSW drivers	SWS_ComM_00462
[SRS_BSW_00348] Standard type header	SWS_ComM_00820
[SRS_BSW_00350] Development error detection keyword	ECUC_ComM_00555
[SRS_BSW_00353] Platform specific type header	Not applicable
[SRS_BSW_00355] Do not redefine AUTOSAR integer data types	SWS_ComM_00466
[SRS_BSW_00357] Standard API return type	SWS_ComM_00820
[SRS_BSW_00358] Return type of init() functions	SWS_ComM_00146
[SRS_BSW_00359] Return type of callback functions	section 8.4
[SRS_BSW_00360] Parameters of callback functions	section 8.4
[SRS_BSW_00361] Compiler specific language extension header	Not applicable
[SRS_BSW_00369] Do not return	SWS_ComM_00649

Requirement	Satisfied by
development error codes via API	
[SRS_BSW_00370] Separation of callback interface from API	section 8.4
[SRS_BSW_00371] Do not pass function pointers via API	SWS_ComM_00462
[SRS_BSW_00373] Main processing function naming convention	SWS_ComM_00429
[SRS_BSW_00374] Module vendor identification	SWS_ComM_00280
[SRS_BSW_00375] Notification of wake-up reason	Not applicable (this module does not implement wake-up interrupts)
[SRS_BSW_00376] Return type and parameters of main processing functions	SWS_ComM_00429
[SRS_BSW_00377] Module specific API return types	SWS_ComM_00649
[SRS_BSW_00378] AUTOSAR boolean type	Not applicable (requirement on implementation, not for specification)
[SRS_BSW_00379] Module identification	SWS_ComM_00280
[SRS_BSW_00380] Separate C-Files for configuration parameters	SWS_ComM_00503
[SRS_BSW_00381] Separate configuration header file for pre-compile time parameters	SWS_ComM_00463
[SRS_BSW_00383] List dependencies of configuration files	section 5.1
[SRS_BSW_00384] List dependencies to other modules	chapter 5
[SRS_BSW_00385] List possible error notifications	SWS_ComM_00234
[SRS_BSW_00386] Configuration for detecting an error	ComM377 SWS_ComM_00234
[SRS_BSW_00387] Specify the configuration class of callback function	SWS_ComM_00620

Requirement	Satisfied by
[SRS_BSW_00388] Introduce containers	SWS_ComM_00549
[SRS_BSW_00389] Containers shall have names	section 10.2
[SRS_BSW_00390] Parameter content shall be unique within the module	chapter 10
[SRS_BSW_00391] Parameter shall have unique names	section 10.2
[SRS_BSW_00392] Parameters shall have a type	section 10.2
[SRS_BSW_00393] Parameters shall have a range	section 10.2
[SRS_BSW_00394] Specify the scope of the parameters	section 10.2
[SRS_BSW_00395] List the required parameters (per parameter)	ECUC_ComM_00565
[SRS_BSW_00396] Configuration classes	section 10
[SRS_BSW_00397] Pre-compile-time parameters	section 10.2
[SRS_BSW_00398] Link-time parameters	Not applicable (this module does not provide link-time parameters)
[SRS_BSW_00401] Documentation of multiple instances of configuration parameters	section 10.2
[SRS_BSW_00402] Published information	SWS_ComM_00280
[SRS_BSW_00404] Reference to post build time configuration	Not applicable (this module does not provide Post-build-time parameters)
[SRS_BSW_00405] Reference to multiple configuration sets	Not applicable (this module does not provide multiple configuration sets)
[SRS_BSW_00406] Check module initialization	SWS_ComM_00242
[SRS_BSW_00407] Function to read out published parameters	SWS_ComM_00370
[SRS_BSW_00408] Configuration parameter naming convention	section 10.2

Requirement	Satisfied by
[SRS_BSW_00409] Header files for production code error IDs	ComM508
[SRS_BSW_00410] Compiler switches shall have defined values	section 10.2
[SRS_BSW_00411] Get version info keyword	ECUC_ComM_00622
[SRS_BSW_00412] Separate H-File for configuration parameters	SWS_ComM_00463
[SRS_BSW_00413] Accessing instances of BSW modules	Not applicable (requirement on implementation, not on specification)
[SRS_BSW_00414] Parameter of init function	SWS_ComM_00146
[SRS_BSW_00415] User dependent include files	SWS_ComM_00463
[SRS_BSW_00416] Sequence of Initialization	Not applicable
[SRS_BSW_00417] Reporting of Error Events by Non-Basic Software	Not applicable (requirement for SW-Cs)
[SRS_BSW_00419] Separate C-Files for pre-compile time configuration parameters	SWS_ComM_00503
[SRS_BSW_00422] Pre--de--bouncing of production relevant error	Not applicable
[SRS_BSW_00423] Usage of SW-C template to describe BSW modules with AUTOSAR Interfaces	Not applicable (requirement on documentation, not on specification)
[SRS_BSW_00424] BSW main processing function task allocation	Not applicable (requirement on implementation, not on specification)
[SRS_BSW_00425] Trigger conditions for schedulable objects	Not applicable (requirement on documentation, not on specification)
[SRS_BSW_00426] Exclusive areas in BSW modules	Not applicable (requirement on documentation, not on specification)
[SRS_BSW_00427] ISR description for BSW modules	Not applicable (this module does not implement any interrupt service routines)
[SRS_BSW_00428] Execution order dependencies of main processing functions	Not applicable (requirement on implementation, not on specification)
[SRS_BSW_00429] Restricted BSW OS functionality access	Not applicable (requirement on implementation, not on specification)
[BSW00431] The BSW Scheduler module implements task bodies	Not applicable (requirement on implementation, not on specification)
[SRS_BSW_00432] Modules should have separate	Not applicable

Requirement	Satisfied by
main processing functions for read/receive and write/transmit data path	(this module does not receive and transmit data path)
[SRS_BSW_00433] Calling of main processing functions	Not applicable (requirement on implementation, not on specification)
[BSW00434] The Schedule Module shall provide an API for exclusive areas	Not applicable (requirement on implementation, not on specification)
[SRS_BSW_00435] Module Header File Structure for the Basic Software Scheduler	SWS_ComM_00463
[SRS_BSW_00436] Module Header File Structure for the Basic Memory Mapping	SWS_ComM_00506
[SRS_BSW_00437] NoInit--Area in RAM	Not applicable
[SRS_BSW_00438] Post Build Configuration Data Structure	Not applicable (this module does not provide Post-build-time parameters)
[SRS_BSW_00439] Declaration of interrupt handlers and ISRs	Not applicable
[SRS_BSW_00440] Function prototype for callback functions of AUTOSAR Services	section 8.4
[SRS_BSW_00441] Enumeration literals and #define naming convention	SWS_ComM_00649 and SWS_ComM_00863

Document: AUTOSAR Requirements on Mode Management [4].

Requirement – Normal Operation	Satisfied by
[SRS_ModeMgm_09078] Coordinating communication requests	SWS_ComM_00686ComM283
[SRS_ModeMgm_00049] Initiating wake-up and keeping awake physical channels	SWS_ComM_00869, SWS_ComM_00870
[SRS_ModeMgm_09080] Physical channel independency	SWS_ComM_00051
[SRS_ModeMgm_09081] API for requesting communication	SWS_ComM_00110
[SRS_ModeMgm_09083] Support of different communication modes	SWS_ComM_00867, SWS_ComM_00868, SWS_ComM_00845, SWS_ComM_00846
[SRS_ModeMgm_09084] API for querying the current communication mode	SWS_ComM_00083

[SRS_ModeMgm_09172] Evaluation of current communication mode	SWS_ComM_00176
[SRS_ModeMgm_09149] API for querying the requested communication mode	SWS_ComM_00079
[SRS_ModeMgm_09085] Indication of communication mode changes	SWS_ComM_00091
[SRS_ModeMgm_09168] Pseudo-channel for local communication	SWS_ComM_00664, ECUC_ComM_00567
[SRS_ModeMgm_09071] Limit Communication Manager modes	SWS_ComM_00303
[SRS_ModeMgm_09157] Revoke Communication Manager mode limitation	SWS_ComM_00156 SWS_ComM_00163 SWS_ComM_00124
[SRS_ModeMgm_09087] Minimum duration of communication request after wakeup	SWS_ComM_00893, SWS_ComM_00894
[SRS_ModeMgm_09089] Preventing waking up physical channels	SWS_ComM_00302
[SRS_ModeMgm_09155] Counting of inhibited communication requests	SWS_ComM_00138
[SRS_ModeMgm_09156] API to retrieve the number of inhibited "Full Communication" mode requests	SWS_ComM_00224 SWS_ComM_00108

Requirement - Configuration	Satisfied by
[SRS_ModeMgm_09090] User-to-channel relationship	SWS_ComM_00159
[SRS_ModeMgm_09133] Assigning physical channels to the Communication Manager	SWS_ComM_00327
[SRS_ModeMgm_09132] Assigning Network Mangement to physical channels	ECUC_ComM_00568
[SRS_ModeMgm_09141] Configuration of	ECUC_ComM_00559

Requirement - Configuration	Satisfied by
physical channel wake-up prevention	
[SRS_ModeMgm_09207] Configurable Assignment of Bus State Managers	ECUC_ComM_00567

7 Functional specification

The Communication Manager (ComM) module simplifies the resource management for the users, whereat users may be runnable entities, SW-Cs, the BswM (e.g. SW-C request via BswM) or DCM (communication needed to diagnostic purpose).

[SWS_ComM_00867] ⌈The ComM shall provide three different Communication Modes. The highest Communication Mode shall be `COMM_FULL_COMMUNICATION`. The lowest Communication Mode shall be `COMM_NO_COMMUNICATION`.⌋(SRS_ModeMgm_09083)

[SWS_ComM_00151] ⌈For a user it shall only be possible to request the Communication Modes `COMM_NO_COMMUNICATION` and `COMM_FULL_COMMUNICATION` (see `ComM_RequestComMode()`, SWS_ComM_00110).⌋()

Rationale for SWS_ComM_00151: The Communication Mode `COMM_SILENT_COMMUNICATION` and sub-modes/sub-states are only necessary for synchronization with AUTOSAR NM.

[SWS_ComM_00868] ⌈The Communication Mode `COMM_SILENT_COMMUNICATION` shall only be used for network synchronization.⌋(SRS_ModeMgm_09083)

Note: The possibility to request `COMM_SILENT_COMMUNICATION` mode is removed since release 2.0.

Comment: The ComM module allows querying the Communication Mode requested by a particular user (see `ComM_GetRequestedComMode()`, SWS_ComM_00079).

Comment: The ComM module allows querying the actual Communication Mode of a channel (see `ComM_GetCurrentComMode()`, SWS_ComM_00083)

[SWS_ComM_00845] ⌈In COMM_FULL_COMMUNICATION mode, the ComM module shall allow transmission and reception on the affected physical channel.⌋(SRS_ModeMgm_09083)

[SWS_ComM_00846] ⌈In COMM_NO_COMMUNICATION mode, the ComM module shall prevent transmission and reception on the affected physical channel.⌋(SRS_ModeMgm_09083)

[SWS_ComM_00686] ⌈If at least one of multiple independent user requests demands a higher Communication Mode (see SWS_ComM_00867 and SWS_ComM_00868), the ComM module shall set this higher Communication Mode as the target Communication Mode.⌋(SRS_ModeMgm_09078)

Rationale for SWS_ComM_00686: ComM coordinates multiple independent user requests according to the "highest wins" strategy: COMM_FULL_COMMUNICATION Communication Mode overrules COMM_NO_COMMUNICATION.

[SWS_ComM_00500] ⌈The ComM module shall not queue user requests. The latest user request of the same user shall overwrite an old user request even if the request is not finished.⌋()

[SWS_ComM_00866] ⌈If configuration parameter ComMNmVariant=FULL|LIGHT|NONE (ECUC_ComM_00568), an DCM_ActiveDiagnostic indication shall be treated as a COMM_FULL_COMMUNICATION request for the specified communication channel (see ComM_DCM_ActiveDiagnostic(channel), SWS_ComM_00873).⌋()

Rationale for SWS_ComM_00866: If more channels needed for diagnostic purpose, DCM needs to indicate DCM_ActiveDiagnostic for each channel.

[SWS_ComM_00092] ⌈There shall be one Communication Mode target state (evaluated according to SWS_ComM_00686) per communication channel. This target mode can differ temporarily from the actual mode controlled by the corresponding Bus State Manager module.⌋()

Comment: Mode switching by the corresponding Bus State Manager module takes time and a mode inhibition can be active.

[SWS_ComM_00084] ⌈The ComM module shall propagate a call of `ComM_GetCurrentComMode()` (see SWS_ComM_00083) to the Bus State Manager module(s) for the channel(s) the user are configured to (see also SWS_ComM_00176 and SWS_ComM_00798)⌋()

Rationale for SWS_ComM_00084: State requests have to be propagated to the corresponding Bus State Manager module since the ComM module does not control the actual bus state.

Comment: This feature is not used by a "normal SW-C" because they don't have knowledge about channels. This feature is necessary for privileged SW-Cs, which (have to) know about the system topology, e.g. system diagnostic functions.

[SWS_ComM_00884] ⌈The ComM module shall store status if communication for a channel is allowed or not allowed in separate `CommunicationAllowed` boolean flags for all supported channels. The default value after ComM initialization shall be communication is not allowed, i.e. `CommunicationAllowed=FALSE.`⌋()

[SWS_ComM_00885] ⌈Status changes for communication allowed or not allowed in SWS_ComM_00884 shall be provided to ComM in `ComM_CommunicationAllowed(<channel>, TRUE|FALSE)` (SWS_ComM_00871) indications.⌋()

7.1 Partial Network Cluster Management

7.1.1 Overview

ComM implements a state machine for each partial network cluster (PNC) to represent the communication mode of a PNC.

Each PNC has its own state. The state definitions are related to the states of ComM for a simple mapping.

ComM users are used to request and release the PNCs.

The status of all PNCs on the nodes of a system channel is exchanged via network management user data.

Each PNC uses a dedicated bit position within a bit vector in the NM user data on CAN and FlexRay. If a PNC is requested by a local ComM user on the node, the node sets the corresponding bit in the NM user data to 1. If the PNC is not requested anymore; the node sets the corresponding bit in the NM user data to 0. The BusNms collect and aggregate the NM user data for the PNCs and provide the status via a COM bit vector by means of a COM signal to ComM.

Each PNC uses the same bit position in the NM user data on every system channel with NM. ComM uses two types of bit vector named EIRA and ERA to exchange PNC status information. The definition of “EIRA” and “ERA” are located in the AUTOSAR SWS CAN NM and AUTOSAR SWS FlexRay NM.

ComM requests and releases the system communication bus channels needed for a PNC on a node.

Enabling or disabling the partial network cluster management in the node shall be post-build configurable. In order to enable or disable the PNC during runtime e.g. by a diagnostic service, the requested enabling or disabling PNC shall be stored non volatile and executed after the ECU reset during the startup.

Partial networking shall be supported on the bus types CAN, FlexRay. Activation and deactivation of the I-PDU groups of the PNCs on a FlexRay node is required to avoid false timeouts. Starting and Stopping of I-PDU groups in COM are handled in BSWM. Deactivation of single FlexRay ECU is not possible.

7.1.2 Partial Network Cluster Management Functionality

[SWS_ComM_00910] ⌈PNC functionality shall only exist if the parameter ComMPncSupport is set to TRUE. (see SWS_ComM_00839_Conf).⌋()

[SWS_ComM_00911]「Enabling or disabling of the PNC functionality shall be post-build configurable using the parameter ComMPncEnabled (see SWS_ComM_00878_Conf).」()

[SWS_ComM_00999]「The parameter ComMPncEnable shall be stored non volatile and evaluated after the ECU reset during the startup.」()

Comment: This is required to be able to enable or disable the PNC during runtime e.g. by a diagnostic service.

Comment: The ComM module notifies the BswM about every state change of the PNC state machine by calling `BswM_ComM_CurrentPncMode()`. (refer to SWS_ComM_00908)

[SWS_ComM_00982]「For exchanging PNC status information, bit vectors shall be used. (i.e. only one signal containing a maximum of 56 PNC status information bits).」()

Comment: ComM expects that the PNC bit vector is configured as an array of type `uint8_n`, see config parameter `ComMPncComSignalRef`.

[SWS_ComM_00825]「 The byteIndex and bitIndex, in which a bit corresponding to one ComMPncId resides, shall be determined as follows:
byteIndex=(ComMPncId div 8)-pncVectorOffset
bitIndex=(ComMPncId mod 8)」()

Comment: ComM825 defines only the calculation of the byteIndex and bitIndex, not how it shall be implemented.

[SWS_ComM_00984]「ComM receives the bit vectors (signals) which can be ComMPncComSignalKind EIRA or ERA using Com_ReceiveSignal()」()

[SWS_ComM_00986]「The ComM shall provide the API `ComM_COMCbk_<sn>()` to indicate a change of signal(s) within the module communication.」()

[SWS_ComM_00916] ⌈The ComM module shall be able to distribute the status of a PNC (result of the PNC state machine) via one or more communications busses using one or more COM signals ,as a bit vector, containing a bit which represents the status of the PNC with ComMPncComSignalDirection “TX” assigned to this PNC. (For more details, refer to SWS_ComM_00988)⌋()

7.1.3 ComM PNC state machine

[SWS_ComM_00953] ⌈If the PNC functionality is enabled using the configuration parameter ComMPncEnabled set to TRUE (see SWS_ComM_00878_Conf), all actions related to PNC changes shall be executed before the channel related actions (channel related actions, see Chapter 7.3). ⌋()

[SWS_ComM_00909] ⌈For every Partial Network, only one PNC state machine shall be implemented (i.e. One PNC state machine per PNC, independent of the amount of ComMChannels).⌋()

[SWS_ComM_00920] ⌈The ComM module shall support up to 56 PNC state machines. ⌋()

[SWS_ComM_00924] ⌈The PNC state machine shall consist of the two main states PNC_FULL_COMMUNICATION and PNC_NO_COMMUNICATION.⌋()

[SWS_ComM_00907] ⌈The PNC main state PNC_FULL_COMMUNICATION shall consist of the
sub states PNC_PREPARE_SLEEP, PNC_READY_SLEEP and PNC_REQUESTED.⌋()

[SWS_ComM_00908] ⌈Every state change (main or substate), excluding entering of the main state PNC_NO_COMMUNICATION coming from PowerOff, shall be notified by the API call BswM_ComM_CurrentPncMode() with the entered PNC state.⌋()

[SWS_ComM_00978]「State transitions of the PNC state machines in ComM, triggered by a call to ComM_RequestComMode() shall be executed in the `ComM_MainFunction_<Channel_Id>` only.」()

Comment: Every PN activation triggers sending of the PN-vector n-times thus it would increase the busload without debouncing.

[SWS_ComM_00944]「If at least one bit corresponding to the PNC within the Rx bitvectors with signal type "EIRA" equals '1', then the bit corresponding to this PNC whithin EIRA in ComM shall be set to '1'」()

[SWS_ComM_00945]「If the configuration parameter ComMPncGatewayEnabled (see SWS_ComM_00840_Conf) is true and the parameter ComMPncGatewayType is set to COMM_GATEWAY_TYPE_ACTIVE for a ComMChannel and at least one bit corresponding to the PNC within the Rx bitvectors with signal type "ERA" equals '1', then the bit corresponding to this PNC within ERA in ComM shall be set to '1'.」()

[SWS_ComM_00971]「The trigger `ComM_COMCbk` represents a notification by the AUTOSAR Communication module about a received signal containing PNC status information called ERA of EIRA.」()

[SWS_ComM_00972]「The trigger “ComMUser” represents a notification about a communication request of a ComMUser by calling the API `ComM_RequestComMode().`」()

[SWS_ComM_00987]「Within the `ComM_MainFunction_<Channel_Id>` of a channel that is mapped to one or more PNCs, the requested state shall be handled in the following order:

1. ComM user requests of ComM users mapped to one or more PNCs of that channel
2. ComM user requests of ComM users mapped to that channel
3. ERA (if the configuration switch `ComMPncGatewayEnabled` is set to TRUE)
4. EIRA」()

Comment: Requests are handled in main functions of those channels they affect.

[SWS_ComM_00919]「It shall be possible to assign more than one COM signal containing bits representing the PNC to one PNC using the configuration container ComMPncComSignal (see SWS_ComM_00881_Conf).」()

Rational: This allows the configurator to assign e.g. one EIRA and n ERAs to one PNC.

Comment: The different IDs of EIRA can be configured to the physical supported channels FlexRay, Can1, Can2 …

[SWS_ComM_00827]「 Regarding "Communication allowed" and mode inhibitions, requests originating from a pnc state machine shall be treated like user requests for the according channels.」()

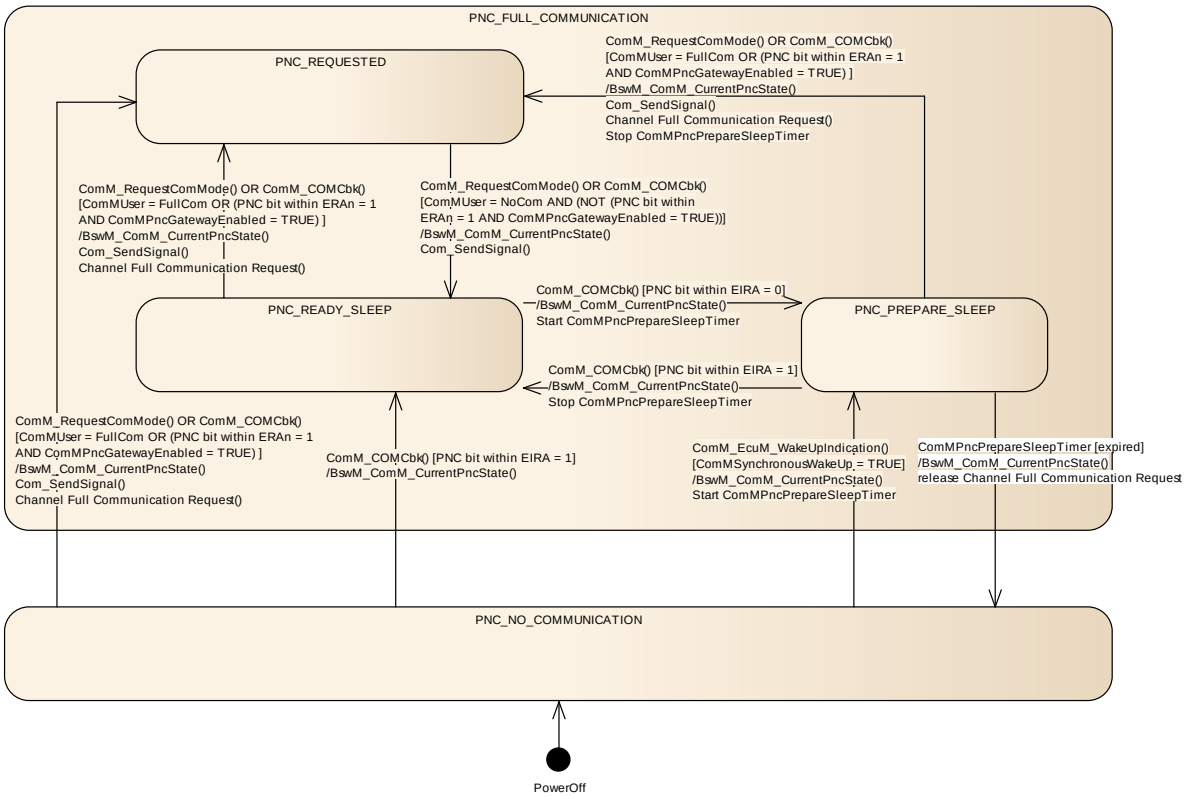

Figure 2: PNC State Machine

[SWS_ComM_00988]「PNC State Machine」()

7.1.3.1 Behavior in PNC main state PNC_NO_COMMUNICATION

[SWS_ComM_00926]「The PNC main state PNC_NO_COMMUNICATION shall be the default PNC state from power off.」()

[SWS_ComM_00925]「The main state PNC_NO_COMMUNICATION shall be the target state as long as the PNC is neither requested ECU internally nor requested externally.」()

[SWS_ComM_00931]⌈If the API `ComM_EcuM_WakeUpIndication()` is called in PNC state PNC_NO_COMMUNICATION, and the configuration switch `ComMSynchronousWakeUp` is set to TRUE (see ComM695), the PNC main state PNC_NO_COMMUNICATION shall be left and the PNC sub state PNC_PREPARE_SLEEP shall be entered.⌋()

[SWS_ComM_00990]⌈If the API `ComM_EcuM_WakeUpIndication()` is called in PNC state PNC_NO_COMMUNICATION, and the configuration switch ComMSynchronousWakeUp is set to FALSE, the PNC main state PNC_NO_COMMUNICATION shall be the current state.⌋()

Comment: In case of asynchronous wake up, the PNC state shall stay in PNC_NO_COMMUNICATION until the PNC request is received (PNC bit in EIRA is set to ‘1’).

[SWS_ComM_00932]⌈When at least one ComMUser assigned to this PNC requests “Full Communication” in PNC main state PNC_NO_COMMUNICATION, this state shall be left and the sub state PNC_REQUESTED of the main state PNC_FULL_COMMUNICATION shall be entered.⌋()

[SWS_ComM_00933]⌈When in main state PNC_NO_COMMUNICATION at least one bit representing this PNC in EIRA changes to ‘1’, the main state PNC_NO_COMMUNICATION shall be left and the PNC_READY_SLEEP shall be entered. ⌋()

[SWS_ComM_00934]⌈When in main state PNC_NO_COMMUNICATION at least one bit representing this PNC in an ERAn changes to ‘1’, the main state PNC_NO_COMMUNICATION shall be left and the sub state PNC_REQUESTED shall be entered if the parameter `ComMPncGatewayEnabled` (SWS_ComM_00840_Conf) equals TRUE.⌋()

[SWS_ComM_00830]⌈ In state PNC_NO_COMMUNICATION all "Full Communication" requests originating from this PNC shall be released.⌋()

7.1.3.2 On entry of PNC main state PNC_NO_COMMUNICATION from PowerOff

[SWS_ComM_00927]⌈After switching on the power supply, main state PNC_NO_COMMUNICATION shall be entered from PowerOff.⌋()

7.1.3.3 Behavior in PNC main state PNC_FULL_COMMUNICATION

[SWS_ComM_00929]⌈All ComMChannels assigned to this PNC shall be in state Full Communication.⌋()

7.1.3.4 On entry of PNC sub state PNC_REQUESTED

[SWS_ComM_00930] ⌈When entering the PNC sub state PNC_REQUESTED and if ComMPncGatewayEnabled = FALSE, the API `Com_SendSignal()` shall be called with the value '1' for the bit representing this PNC for the Com signal assigned to this PNC with `ComMPncComSignalDirection` "TX".⌋()

[SWS_ComM_00992] ⌈When entering the PNC sub state PNC_REQUESTED and if ComMPncGatewayEnabled = TRUE, the PNC bit within ERA shall be calculated according to SWS_ComM_00959. The API `Com_SendSignal()` shall be then called with the result of the bits representing this PNC for all Com signals assigned to this PNC with `ComMPncComSignalDirection` "TX".⌋()

[SWS_ComM_00993] ⌈Every time the sub state PNC_REQUESTED is entered from other states, all configured ComM channels for this PNC shall be requested "Full communication", even if the channel is already requested.⌋()

7.1.3.5 Behavior in PNC sub state PNC_REQUESTED

[SWS_ComM_00936] ⌈As long as at least one ComMUser assigned to this PNC requests "Full Communication", PNC_REQUESTED shall be the current PNC state.⌋()

[SWS_ComM_00937] ⌈As long as a PNC is requested remotely (i.e. at least one bit within the ERA signal assigned to this PNC equals '1') and the configuration switch `ComMPncGatewayEnabled` is set to TRUE (see SWS_ComM_00840_Conf), PNC_REQUESTED shall be the current PNC state.⌋()

[SWS_ComM_00938] ⌈When all ComMUsers assigned to this PNC request "No Communication", the sub state PNC_REQUESTED shall be left and the sub state PNC_READY_SLEEP shall be entered, if the configuration switch `ComMPncGatewayEnabled` is set to FALSE.⌋()

[SWS_ComM_00991] ⌈When all ComMUsers assigned to this PNC request "No Communication" and the PNC bit in all ERAn is equal to 0, the sub state PNC_REQUESTED shall be left and the sub state PNC_READY_SLEEP shall be entered, if the configuration switch `ComMPncGatewayEnabled` is set to TRUE.⌋()

7.1.3.6 On entry PNC sub state PNC_READY_SLEEP

[SWS_ComM_00960] ⌈When entering the PNC sub state PNC_READY_SLEEP from PNC_REQUESTED, the API `Com_SendSignal()` shall be called with the value '0' for the bit representing this PNC for all Com signals assigned to this PNC with `ComMPncComSignalDirection` "TX".⌋()

7.1.3.7 Behavior in PNC sub state PNC_READY_SLEEP

[SWS_ComM_00942] ⌈As long as the PNC is requested (i.e. at least one PNC bit within EIRA equals '1') and no ComMUser assigned to this PNC requests "Full Communication", PNC_READY_SLEEP shall be the current state.⌋()

[SWS_ComM_00940] ⌈If the PNC is released (i.e. all PNC bits within EIRA equals '0'), the sub state PNC_READY_SLEEP shall be left and the sub state PNC_PREPARE_SLEEP shall be entered. ⌋()

7.1.3.8 On entry of PNC sub state PNC_PREPARE_SLEEP

[SWS_ComM_00952] ⌈If the sub state PNC_PREPARE_SLEEP is entered, the timer `ComMPncPrepareSleepTimer` (see SWS_ComM_00841_Conf) shall be started with the configured initial value.⌋()

7.1.3.9 Behavior in PNC sub state PNC_PREPARE_SLEEP

[SWS_ComM_00943] ⌈As long as the timer `ComMPncPrepareSleepTimer` (see SWS_ComM_00841_Conf) is running and no changes in ComMUser, EIRA or ERAn occur, PNC_PREPARE_SLEEP shall be the current state.⌋()

[SWS_ComM_00947] ⌈When the timer `ComMPncPrepareSleepTimer` (see SWS_ComM_00841_Conf) expires, the PNC sub state PNC_PREPARE_SLEEP shall be left and the PNC main state PNC_NO_COMMUNICATION shall be entered.⌋()

[SWS_ComM_00948] ⌈When in PNC_PREPARE_SLEEP at least one ComMUser assigned to this PNC requests "Full Communication", the PNC_PREPARE_SLEEP state shall be left. The timer `ComMPncPrepareSleepTimer` shall be stopped and the sub state PNC_REQUESTED state shall be entered.⌋()

[SWS_ComM_00950] ⌈When in PNC_PREPARE_SLEEP at least one PNC bit within EIRA changes to '1', the sub state PNC_PREPARE_SLEEP shall be left. The timer `ComMPncPrepareSleepTimer` shall be stopped and the sub state PNC_READY_SLEEP shall be entered. ⌋()

[SWS_ComM_00951] ⌈When in sub state PNC_PREPARE_SLEEP at least one PNC bit within ERAn changes to '1' and the parameter `ComMPncGatewayEnabled` equals TRUE, the sub state PNC_PREPARE_SLEEP shall be left. The timer `ComMPncPrepareSleepTimer` shall be stopped and the sub state PNC_REQUESTED shall be entered.⌋()

7.1.4 PNC Gateway

[SWS_ComM_00981]「If the configuration parameter `ComMPncGatewayEnabled` (see SWS_ComM_00840_Conf) is TRUE, the default gateway type shall be active (COMM_GATEWAY_TYPE_ACTIVE).」()

Comment to SWS_ComM_00981:
It can be assumed that both signal types (i.e. `ComMPncComSignalKind = EIRA` and `ComMPncComSignalKind = ERA)` are configured.

7.1.4.1 Active PNC Gateway

[SWS_ComM_00964]「If the configuration parameter `ComMPncGatewayEnabled` (see SWS_ComM_00840_Conf) is TRUE and the parameter `ComMPncGatewayType` is set to COMM_GATEWAY_TYPE_ACTIVE for a ComMChannel (see SWS_ComM_00842_Conf), the active PNC gateway shall behave as described in SWS_ComM_00988. 」()

Comment: An active PNC gateway on a system channel shall be the last node on a system channel that releases a PNC.

[SWS_ComM_00966]「An active PNC gateway shall evaluate all system channels ERAn signals (ERAn bit vectors) if the active PNC gateway is the last node requesting a PNC.」()

Comment: If the bit for a PNC is equal to zero in all ERAn, no other node than the PNC gateway is requesting the PNC.

7.1.4.2 Passive PNC Gateway

Comment: The passively coordinated channels exist only if they are connected to more than one PNC gateway. If the PNC gateway functionality of ComM is enabled (ComMPncGatewayEnabled = true) ComM channels mapped to this gateway can be set to type active or passive (COMM_GATEWAY_TYPE_ACTIVE or COMM_GATEWAY_TYPE_PASSIVE). If a ComM channel is mapped to two different PNC gateways, only one gateway coordinates this channel actively, while the other passively. That means, a PNC gateway is always mapped to at least one ComM channel type active and may be mapped to one or some ComM channels type passive.

[SWS_ComM_00955]「If the configuration parameter `ComMPncGatewayEnabled` (see SWS_ComM_00840_Conf) is enabled and the parameter `ComMPncGatewayType` is set to COMM_GATEWAY_TYPE_PASSIVE for a ComMChannel (see SWS_ComM_00842_Conf), the passive PNC Gateway behavior for this ComMChannel shall be implemented by using the filter mechanism for the COM Tx signals as described in [SWS_ComM_00959.」()

Comment: A PNC gateway requests the PNC if a local ComM user requests the PNC or at least one PNC bit within ERA originate from the actively coordinated system channels of a passive PNC gateway is not equal to 0.

[SWS_ComM_00959] ⌈The bit representing this PNC within the COM Tx signals shall be set to '0' (before calling the AUTOSAR COM module) for all ComMChannels configured as ComMPncGatewayType = "COMM_GATEWAY_TYPE_PASSIVE" if

- all ComMUsers assigned to this PNC request "No Communication", AND, all ComMPncComSignals, received by `Com_ReceiveSignal()` from a channel having the channel attribute ComMPncGatewayType "COMM_GATEWAY_TYPE_ACTIVE" and having the signal attribute `ComMPncComSignalDirection` "RX" and having the signal attribute `ComMPncComSignalKind` "ERA" are equal to "0".⌋()

Comment to SWS_ComM_00959: A PNC gateway calculates the PNCs bit value in the ERA Tx bitvectors to be sent for a passively coordinated channel, in the same manner as the bit value in ERA for an actively coordinated channel (SWS_ComM_00946), but sets the NC's bit to '0' according to the rules of SWS_ComM_00959.

[SWS_ComM_00946] ⌈In case the configuration switch ComMPncGatewayEnabled is set to TRUE and the parameter ComMPncGatewayType is set to COMM_GATEWAY_TYPE_PASSIVE, the signal value representing a PNC in ERA shall be new calculated according to SWS_ComM_00959 before calling ComSendSignal().⌋()

7.1.5 ComM User to PNC Relations

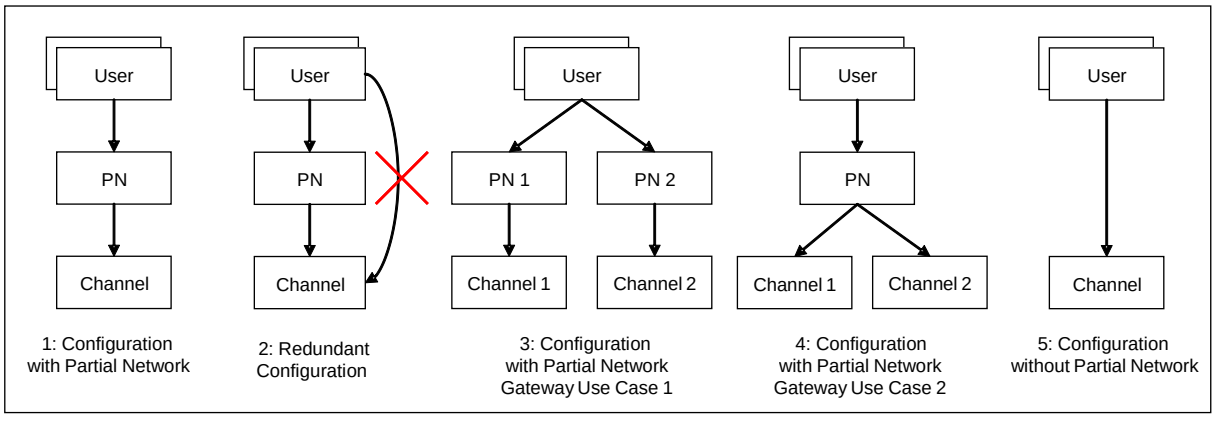

Figure 3: User to Partial network and channel Mapping Use Cases

[SWS_ComM_00912] ⌈It shall be possible to map a configurable amount of ComMUsers to one or more PNCs using the parameter ComMUserPerPnc (see SWS_ComM_00876_Conf).⌋()

[SWS_ComM_00994]「No restrictions from the configuration of the BusNm Filter for partial networking shall apply to ComM user assignment to PNCs.」()

Comment: The BusNM Filter configuration shall be independant from the ComM PNC configuration.

Rational: This enables waking up a PNC without being a member of the PNC, e.g. if a node just triggers a wake up of a PNC but the node is not kept awake by the PNC and other nodes keep the PNC awake

[SWS_ComM_00995]「It shall be possible to map a configurable amount of ComMUsers to one or more ComM channels using the parameter ComMUserPerChannel.」()

Comment: The existing mapping of ComM users to system channels shall still be possible for backward compatibility. (i.e. the configuration containers will stay untouched)

[SWS_ComM_00913]「It shall be possible to map a configurable amount of PNC(s) to a configurable amount of ComM channels using the parameter ComMChannelPerPnc (see SWS_ComM_00880_Conf).」()

[SWS_ComM_00996]「It shall not be possible to map a ComMUsers to a PNC and in addition to a ComM channel which is already referenced by the PNC (see figure 3 Use Case 2)」()
Rational: Avoid redundant configuration since the channel is implicitly already referenced by the PNC.

7.2 ComM channel state machine

[SWS_ComM_00979]「If the optional PNC functionality is enabled (see SWS_ComM_00839_Conf), all PNC actions shall be performed before the channel related actions are executed. 」()

[SWS_ComM_00980]「If the parameter ComMPncNmRequest equals TRUE (see SWS_ComM_00886_conf), if the “FULL Communication” is requested due to a change in the PNC state machine to PNC_REQUESTED (see SWS_ComM_00993)API Nm_NetworkRequest() shall be called, even if the current state is already "Full communication". 」()

Rationale: It is the trigger to enable the NM to transmit the NM message immediately n-times (n=configurable) to ensure a wake up and a synchronization of the PNC transceiver.

[SWS_ComM_00051] 「ComM shall implement one channel state machine as shown in Figure 4 with requirements as listed in Table 1 for every communication channel independently.」(SRS_ModeMgm_09080)

Rationale for SWS_ComM_00051: Needed communication capability of channels may be different, thus the controlling must be independent.

Use Case for SWS_ComM_00051: On an ECU with CAN and LIN channel, only the LIN requires full communication to request e.g. sensor values while the CAN remains inactive.

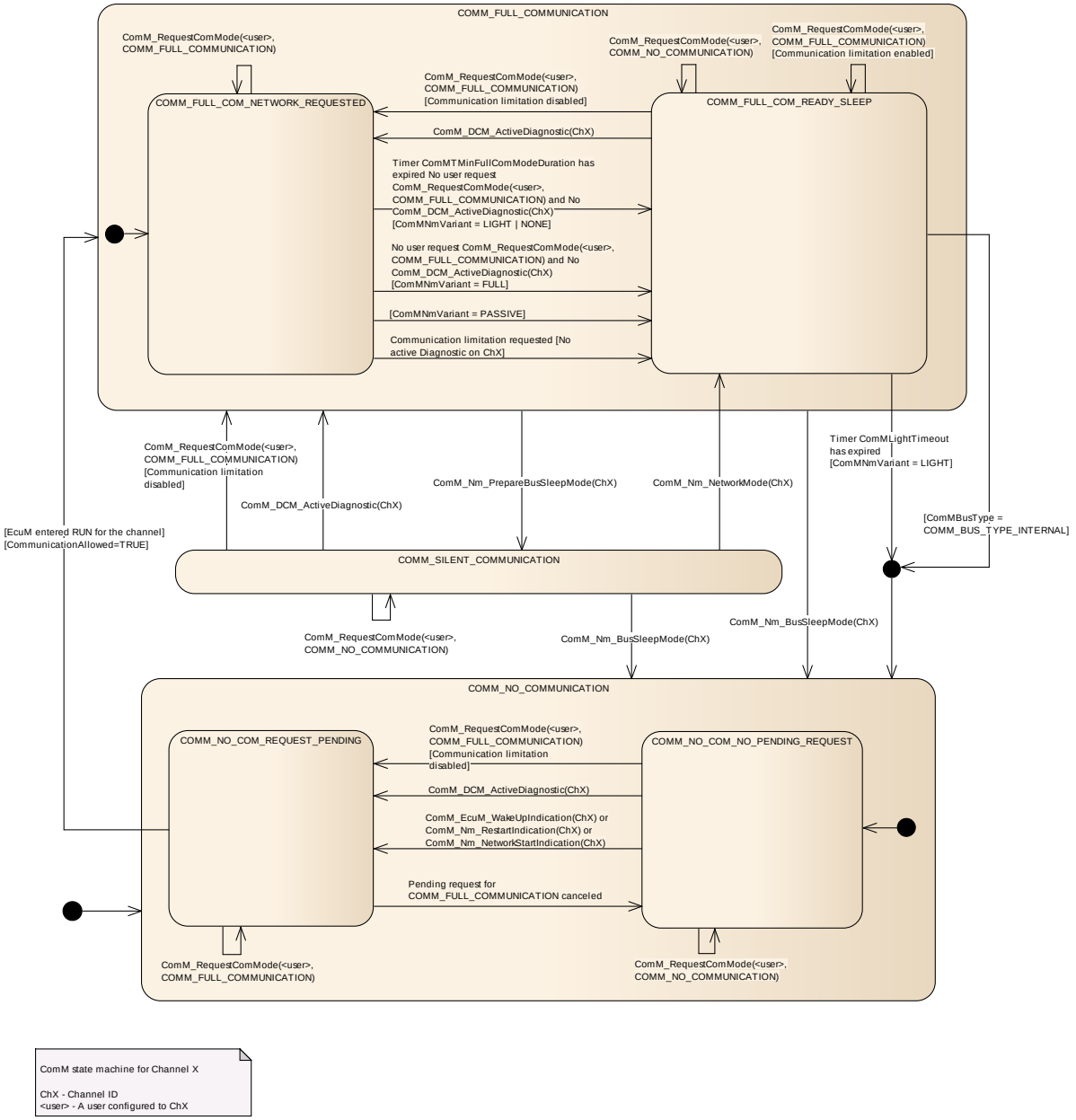

Figure 4: ComM channel state machine

State	Section / Requirement
COMM_NO_COMMUNICATION	7.2.1 Entering state: SWS_ComM_00898, SWS_ComM_00313, SWS_ComM_00073, SWS_ComM_00288 In sub-state COMM_NO_COM_NO_PENDING_REQUEST: SWS_ComM_00875, SWS_ComM_00876, SWS_ComM_00893, SWS_ComM_00894, SWS_ComM_00694 In sub-state COMM_NO_COM_REQUEST_PENDING: SWS_ComM_00895 ,SWS_ComM_00897 ComM128
COMM_SILENT_COMMUNICATION	7.2.2

	Entering state: SWS_ComM_00071 In state: SWS_ComM_00877, SWS_ComM_00878 SWS_ComM_00295, SWS_ComM_00296
COMM_FULL_COMMUNICATION	7.2.3 Entering state: SWS_ComM_00069 In state: SWS_ComM_00637, [SWS_ComM_00826] 7.1.3.1 sub-state COMM_FULL_COM_NETWORK_REQUESTED: In sub-state: SWS_ComM_00869, SWS_ComM_00870, SWS_ComM_00665, SWS_ComM_00888, SWS_ComM_00889, SWS_ComM_00890 7.1.3.2 sub-state COMM_FULL_COM_READY_SLEEP Entering sub-state: SWS_ComM_00133 In sub-state: SWS_ComM_00610, SWS_ComM_00671, SWS_ComM_00882, SWS_ComM_00883
Transition	**Requirement**
COMM_NO_COMMUNICATION → COMM_FULL_COMMUNICATION	SWS_ComM_00893, SWS_ComM_00894, SWS_ComM_00694, SWS_ComM_00875 SWS_ComM_00876,
COMM_FULL_COM_NETWORK_REQUESTED → COMM_FULL_COM_READY_SLEEP	SWS_ComM_00665
COMM_FULL_COM_READY_SLEEP → COMM_FULL_COM_NETWORK_REQUESTED	SWS_ComM_00882, SWS_ComM_00883
COMM_FULL_ COMMUNICATION → COMM_SILENT_COMMUNICATION	SWS_ComM_00826
COMM_FULL_COM_READY_SLEEP → COMM_NO_COMMUNICATION	SWS_ComM_00610, SWS_ComM_00671
COMM_FULL_COMMUNICATION → COMM_NO_COMMUNICATION	SWS_ComM_00637
COMM_SILENT_COMMUNICATION → COMM_FULL_COMMUNICATION	SWS_ComM_00877, SWS_ComM_00878
COMM_SILENT_COMMUNICATION → COMM_FULL_COM_READY_SLEEP	SWS_ComM_00296
COMM_SILENT_COMMUNICATION → COMM_NO_COMMUNICATION	SWS_ComM_00295

Table 1: Link to detailed explanation of the channel state machine resp. transition

[SWS_ComM_00879] ⌈The ComM channel state machine shall consist of the three main states corresponding to the Communication Modes: `COMM_NO_COMMUNICATION`, `COMM_SILENT_COMMUNICATION` and `COMM_FULL_COMMUNICATION`. ⌋()

[SWS_ComM_00880] ⌈The COMM_FULL_COMMUNICATION state shall have two sub-states COMM_FULL_COM_NETWORK_REQUESTED and COMM_FULL_COM_READY_SLEEP. ⌋()

[SWS_ComM_00881] ⌈The COMM_NO_COMMUNICATION state shall have two sub-states COMM_NO_COM_REQUEST_PENDING and COMM_NO_COM_NO_PENDING_REQUEST⌋()

Rationale for SWS_ComM_00879 and SWS_ComM_00880:
`COMM_FULL_COM_READY_SLEEP` and `COMM_SILENT_COMMUNICATION` are necessary to synchronize a communication shutdown on the bus. If only one ECU switches the communication off, the others store errors because this ECU stops sending application signals.

Comment: The main states present an abstracted status of communication capabilities per channel, which are in focus of the users' interests. The sub-states represent intermediate states, which perform activities to support a synchronized transition with external partners and managing protocols (e.g. NM)

[SWS_ComM_00485] ⌈The default state for each ComM channel state machine shall be `COMM_NO_COMMUNICATION`.⌋()

[SWS_ComM_00896] ⌈Each ComM channel state machine shall only evaluate its corresponding communication status flag `CommunicationAllowed`

according to SWS_ComM_00884 in sub-state COMM_NO_COM_REQUEST_PENDING.」()

Rationale for SWS_ComM_00896: A ComM_CommunicationAllowed(<channel>,FALSE) (SWS_ComM_00871) indication has no visible effect if the channel is not in sub-state COMM_NO_COM_REQUEST_PENDING, i.e. ComM channel state machine will not immediately change to state COMM_NO_COMMUNICATION if in another state as e.g. COMM_FULL_COMMUNICATION

[SWS_ComM_00472] 「Main state changes (see SWS_ComM_00879) shall be indicated to the users with the corresponding notifications (see section 8.6.1.5 and 8.6.1.6). Exception: Default state after initialization, see SWS_ComM_00313.」()

Comment: If more than one user is related to the corresponding channel state machine, the ComM module has to perform a Fan-out to all users.

[SWS_ComM_00191] 「The internal functionality of the ComM channel state machine(s) shall be invisible for the users. The user neither needs nor shall get any information about the internal mechanisms and rules (e.g. "highest wins" strategy) of the ComM channel state machine.」()

An overview of the requested communication capabilities in the Corresponding Mode is shown in Table 2.

Communication Mode	Message Transmission	Message Reception	NM (COMM_NM_VARIANT=FULL)	Wake-up/Restart capability
COMM_FULL_COMMUNICATION	On	On	Bus communication requested	N/A
COMM_SILENT_COMMUNICATION	Off	On	Bus communication released	• User/diagnostic request • Network indication
COMM_NO_COMMUNICATION	Off	Off	Bus communication released	• User/diagnostic request • Passive wake-up

Table 2: Granted communication capabilities in the corresponding modes

Note for section 7.1.1 - 7.1.3: *Each ComM channel state machine is responsible to handle one channel/network with a connected Bus State Manager ("corresponding" = the channel/network the ComM channel state machine is responsible for).*

Note for section 7.1.1 - 7.1.3: *The ComM module contains one or several ComM channel state machine(s). ComM channel state machine communicates directly with its connected Bus State Manager, other interfaces are handled by the ComM module.*

7.2.1 Behavior in state `COMM_NO_COMMUNICATION`

[SWS_ComM_00898] ⌈On entering state `COMM_NO_COMMUNICATION` the ComM channel state machine shall go to sub-state `COMM_NO_COM_NO_PENDING_REQUEST`.⌋()

[SWS_ComM_00313] ⌈On entering state `COMM_NO_COMMUNICATION` by default after initialization, ComM module shall not indicate the mode change to users via RTE or BswM.⌋()

Rationale for SWS_ComM_00313: The RTE is not yet initialized at this point in time.

[SWS_ComM_00073] ⌈On entering state `COMM_NO_COMMUNICATION` the ComM channel state machine shall switch off the transmission and reception capability. This shall be performed by the ComM channel state machine requesting the corresponding Communication Mode from the Bus State Manager module (`XXSM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_NO_COMMUNICATION`, see SWS_ComM_00829)).⌋()

Rationale for SWS_ComM_00073: The `COMM_NO_COMMUNICATION` mode forbids sending and receiving of bus communication PDUs for the corresponding channels.

Rationale for ***SWS_ComM_00073***, ***SWS_ComM_00875*** and ***SWS_ComM_00876***: FlexRay shutdown cannot be interrupted to avoid partial networks.

[SWS_ComM_00288] ⌈On entering state COMM_NO_COMMUNICATION and configuration parameter ComMNmVariant=FULL (see ECUC_ComM_00568) the ComM module shall request release of the network from the Network Management module, Nm_NetworkRelease().⌋()

Comment: In state COMM_NO_COMMUNICATION ComM channel state machine may not request bus communication for the configured channel from the Bus State Manager module.

Use Case for above Comment: The ECU is performing control functions locally without participation in bus communication.

Comment: The communication mode is local for one channel, thus the ECU may still communicate via other channels.

7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state

[SWS_ComM_00875] ⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST and user requests COMM_FULL_COMMUNICATION and communication limitation is disabled (see Section 7.3.2), the ComM channel state machine shall immediately switch to sub-state COMM_NO_COM_REQUEST_PENDING.⌋()

[SWS_ComM_00876] ⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST ,configuration parameter ComMNmVariant=FULL|LIGHT|NONE (ECUC_ComM_00568) and DCM indicate ComM_DCM_ActiveDiagnostic(SWS_ComM_00873), the ComM channel state machine shall immediately switch to sub-state COMM_NO_COM_REQUEST_PENDING.⌋()

Rationale for SWS_ComM_00876: A potential communication limitation (see Section 7.3.2) shall temporarily be inactive during an active diagnostic session, see SWS_ComM_00182

[SWS_ComM_00893] ⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST and a wake-up-indication is indicated by the EcuM module, ComM_EcuM_WakeUpIndication() SWS_ComM_00275, the ComM channel state machine shall immediately switch to sub-state COMM_NO_COM_REQUEST_PENDING.⌋(SRS_ModeMgm_09087)

[SWS_ComM_00894] ⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST and the NM module indicates a restart, ComM_Nm_RestartIndication() SWS_ComM_00792, the ComM channel state machine shall immediately switch to sub-state COMM_NO_COM_REQUEST_PENDING. ⌋(SRS_ModeMgm_09087)

Rationale for SWS_ComM_00893 and SWS_ComM_00894: It must be guaranteed that communication starts as soon as possible after a bus wake up.

Comment: The ComM channel state machine switches immediately to sub-state COMM_FULL_COM_NETWORK_REQUESTED after entering the COMM_FULL_COMMUNICATION state. If no user requests COMM_FULL_COMMUNICATION mode, the AUTOSAR NM resp. the ComM module timer for ComMTMinFullComModeDuration (ECUC_ComM_00557) prevent toggling between COMM_NO_COMMUNICATION and COMM_FULL_COMMUNICATION to overcome the init-/start-up time of the system, before possible user requests occur.

[SWS_ComM_00694] ⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.⌋()

7.2.1.2 COMM_NO_COM_REQUEST_PENDING sub-state

[SWS_ComM_00895] ⌈In sub-state COMM_NO_COM_REQUEST_PENDING the ComM channel state machine shall evaluate its corresponding CommunicationAllowed

flag, stored and set according to SWS_ComM_00884 and SWS_ComM_00885. If evaluated to `CommunicationAllowed=TRUE`, the ComM channel state machine shall immediately switch to state `COMM_FULL_COMMUNICATION`.」()

[SWS_ComM_00897] 「In sub-state `COMM_NO_COM_REQUEST_PENDING` and no longer any valid pending request for `COMM_FULL_COMMUNICATION`, the ComM channel state machine shall switch back to default sub-state `COMM_NO_COM_NO_PENDING_REQUEST`.」()

Rationale for SWS_ComM_00897: The possibility to switch back to default sub-state if communication for some reason was never allowed. E.g. transition to `COMM_NO_COM_REQUEST_PENDING` triggered by user request for `ComM_RequestComMode(<user>,COMM_FULL_COMMUNICATION)` (SWS_ComM_00871) or DCM indicated `ComM_DCM_ActiveDiagnostic(<channel>)` (SWS_ComM_00873), but now canceled with `ComM_RequestComMode(<user>,COMM_NO_COMMUNICATION)` (SWS_ComM_00871) or DCM `ComM_DCM_InactiveDiagnostic(<channel>)` (SWS_ComM_00874).

Comment: EcuM –Fixed shall read and evaluate ComM channel state machine sub-states, with `ComM_GetState()` (SWS_ComM_00872) before a sleep/shutdown.

7.2.2 Behaviour in state `COMM_SILENT_COMMUNICATION`

[SWS_ComM_00071] 「On entering state `COMM_SILENT_COMMUNICATION` the ComM channel state machine shall switch off the transmission capability (and keep reception capability on). This shall be performed by the ComM channel state machine requesting the corresponding Communication Mode from the Bus State Manager module (`XXSM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_SILENT_COMMUNICATION)` SWS_ComM_00829).」()

Rationale for SWS_ComM_00071: The `COMM_SILENT_COMMUNICATION` mode permits receiving of bus communication PDUs and forbids sending of bus communication PDUs.

Comment: It may happen that nothing is received (e.g. during bus off) despite receiving capability is switched on.

Use Case: Shut down coordination with means of the NM module (prepare bus sleep state).

[SWS_ComM_00877] ⌈In state COMM_SILENT_COMMUNICATION and user requests COMM_FULL_COMMUNICATION and communication limitation is disabled (see Section 7.3.2), the ComM channel state machine shall switch to state COMM_FULL_COMMUNICATION.⌋()

[SWS_ComM_00878] ⌈In state COMM_SILENT_COMMUNICATION ,configuration parameter ComMNmVariant=FULL|LIGHT|NONE (ECUC_ComM_00568) and DCM indicate ComM_DCM_ActiveDiagnostic(SWS_ComM_00873), the ComM channel state machine shall switch to state COMM_FULL_COMMUNICATION.⌋()

Rationale for SWS_ComM_00878: A potential communication limitation (see Section 7.3.2) shall temporarily be inactive during an active diagnostic session, see SWS_ComM_00182

[SWS_ComM_00295] ⌈In state COMM_SILENT_COMMUNICATION and the Network Manager module indicates ComM_Nm_BusSleepMode() (SWS_ComM_00392), the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION.⌋()

[SWS_ComM_00296] ⌈In state COMM_SILENT_COMMUNICATION and the Network Manager module indicates See ComM_Nm_NetworkMode() SWS_ComM_00390, the ComM channel state machine shall switch to state COMM_FULL_COMMUNICATION and sub-state COMM_FULL_COM_READY_SLEEP.⌋()

7.2.3 Behaviour in state COMM_FULL_COMMUNICATION

[SWS_ComM_00899] ⌈On entering state COMM_FULL_COMMUNICATION the ComM channel state machine shall go to sub-state COMM_FULL_COM_NETWORK_REQUESTED, if not a specific sub-state is specified in the transition.⌋()

Rationale for SWS_ComM_00899: When switching from COMM_SILENT_COMMUNICATION, the ComM channel state machine can switch directly to sub-state COMM_FULL_COM_READY_SLEEP, if specified in the transition, see SWS_ComM_00296.

[SWS_ComM_00069] ⌈On entering state COMM_FULL_COMMUNICATION the ComM channel state machine shall switch on the transmission and reception capability. This shall be performed by the ComM channel state machine requesting the corresponding Communication Mode from the Bus State Manager module (XXSM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_FULL_COMMUNICATION) SWS_ComM_00829).⌋()

Rationale for SWS_ComM_00069: The COMM_FULL_COMMUNICATION mode permits sending and receiving of bus communication PDUs for the corresponding channels.

[SWS_ComM_00637] ⌈In state COMM_FULL_COMMUNICATION and the Network Manager module indicates ComM_Nm_BusSleepMode() SWS_ComM_00392, the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION.⌋()

Rationale for SWS_ComM_00637: A user may request to keep the bus awake "too late" (NM is not able to send a vote to keep the bus awake because the cluster already agreed to shutdown).

[SWS_ComM_00826]⌈ In COMM_FULL_COMMUNICATION and configuration parameter ComMNmVariant=FULL|PASSIVE (ECUC_ComM_00568) and the Network Manager module indicates

ComM_Nm_PrepareBusSleepMode() (SWS_ComM_00391), the ComM state machine shall switch to state COMM_SILENT_COMMUNICATION.」()

Rationale for SWS_ComM_00826: ComM_Nm_PrepareBusSleepMode() cannot be received before an active request is released via Nm_NetworkRelease(), and a PASSIVE channel cannot be woken up by an active wake-up, therefore it is safe to assume that the transition is always valid.

7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state

[SWS_ComM_00886] 「On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=LIGHT|NONE (ECUC_ComM_00568), the timer for ComMTMinFullComModeDuration (ECUC_ComM_00557) shall be started.」()

[SWS_ComM_00665] 「On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED and EcuM module has indicated a wake-up, ComM_EcuM_WakeUpIndication(<channel>) (SWS_ComM_00275), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management.」()

[SWS_ComM_00902] 「On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED and Nm module has indicated a restart, ComM_Nm_RestartIndication(<channel>) (SWS_ComM_00792), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management」()

[SWS_ComM_00903] 「On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED and Nm module has indicated a Network start, ComM_Nm_NetworkStartIndication(<channel>) (SWS_ComM_00383), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management」()

Comment for SWS_ComM_00903: This is not a “normal” transition to COMM_FULL_COMMUNICATION, ComM handle ComM_Nm_NetworkStartIndication() as “race condition” error, see section 7.6.1

[SWS_ComM_00869] 「On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED from another state or substate, if configuration parameter ComMNmVariant=FULL (ECUC_ComM_00568)

and if a user has requested ComM_RequestComMode(<user>,COMM_FULL_COMMUNICATION) (SWS_ComM_00110) the ComM module shall request Nm_NetworkRequest(<channel>) from the Network Management for the corresponding NM channel.」(SRS_ModeMgm_00049)

Note: Additionally Nm_NetworkRequest may be invoked due to SWS_ComM_00980.

[SWS_ComM_00870] 「On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED, if configuration parameter ComMNmVariant=FULL (ECUC_ComM_00568) and the DCM has indicated ComM_DCM_ActiveDiagnostic(<channel>) (SWS_ComM_00873), the ComM module shall request Nm_NetworkRequest(<channel>) from the Network Management for the corresponding NM channel.」(SRS_ModeMgm_00049)

[SWS_ComM_00889] 「In sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=LIGHT|NONE (ECUC_ComM_00568) and timer for ComMTMinFullComModeDuration (ECUC_ComM_00557) has expired and no user request ComM_RequestComMode(<user>,COMM_FULL_COMMUNICATION) and the DCM does not indicate ComM_DCM_ActiveDiagnostic(<channel>) (SWS_ComM_00873), the ComM channel state machine shall switch to sub-state COMM_FULL_COM_READY_SLEEP.」()

Rationale for SWS_ComM_00889: As long as timer for ComMTMinFullComModeDuration has not expired the sub-state shall be kept, to prevent toggling.

[SWS_ComM_00888] 「In sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=FULL (ECUC_ComM_00568)

and no user request `ComM_RequestComMode(<user>,COMM_FULL_COMMUNICATION)` and the DCM does not indicate `ComM_DCM_ActiveDiagnostic(<channel>)` (SWS_ComM_00873), the ComM channel state machine shall switch to sub-state `COMM_FULL_COM_READY_SLEEP`.」()

Rationale for SWS_ComM_00888: No timer needed if AUTOSAR NM is used. This avoids redundant functionality because AUTOSAR NM also ensures this functionality

[SWS_ComM_00915]「 In sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=PASSIVE (ECUC_ComM_00568), the ComM channel state machine shall switch to sub-state COMM_FULL_COM_READY_SLEEP.」()

[SWS_ComM_00890] 「In sub-state `COMM_FULL_COM_NETWORK_REQUESTED` and the DCM does not indicate `ComM_DCM_ActiveDiagnostic(<channel>)` (SWS_ComM_00873) and communication limitation is requested (see section 7.3.2), ComM channel state machine shall immediately switch to sub-state `COMM_FULL_COM_READY_SLEEP`.」()

7.2.3.2 `COMM_FULL_COM_READY_SLEEP` sub-state

[SWS_ComM_00133]「On entering sub-state `COMM_FULL_COM_READY_SLEEP` and configuration parameter `ComMNmVariant=FULL` (ECUC_ComM_00568), the ComM module shall request `Nm_NetworkRelease()` from the Network Management for the corresponding NM channels.」()

[SWS_ComM_00891] 「On entering sub-state `COMM_FULL_COM_READY_SLEEP` and configuration parameter `ComMNmVariant=LIGHT` (ECUC_ComM_00568), the timer for `ComMNmLightTimeout` (ECUC_ComM_00606) shall be started.」()

[SWS_ComM_00610] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMNmVariant=LIGHT (ECUC_ComM_00568) and the timer for ComMNmLightTimeout (ECUC_ComM_00606) has expired, the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION.⌋()

[SWS_ComM_00671] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMBusType=COMM_BUS_TYPE_INTERNAL (ECUC_ComM_00567), the ComM channel state machine shall immediately switch to state COMM_NO_COMMUNICATION.⌋()

[SWS_ComM_00882] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and a user request COMM_FULL_COMMUNICATION and communication limitation is disabled (see Section 7.3.2), the ComM channel state machine shall immediately switch to sub-state COMM_FULL_COM_NETWORK_REQUESTED.⌋()

[SWS_ComM_00883] ⌈In sub-state COMM_FULL_COM_READY_SLEEP ,configuration parameter ComMNmVariant=FULL|LIGHT|NONE (ECUC_ComM_00568) and DCM indicate ComM_DCM_ActiveDiagnostic(SWS_ComM_00873), the ComM channel state machine shall switch to sub-state COMM_FULL_COM_NETWORK_REQUESTED.⌋()

Rationale for SWS_ComM_00883: A potential communication limitation (see Section 7.3.2) shall temporarily be inactive during an active diagnostic session, see SWS_ComM_00182

[SWS_ComM_00892] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMNmVariant=LIGHT (ECUC_ComM_00568) and a switch to sub-state COMM_FULL_COM_NETWORK_REQUESTED, due to

request for COMM_FULL_COMMUNICATION according to requirements in SWS_ComM_00882 or SWS_ComM_00883, the timer for ComMNmLightTimeout (ECUC_ComM_00606) shall be canceled.」()

7.3 Extended functionality

[SWS_ComM_00470] 「The extended functionality described in this chapter shall be individually configurable during runtime per feature (e.g. enable wake up inhibition but disable limitation to no communication).」()

Rationale for SWS_ComM_00470: During runtime a change in the inhibition / limitation
strategy is required in order to cope with changing conditions.

Use Case: Change the wakeup inhibition via diagnostics.

Comment: Configurable with parameter ComMEcuGroupClassification (see ECUC_ComM_00563).

7.3.1 State duration extensions

Comment: Obsolete section and can be removed. Requirement for "state duration extension" for NM variant LIGHT and NONE, moved to state-machine section.

7.3.2 Communication inhibition

Note: The purpose of mode inhibition is to limit the communication capabilities. For details see Section 7.3.2.1 and Section 7.3.2.2.

[SWS_ComM_00301] 「The ComM module shall offer interfaces to request and release the corresponding mode inhibitions.」()

Comment: The ComM module doesn't care about who requests the mode inhibition but it is not a "normal" SW-C. It is a privileged SW-C or an OEM specific BSW.

[SWS_ComM_00488] ⌈It shall be possible to enable and disable the mode inhibition for each channel (channel state machine) independently. This functionality shall not be used by the ComM module itself.⌋()

[SWS_ComM_00839] ⌈The ComM module shall store the status of the user requests.⌋()

Comment: SWS_ComM_00839 describes the desired behaviour during an active mode limitation.

[SWS_ComM_00840] ⌈The ComM module shall store the updated status of the user requests if a user releases a request during an active mode inhibition.⌋()

Rationale for SWS_ComM_00840: User requests shall be granted if the inhibition gets disabled.

Comment: Amount of active user requests from different users. SWS_ComM_00840 describes the desired behaviour during an active mode limitation.

[SWS_ComM_00182] ⌈The communication inhibition shall get temporarily inactive during an active diagnostic session.⌋()

Rationale for SWS_ComM_00182: ECUs must not fall asleep during an active diagnostic session.

Comment: The DCM indicates the start of an active diagnostic session with `ComM_DCM_ActiveDiagnostic(<channel>)` (SWS_ComM_00873) and the end of a diagnostic session with `ComM_DCM_ActiveDiagnostic(<channel>)` (SWS_ComM_00874).

7.3.2.1 Bus wake up inhibition

Information: Bus wake up inhibition in context of the ComM module means that the ComM module should take precautions against awaking other ECUs by starting the communication.

Rationale: Awaking other ECUs by communication should be avoided because it is assumed that the ECU wakes up the bus because of an error (e.g. broken sensor).

Use Case: An error was detected on signal path of an active wake up line and this non reliable wake-up-source should not be able to awake the whole system anymore. An SW-C that controls error-reactions could set the wake up inhibition-status of related communication channels that usually get communication-requests from SW-Cs as the consequence of this event. This corrupts the forwarding of communication system-wide, based on unreliable wake up events. Or in case of application-specific system control, there is an SW-C that should switch off forwarding system wide wakeup's by communication under conditions like e.g. transport mode.

[SWS_ComM_00302] ⌈Bus wake up Inhibition shall be performed by ignoring user requests.⌋(SRS_ModeMgm_09089)

Comment: Ignoring user requests means accepting the requests but not executing them due to mode inhibition. The “highest win” strategy would apply immediately as soon as mode inhibition is switched off (see SWS_ComM_00839 and SWS_ComM_00840).

[SWS_ComM_00218] ⌈A communication request (`COMM_FULL_COMMUNICATION`) by a user shall be inhibited if the ComM Inhibition status is equal to `ComMNoWakeup=TRUE` (ECUC_ComM_00569) for the corresponding channel and the current state of the channel is `COMM_NO_COMMUNICATION` or `COMM_SILENT_COMMUNICATION` ⌋()

Rationale for SWS_ComM_00218: The inhibition should not get active, if the inhibition-status is set but the communication channel is already active.

[SWS_ComM_00219] ⌈The inhibition shall not get active if the current communication state is `COMM_FULL_COMMUNICATION` .⌋()

Rationale for SWS_ComM_00219: The bus is already awake if the current communication state is COMM_FULL_COMMUNICATION.

[SWS_ComM_00066] ⌈The ComM module shall never inhibit the “passive wake-up” capability.⌋()

Rationale for SWS_ComM_00066: It must be always possible to react on bus wake ups indicated by the EcuM module.

Comment: Reception is switched off in COMM_NO_COMMUNICATION mode but the wake up capability is switched on.

[SWS_ComM_00157] ⌈ComMNoWakeup status must be stored non volatile.⌋()

Rationale for SWS_ComM_00157: Information must be available during start-up, before the communication is active (“Full Communication” mode entered). Changing or query is only possible after start-up with active communication (usually the "master", who decides if the inhibition is active or not, is not on the same ECU).

[SWS_ComM_00625] ⌈The status of the user requests shall also be updated if a user releases a request.⌋()

7.3.2.2 Limit to COMM_NO_COMMUNICATION mode

[SWS_ComM_00303] ⌈The ComM module shall perform the limit to COMM_NO_COMMUNICATION mode by switching to COMM_FULL_COM_READY_SLEEP state to initiate a shutdown despite user requests for COMM_FULL_COMMUNICATION mode and ignoring new COMM_FULL_COMMUNICATION mode requests.⌋(SRS_ModeMgm_09071)

Rationale for SWS_ComM_00303: Forcing into COMM_NO_COMMUNICATION mode is needed to shut down software components, which keeps the bus awake.

[SWS_ComM_00355]⌈ ComM shall force an ECU reset by invoking BswM_ComM_InitiateReset() after entering "No Communication" mode if configured (ComMResetAfterForcingNoComm=TRUE).⌋()

Rationale: It is assumed that a faulty user will not release his "Full Communication" request without a re-initialization. Keeping the "Full Communication" request active leads to a toggling between network shutdown and network startup.

Use Case: It is assumed that a faulty ECU keeps the bus awake. As a consequence a "network master" decides to force all ECUs to go to sleep.

[SWS_ComM_00841] ⌈The ComM module shall only perform the limit to `COMM_NO_COMMUNICATION` mode if the current state is `COMM_FULL_COM_NETWORK_REQUESTED`.⌋()

[SWS_ComM_00842] ⌈The ComM module shall ignore requests in other states than `COMM_FULL_COM_NETWORK_REQUESTED`.⌋()

[SWS_ComM_00215] ⌈All active user requests for communication channel X shall be ignored if the ComM Inhibition `ComMNoCom=TRUE` (see ComM561_ConfECUC_ComM_00571) for the corresponding channel to guarantee entering the `COMM_NO_COMMUNICATION` state for channel X.⌋()

[SWS_ComM_00582] ⌈The ComM module shall clear the user requests after all the channels that belong to the corresponding user enter `COMM_NO_COMMUNICATION` mode.⌋()

Rationale for SWS_ComM_00582: Stored (faulty) user requests, which are assumed to keep the bus awake, must be cleared.

Description: The ComM module shall reload the default value of the ComM inhibition status from `ComMNoCom` (see ECUC_ComM_00571) during initialization.

Comment: The current ComMNoCom status for each channel shall not be stored persistently. SWS_ComM_00582 describes the desired behaviour after an executed mode limitation.

7.4 Bus communication management

[SWS_ComM_00402] ⌈The ComM module shall use the corresponding interfaces of the Bus State Manager modules to control the communication capabilities.⌋()

[SWS_ComM_00664] ⌈The ComM module shall omit calls to control the communication capabilities if configuration parameter `ComMBusType=COMM_BUS_TYPE_INTERNAL` (ECUC_ComM_00567).⌋(SRS_ModeMgm_09168)

Rationale for SWS_ComM_00664: Internal communication has no corresponding bus interface.

7.5 Network management dependencies

[SWS_ComM_00599] ⌈The ComM module shall support the shutdown synchronization variants (configured with `ComMNmVariant`, see ECUC_ComM_00568) `LIGHT`, `PASSIVE` and `FULL` described in Table 3.⌋()

Comment: Only variant `FULL` and `PASSIVE` guarantees a synchronized shutdown between all nodes of a network. Note that since the NmIf cannot start the synchronized shutdown of coordinated networks before all networks are ready to go to sleep, requests from ComM to NmIf to release network communication on such a coordinated bus will be considered, but not always acted on directly. The NmIf will still answer with `NM_E_OK`, but network will not be released until all coordinated networks are ready to go to sleep.

NM variant	Keep bus awake capability	Shutdown synchronization
`NONE`		No shutdown synchronization by ComM. Shutdown by switching off the power of the ECU.
`LIGHT`		Shutdown synchronization by ComM with

		means of a timeout (configured with ComMNmLightTimeout, ECUC_ComM_00606)
PASSIVE	ECU is not allowed to keep the bus awake	Shutdown synchronization by ComM with means of AUTOSAR NM.
FULL	ECU is allowed to keep the bus awake.	Shutdown synchronization by ComM with means of AUTOSAR NM.

Table 3: Network management variants supported by the Communication Manager Module

Comment: A synchronized shutdown is not possible with the LIGHT variant thus the ECU may continuously restart ("toggle") because of a message from a node shutting down later.

[SWS_ComM_00501]⌈ If ComMNmPassiveModeEnable is set to ENABLED the parameter ComMNmVariant shall be limited to the values LIGHT, NONE, and PASSIVE.⌋()

[SWS_ComM_00502]⌈ If ComMNmPassiveModeEnable is set to DISABLED the ComMNmVariant shall be limited to the values LIGHT, NONE, and FULL.⌋()

[SWS_ComM_00602] ⌈The ComM module shall omit calls of NM services if configuration parameter ComMNmVariant=LIGHT|NONE (see ECUC_ComM_00568).⌋()

Rationale for SWS_ComM_00602: NM services are not available if no NM is available.

[SWS_ComM_00667] ⌈The ComM module shall omit to call Nm_NetworkRequest() from NM if configuration parameter ComMNmVariant=PASSIVE (see ECUC_ComM_00568).⌋()

Rationale for *SWS_ComM_00667*: Service Nm_NetworkRequest() is not available.

7.6 Bus error management

7.6.1 Network Start Indication

[SWS_ComM_00583] ⌈The ComM module shall switch channel X to `COMM_FULL_COMMUNICATION` if NM indicates `ComM_Nm_NetworkStartIndication(<channel X>)` and CommunicationAllowed flag is set to TRUE.⌋()

Use Case for SWS_ComM_00583: A node sends an NM message in "Prepare Bus Sleep" state but other nodes are already in "Bus Sleep" state because of "race conditions".

7.7 Test support requirements

7.7.1 Inhibited Full Communication Request Counter

[SWS_ComM_00138] ⌈The ComM module shall provide one Inhibit counter for all rejected `COMM_FULL_COMMUNICATION` mode requests. It shall count user requests, which cannot be fulfilled because the system has inhibited communication modes.⌋(SRS_ModeMgm_09155)

Rationale for SWS_ComM_00138: The counter is used for detecting latent software problems related to unmotivated communication bus wake ups.

[SWS_ComM_00140] ⌈The Inhibit counter (SWS_ComM_00138) for all rejected `COMM_FULL_COMMUNICATION` mode requests shall be stored in non-volatile memory.⌋()

[SWS_ComM_00141] ⌈The range of the Inhibit counter (SWS_ComM_00138) for all rejected `COMM_FULL_COMMUNICATION` mode requests shall be 0 to 65535.⌋()

[SWS_ComM_00142] ⌈The Inhibit counter (SWS_ComM_00138) for all rejected COMM_FULL_COMMUNICATION mode requests shall stop to increment if the maximum counter value is reached.⌋()

[SWS_ComM_00143] ⌈It shall be possible to read out and reset the Inhibit counter (SWS_ComM_00138) for all rejected COMM_FULL_COMMUNICATION mode requests value by a ComM module API call.⌋()

Use Case for SWS_ComM_00143: It shall be possible to read out and reset the current status of the counter by a diagnostic service.

7.8 Error classification

[SWS_ComM_00234] ⌈The ComM module shall use the error codes of table 4 to report errors.

Type or error	*Relevance*	*Related error code*	*Value [hex]*
API service used without module initialization	Development	COMM_E_NOT_INITED	0x1
API service used with wrong parameters (e.g. a NULL pointer)	Development	COMM_E_WRONG_PARAMETERS	0x2

Table 4: Error classification⌋(SRS_BSW_00323, SRS_BSW_00327, SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386)

[SWS_ComM_00612] ⌈If ComM is not initialized, all ComM module all API service other than ComM_Init() (see SWS_ComM_00146), ComM_GetVersionInfo() (see SWS_COMM_00370) and ComM_GetStatus() (see SWS_COMM_00242); shall:

- not execute their normal operation,
- and return E_NOT_OK, if it has a standard return type.⌋(SRS_BSW_00406)

[SWS_ComM_00858] ⌈If ComM is not initialized and development error detection has been switched on by ComMDevErrorDetect (see ECUC_ComM_00555), the ComM module shall report a development error COMM_E_NOT_INITED (by using the Det_ReportError service of the Development Error Tracer module) for all ComM module API services other than ComM_Init() and ComM_GetVersionInfo(), and ComM_GetStatus().⌋(SRS_BSW_00406)

7.9 Non functional requirements

[SWS_ComM_00459] ⌈It shall be possible to integrate the ComM module delivered as source or object code into the AUTOSAR stack.

Rationale:

- Allow IP protection and guaranteed test coverage: object code
- Allow high efficiency and configurability at system generation time (by integrator): source code.⌋(SRS_BSW_00342)

7.10 Communication Manager Module Services

This section defines the AUTOSAR Interfaces of the Communication Manager Module Service (ComM).

7.10.1 Architecture

The overall architecture of the Communication Manager Module service is depicted in Figure 5:

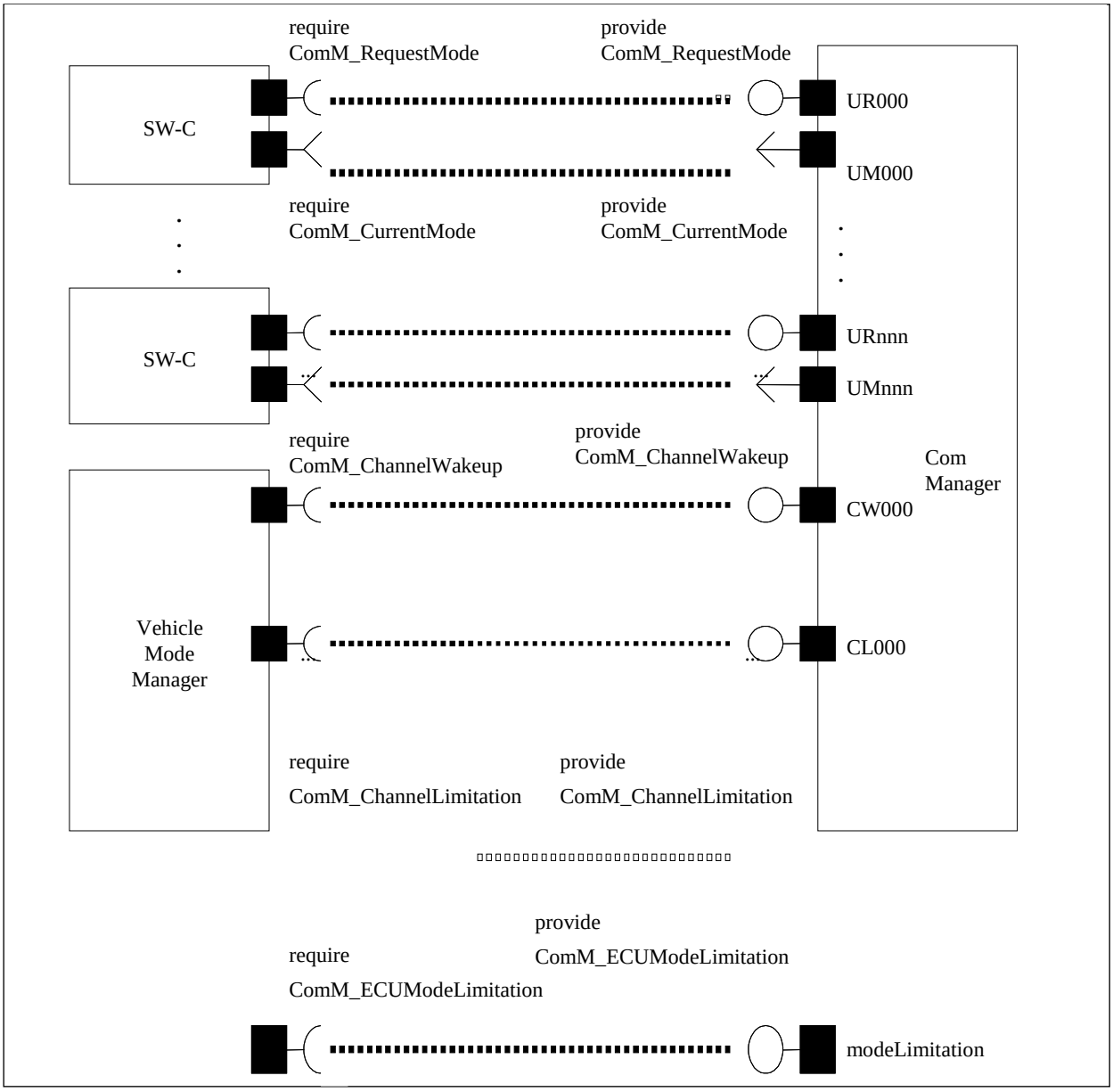

Figure 5: ARPackage of the Communication Manager Module

7.10.2 Use Cases

7.10.2.1 SW-Cs does not care about the ComM module at all

A SW-C that does not care about the Communication Manager Module will not require any of the interfaces defined in the ARPackage of the Communication Manager Module.

7.10.2.2 SW-Cs only cares about the state of its communication system

In this use case, a SW-C wants to know what communication capabilities it has (expressed by a communication mode ‘none’, ‘silent’ or ‘full’ - see ComM_ModeType). The SW-C finds out about that by defining a port requiring the Interface ComM_GetCurrentComMode. Depending on the available communication capabilities, the SW-C can specify that certain runnables of the SW-C should be executed or not. The Communication Manager Module must be configured correctly

(with e.g. the physical channels that this SW-C uses for its logical communication) such that it has a port that provides this information about the current communication mode to the SW-C.

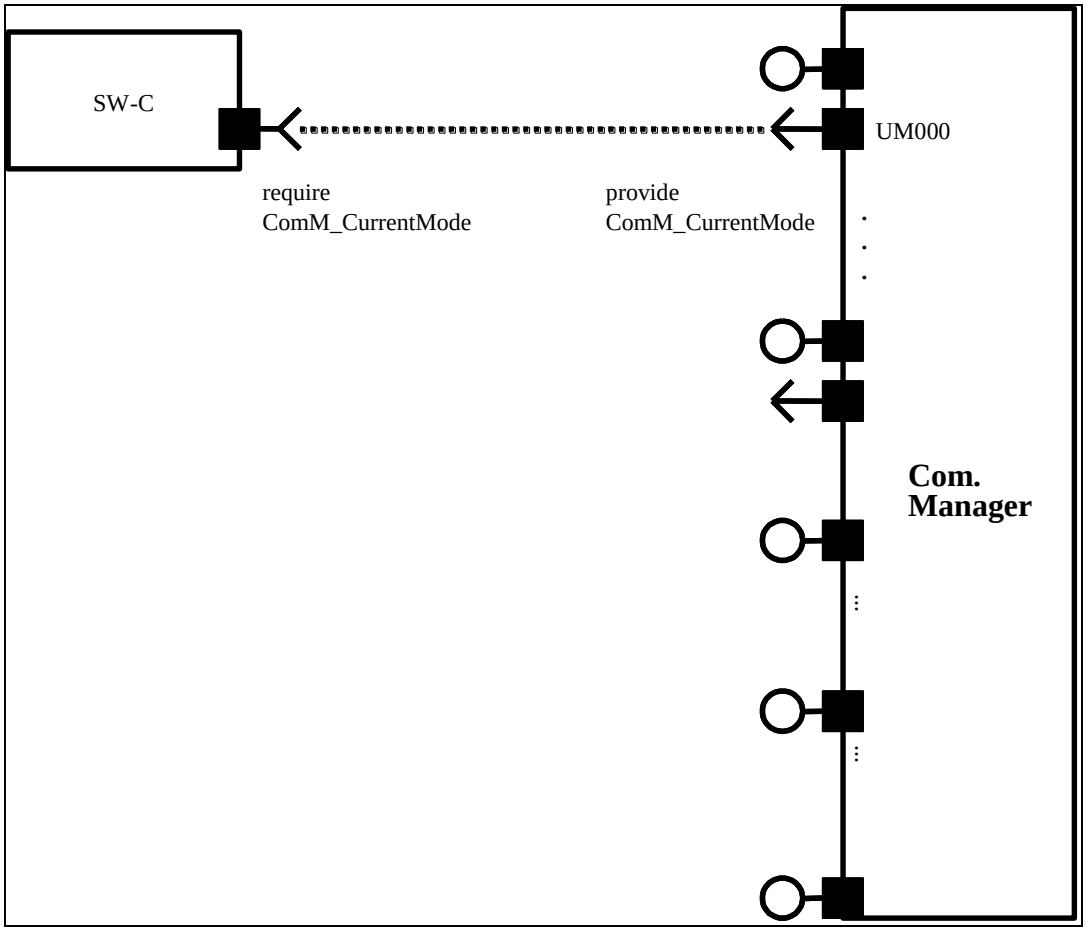

Figure 6: SW-C requests state changes to the Communication Manager Module

7.10.2.3 SW-Cs explicitly wants to take influence on its communication state

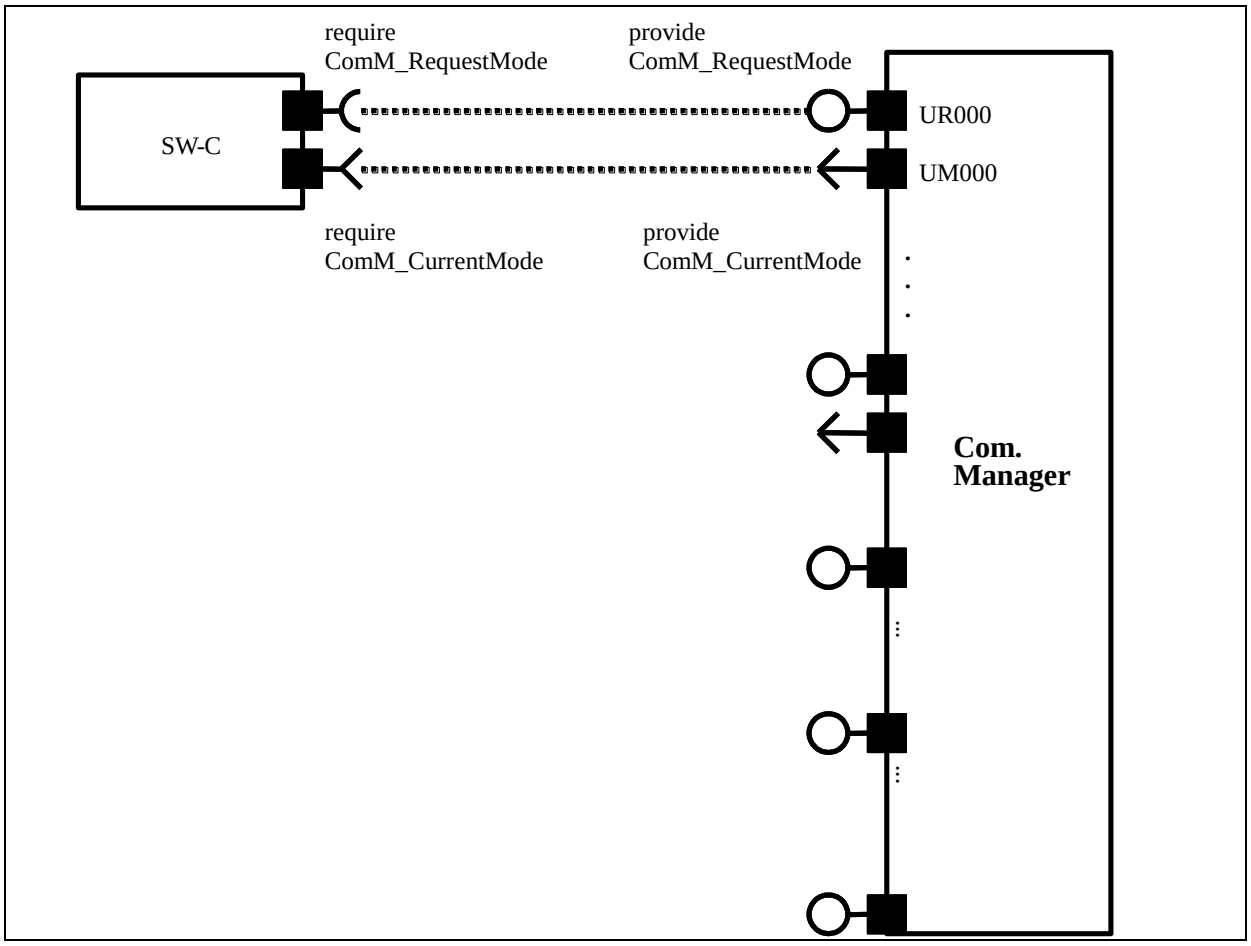

Figure 7: SW-C requires state changes within the Communication Manager Module and reads out current communication state

In this use case, the SW-C wants to explicitly take influence on the communication-state of the physical channels it needs. The SW-C indicates this by a specific port. Through this port, the SW-C can then request the Communication Manager Module mode “No Communication” or “Full Communication”. The Communication Manager Module will use these calls to request the corresponding communication mode from the corresponding Bus State Manager module.

[SWS_ComM_00848] ⌈The Communication Manager Module shall provide an AUTOSAR port to allow the request of an communication mode by calling ‘`ComM_RequestComMode`’ (see SWS_ComM_00110).⌋()

For a SW-C using the “direct API” of the RTE, the SW-C could for example do the following:

```
MySW-C_Runnable_Init(self)
{
     // SW-C wants to send and receive data
     e = Rte_Call_comRequest_RequestComMode(COMM_FULL_COMMUNICATION);
     if (e == RTE_E_OK)
     {
```

```
          // successfully requested the Com Manager Module to move to
          // full communication mode
     }
     else
     {
          // an error occurred when
          // interacting with the Com Manager module
          if (e == E_MODE_LIMITATION)
          {
               // a current ComMMode limitation forbids going into
               // that mode;
               // let's ask what the maximal allowed ComMMode is
               Rte_Call_comRequest_GetMaxComMode(&max);
               if (max==COMM_NO_COMMUNICATION)
               {
                    …
               };
          }
          else
          {
               // a more serious error occurred …
          };
     };
     …
};

MySW-C_Runnable_Loop(self)
{
     if (status == ready_to_sleep)
     {
          //no need to send; ready for shutdown communication
          Rte_Call_comRequest_RequestComMode(COMM_NO_COMMUNICATION);
          …
     };
};
```

Comment: Note that these APIs do not require that the SW-C has knowledge of the channels that it needs.

7.10.2.4 SW-C wants to interact directly with physical channels activate ECU Mode Limitation

The SW-C shall request mode from BswM. BswM will handle the direct communication with ComM.

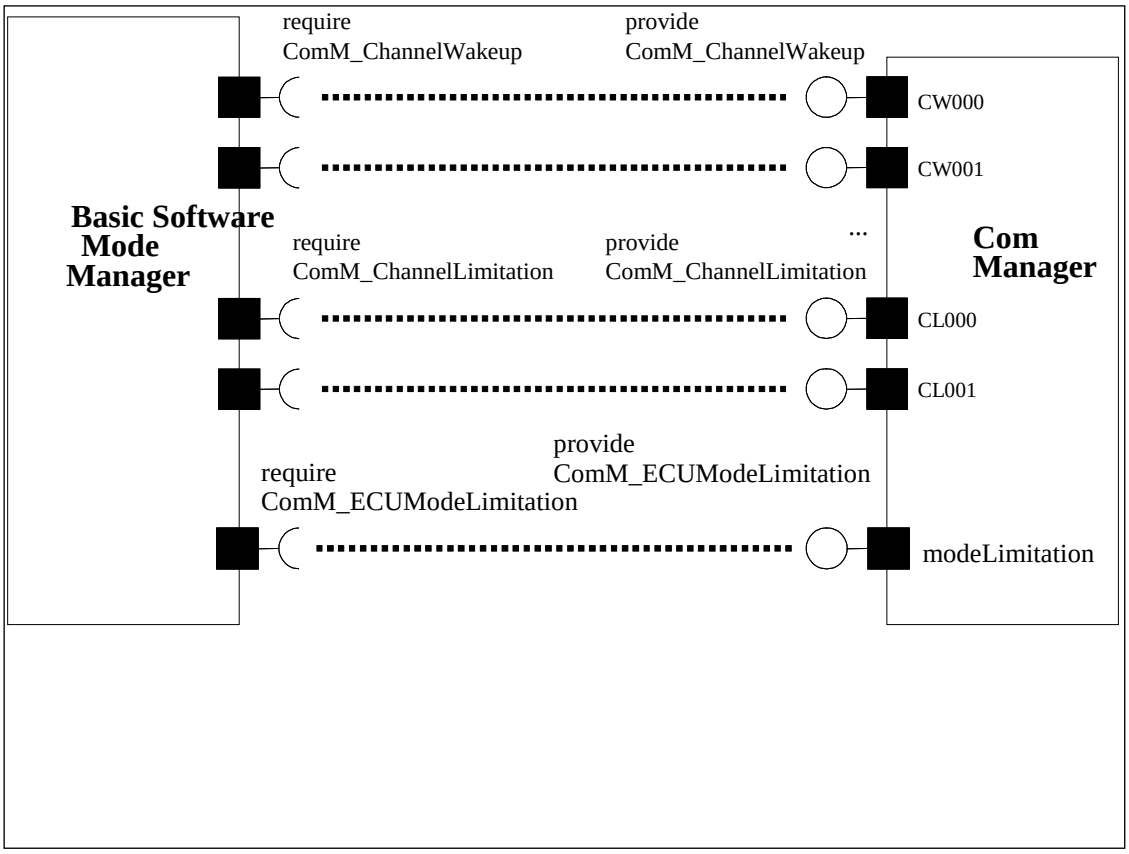

Figure 8: Interaction between BswM and the ComM module

7.10.3 Specification of Ports and Port Interfaces

This section specifies the Port Interfaces that are needed to operate the Communication Manager Module functionality over the RTE.

7.10.3.1 Types used by the interfaces

See 8.7.4 Implementation Data Types.

7.10.3.2 Ports and Port Interface for User Requests

7.10.3.2.1 General Approach

A SW-C that wants to explicitly direct the local Communication Manager Module of the ECU towards a certain state requires the client-server interface `ComM_UserRequest`. Through this interface the SW-C can set the desired state of all communication channels that are relevant for that component, to “No Communication” or “Full Communication”. In order to keep the SW-Cs code independent from the values of the handles that are used to identify the user towards the Communication Manager Module, these handles are not passed from the SW-C to the Communication Manager Module. Rather they are modeled as “port defined argument values” of the Provide Ports on the Communication Manager Module’s side. As a consequence, these handles do not show up as arguments in the operations of the client-server interface `ComM_UserRequest`. As a further consequence of this approach, the Communication Manager Module has a separate port for each user.

7.10.3.2.2 Data Types

No data types are needed for this interface.

7.10.3.2.3 Port interface ComM_UserRequest

See 8.7.2.4 ComM_UserRequest.

7.10.3.3 Ports and Port Interfaces for the current mode of the Communication Manager Module

7.10.3.3.1 General approach

[SWS_ComM_00847] ⌈The Communication Manager Module shall have an AUTOSAR port providing the ModeSwitchInterface interface 'ComM_CurrentMode'.⌋()

[SWS_ComM_00733] ⌈The Communication Manager Module shall have a separate port providing the ModeSwitchInterface interface 'ComM_CurrentMode' for each configured user, to which a SW-C is connected. ⌋()

A SW-C that wants to get informed about its current Communication Manager Module Mode requires the ModeSwitchInterface interface ComM_CurrentMode.

7.10.3.3.2 Port interface ComM_CurrentMode

See 8.7.3.1 ComM_CurrentMode.

7.10.3.4 Ports and Port Interfaces for the ComM users currently requesting FULL_COMM

7.10.3.4.1 General approach

[SWS_ComM_00734] ⌈The Communication Manager Module shall have an optional (see **ECUC_ComM_00787**) separate port providing the sender-receiver interface 'ComM_CurrentChannelRequest' for each configured ComM channel.⌋()

Rationale for SWS_ComM_00734: A SW-C that wants to get informed about, which users are currently requesting FULL_COM requires the sender-receiver interface 'ComM_CurrentChannelRequest'.

[SWS_ComM_00736] ⌈Whenever the set of ComM users currently requesting FULL_COMM for a channel changes, the Communication Manager Module shall update the data element fullComRequestors. A change shall update the data element only, when the Communication Manager Module accepts the communication request of the ComM user.⌋()

Rationale for Com736: Requests rejected because of active ModeLimitations will not lead to an update of the data element.

7.10.3.4.2 Data Types

See 8.7.4.4 ComM_UserHandleArrayType.

7.10.3.4.3 Port Interface ComM_CurrentChannelRequest

See 8.7.1.1 ComM_CurrentChannelRequest.

7.10.3.5 Ports and Port Interface for ECU Mode Limitation

7.10.3.5.1 General approach

[SWS_ComM_00740] ⌈The Communication Manager Module can be configured to have an AUTOSAR port providing the client-server interface

`ComM_ECUModeLimitation`.⌋()

A SW-C, which plays the role of a “Mode Manager”, can use this interface to change the behaviour of the entire ECU.

7.10.3.5.2 Port interface ComM_ECUModeLimitation

See 8.7.2.3 ComM_ECUModeLimitation.

7.10.3.6 Ports and Port Interface for Channel Wake up

7.10.3.6.1 General approach

[SWS_ComM_00747] ⌈The Communication Manager Module can be configured to have an AUTOSAR port providing the Client-Server Interface

`ComM_ChannelWakeup`.⌋()

A SW-C playing the role of a “Mode Manager” can use this interface to configure the Communication Manager Module to take precautions against awaking other ECU's by starting the communication. In order to keep the SW-Cs code independent from the values of the handles that are used to identify a specific handle towards the Communication Manager Module, these handles are **not** passed from the SW-C to the Communication Manager Module. Rather they are modeled as “port defined argument values” of the Provide Ports on the Communication Manager Module’s side. As a consequence, these handles do not show up as arguments in the operations of the client-server interface `ComM_ChannelWakeup`. As a further consequence of this approach, the Communication Manager Module has separate ports for each channel.

7.10.3.6.2 Port interface ComM_ChannelWakeup

See 8.7.2.2 ComM_ChannelWakeup.

7.10.3.7 Ports and Port Interface for interface Channel Limitation

7.10.3.7.1 General approach

[SWS_ComM_00752] ⌈The Communication Manager Module can be configured to have an AUTOSAR port providing the Client-Server Interface

`ComM_ChannelLimitation`.⌋()

A SW-C playing the role of a “Mode Manager” can use this interface to configure the Communication Manager Module to inhibit communication mode for a given channel. In order to keep the SW-Cs code independent from the values of the handles that are used to identify a specific handle towards the Communication Manager Module, these handles are **not** passed from the SW-C to the Communication Manager Module. Rather they are modelled as “port defined argument values” of the Provide Ports on the Communication Manager Module side. As a consequence, these handles do not show up as arguments in the operations of the client-server interface `ComM_ChannelLimitation`. As a further consequence of this approach, the Communication Manager Module has separate ports for each channel.

7.10.3.7.2 Port interface ComM_ChannelLimitation

See 8.7.2.1 ComM_ChannelLimitation.

7.10.3.8 Definition of the Service of the Communication Manager Module

This section provides guidance on the definition of the Communication Manager Module service. There are ports on both sides of the RTE. This description of the Communication Manager Module service defines the ports below the RTE. Each SW-C, which uses the Service, must contain “service ports” in its own SW-C description which will be connected to the ports of the COM Manager module, so that the RTE can be generated.

Comment: Note that these definitions can only be completed during ECU configuration (because it depends on certain configuration parameters of the Communication Manager Module, which determine the number of ports provided by the Communication Manager Module service). Also note that the implementation of an SW-C does *not* depend on these definitions.

[SWS_ComM_00744]

⌈

```
/* This is the definition of the Communication Manager Module as a service.
This is the 'outside-view' of the Communication Manager Module */
Service ComM
{
     // port present if ComMModeLimitationEnabled (see ECUC_ComM_00560)
     ProvidePort ComM_ECUModeLimitation modeLimitation;

     // port present for each channel
     // if ComMModeLimitationEnabled (see ECUC_ComM_00560);
     // there are NC channels;
     ProvidePort ComM_ChannelLimitation CL000;
     …
     ProvidePort ComM_ChannelLimitation CL<NC-1>;

     // port present for each channel
     // if COMM_WAKEUP_INHIBITION_ENABLED (see ECUC_ComM_00559)
     ProvidePort ComM_ChannelWakeup CW000;
     …
     ProvidePort ComM_ChannelWakeup CW<NC-1>;

     // For each user the Communication Manager Module provides 2 ports.
// To facilitate configuration, the index of this user shall
// correspond to the index in the array COMM_USER_LIST used for the
     // configuration of the Communication Manager Module (see ComM562).
     // The number of users must correspond to the size of this array.
     ProvidePort ComM_UserRequest UR000;   // (see 7.10.3.2.2)
     ProvidePort ComM_CurrentMode UM000;
     ProvidePort ComM_UserRequest UR001;   //(see 7.10.3.2.2)
     ProvidePort ComM_CurrentMode UM001;
     …
     ProvidePort ComM_UserRequest UR<COMM_USER_LIST.size-1>;
     ProvidePort ComM_CurrentMode UM<COMM_USER_LIST.size-1>;

     // port present for each channel if configured
     // (see ECUC_ComM_00787)
     // there are NC channels;
     ProvidePort ComM_CurrentChannelRequest CR000;
     …
     ProvidePort ComM_CurrentChannelRequest CR<NC-1>;
```

};⌋()

7.10.4 Runnables and Entry points

7.10.4.1 Internal behaviour

This is the inside description of the Communication Manager Module. This detailed description is only needed for the configuration of the local RTE.

[SWS_ComM_00745]

⌈

```
InternalBehavior of the Communication Manager Module
{
     // Runnable entities of the Communication Manager Module
     RunnableEntity LimitECUToNoComMode
          symbol "ComM_LimitECUToNoComMode" /* see SWS_ComM_00124*/
          canbeInvokedConcurrently = FALSE

     RunnableEntity ReadInhibitCounter
          symbol "ComM_ReadInhibitCounter" /* see SWS_ComM_00224 */
          canbeInvokedConcurrently = FALSE

     RunnableEntity ResetInhibitCounter
          symbol "ComM_ResetInhibitCounter" /* see SWS_ComM_00108 */
          canbeInvokedConcurrently = FALSE

     RunnableEntity SetECUGroupClassification
          symbol "ComM_SetECUGroupClassification" /* see SWS_ComM_00552 */
          canbeInvokedConcurrently = FALSE

     RunnableEntity LimitChannelToNoComMode
          symbol "ComM_LimitChannelToNoComMode" /* see SWS_ComM_00163 */
          canbeInvokedConcurrently = FALSE

     RunnableEntity GetInhibitionStatus
          symbol "ComM_GetInhibitionStatus" /*see SWS_ComM_00619 */
          canbeInvokedConcurrently = FALSE

     RunnableEntity PreventWakeup
          symbol "ComM_PreventWakeup"
          canbeInvokedConcurrently = FALSE

     RunnableEntity RequestComMode
          symbol "ComM_RequestComMode" /* see SWS_ComM_00110 */
          canbeInvokedConcurrently = TRUE

     RunnableEntity GetMaxComMode
          symbol "ComM_GetMaxComMode" /* see SWS_ComM_00085 */
          canbeInvokedConcurrently = TRUE

     RunnableEntity GetRequestedComMode
          symbol "ComM_GetRequestedComMode"
          canbeInvokedConcurrently = TRUE

     RunnableEntity GetCurrentComMode
          symbol "ComM_GetCurrentComMode" /*see SWS_ComM_00083 */
          canbeInvokedConcurrently = TRUE

     // the following applies if ComMModeLimitationEnabled
     // (see ECUC_ComM_00560)
```

```
    modeLimitation.LimitECUToNoComMode -> LimitECUToNoComMode
    modeLimitation.ReadInhibitCounter -> ReadInhibitCounter
    modeLimitation.ResetInhibitCounter -> ResetInhibitCounter
    modeLimitation.SetECUGroupClassification -> SetECUGroupClassification

    // per-channel behaviour only present
    // if ComMModeLimitationEnabled (see ECUC_ComM_00560)
    // there are NC channels
    // To facilitate configuration, the names of the channels correspond
    // to the index of the channel in the "Channel" container used to
    // configure the Communication Manager Module
    CL000.LimitChannelToNoComMode -> LimitChannelToNoComMode
    CL000.GetInhibitionStatus -> GetInhibitionStatus
    PortArgument {port=CL000,
                  value.type=NetworkHandleType,
                  value.value=Channel[0].COMM_CHANNEL_ID}
    …
CLnnn.LimitChannelToNoComMode -> LimitChannelToNoComMode
CLnnn.GetInhibitionStatus -> GetInhibitionStatus
    PortArgument {port=CLnnn,
                  value.type=NetworkHandleType,
                  value.value=Channel[nnn].COMM_CHANNEL_ID}

    // per-channel behaviour only present
    // if COMM_WAKEUP_INHIBITION_ENABLED (see ECUC_ComM_00559)
    CW000.preventWakeUp -> PreventWakeUp
    PortArgument {port=CW000,
                  value.type=NetworkHandleType,
                  value.value=Channel[0].COMM_CHANNEL_ID}
    …
    CWnnn.preventWakeUp -> PreventWakeUp
    PortArgument {port=CWnnn,
                  value.type=NetworkHandleType,
                  value.value=Channel[nnn].COMM_CHANNEL_ID}

    // per-user behaviour
    // Note that the port-argument value must be consistent with the
    // value in the configuration COMM_USER_LIST
    // Note that the exact data-type of the UserHandleType must of course
    // be defined BEFORE RTE_configuration, but does NOT affect the
    // API seen by the SW-Cs that use the service
    UR000.RequestComMode -> RequestComMode
    UR000.GetMaxComMode -> GetMaxComMode
    UR000.GetRequestedComMode -> GetRequestedComMode
    UR000.GetCurrentComMode -> GetCurrentComMode
    PortArgument {port=UR000,
                  value.type= ComM_UserhandleType,
                  value.value=COMM_USER_LIST[0]}
    …
    URnnn.RequestComMode -> RequestComMode
    URnnn.GetMaxComMode -> GetMaxComMode
    URnnn.GetRequestedComMode -> GetRequestedComMode
    URnnn.GetCurrentComMode -> GetCurrentComMode
    PortArgument {port=URnnn,
                  value.type= ComM_UserhandleType,
                  value.value=COMM_USER_LIST[n]}
};()
```

Comment: 'modeLimitation.LimitECUToNoComMode -> LimitECUToNoComMode' is supposed to define an OperationInvokedEvent that links the OperationPrototype to the runnable entity that is supposed to be executed.

7.10.4.2 Header file to be included by the Communication Manager Module

The RTE deals with the Communication Manager Module as with any normal SW-C. The RTE will be able to generate a header-file based on the internal-behaviour description of the Communication Manager Module which contains for instance a definition of the API's (like `Rte_Ports_CurrentMode_P`) which are available to the Communication Manager Module. This implies that an implementation of the Communication Manager Module must include this generated header-file.

8 API specification

8.1 Imported types

8.1.1 Standard types

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

[SWS_ComM_00820] ⌈

Header file	*Imported Type*
Std_Types.h	Std_VersionInfoType
	Std_ReturnType
ComStack_Types.h	NetworkHandleType
	PNCHandleType
NvM_Types.h	NvM_BlockIdType

⌋(SRS_BSW_00348, SRS_BSW_00357)

[SWS_ComM_00649] ⌈The Std_ReturnType shall be extended with the following #define values:

#define	Value	Description
COMM_E_MODE_LIMITATION	0x02	Function call has been successfully but mode can not be granted because of mode inhibition.

⌋(SRS_BSW_00331, SRS_BSW_00369, SRS_BSW_00377, SRS_BSW_00441)

8.2 Type definitions

[SWS_ComM_00863] ⌈The following Data Types shall be used for the functions defined in this Specification.⌋(SRS_BSW_00441)

8.2.1 ComM_InitStatusType

[SWS_ComM_00668]⌈

Name:	`ComM_InitStatusType`	
Type:	`Enumeration`	
Range:	`COMM_UNINIT`	The COM Manager is not initialized or not usable. This shall be the default value after reset. This status shall have the value 0.
	`COMM_INIT`	The COM Manager is initialized and usable.
Description:	Initialization status of ComM.	

⌋()

8.2.2 ComM_InhibitionStatusType

[SWS_ComM_00669]⌈

Name:	ComM_InhibitionStatusType		
Type:	uint8		
Range:	WakeupInhibitionActive	0x01	Bit 0 (LSB): Wake Up inhibition active
	LimitedToNoCom	0x02	Bit 1: Limit to COMM_NO_COMMUNICATION mode
Description:	Defines whether a mode inhibition is active or not. Inhibition status of ComM. e.g. status=00000011 -> Wake up inhibition and limitation to COMM_NO_COMMUNICATION mode active		

⌋()

8.2.3 ComM_UserHandleType

[SWS_ComM_00670]⌈

Name:	ComM_UserHandleType
Type:	uint8
Description:	Handle to identify a user. For each user, a unique value must be defined at system generation time. Maximum number of users is 255. Legal user IDs are in the range 0 .. 254; user ID 255 is reserved and shall have the symbolic representation COMM_NOT_USED_USER_ID.

⌋()

Comment: This handle has local scope for only one ECU.

8.2.4 ComM_ModeType

[SWS_ComM_00672]⌈

Name:	ComM_ModeType		
Type:	uint8		
Range:	COMM_NO_COMMUNICATION	0	ComM state machine is in "No Communication" mode. Configured channel shall have no transmission or reception capability.
	COMM_SILENT_COMMUNICATION	1	ComM state machine is in "Silent Communication" mode. Configured channel shall have only reception capability, no transmission capability.
	COMM_FULL_COMMUNICATION	2	ComM state machine is in "Full Communication" mode. Configured channel shall have both transmission and reception capability.
Description:	Current mode of the Communication Manager (main state of the state machine).		

⌋()

8.2.5 ComM_PncModeType

[SWS_ComM_00673]⌈

Name:	ComM_PncModeType	
Type:	Enumeration	
Range:	COMM_PNC_REQUESTED	PNC is requested by a local ComM user
	COMM_PNC_READY_SLEEP	PNC is requested by a remote ComM user
	COMM_PNC_PREPARE_SLEEP	PNC is active with no deadline monitoring
	COMM_PNC_NO_COMMUNICATION	PNC does not communicate
	COMM_PNC_FULL_COMMUNICATION	PNC is able to communicate
Description:	Current mode of a PNC	

⌋()

8.2.6 ComM_StateType

[SWS_ComM_00674]⌈

Name:	ComM_StateType		
Type:	uint8		
Range:	COMM_NO_COM_NO_PENDING_REQUEST	0	--
	COMM_NO_COM_REQUEST_PENDING	1	--
	COMM_FULL_COM_NETWORK_REQUESTED	2	--
	COMM_FULL_COM_READY_SLEEP	3	--
	COMM_SILENT_COM	4	--
Description:	State and sub-state of ComM state machine ComM states vs. Communication Modes: COMM_NO_COM* : Communication Mode='No Communication' COMM_FULL_COM*: Communication Mode='Full Communication' COMM_SILENT_COM: Communicatio Mode='Silent Communication'		

⌋()

8.2.7 ComM_ConfigType

[SWS_ComM_00162]⌈

Name:	ComM_ConfigType	
Type:	Structure	
Range:	implementation specific	The contents of the initialization data structure are implementation specific
Description:	This type contains the implementation-specific post build configuration structure.	

⌋()

8.3 Function definitions

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

8.3.1 ComM_Init

[SWS_ComM_00146]

⌈

Service name:	ComM_Init	
Syntax:	`void ComM_Init( const ComM_ConfigType* ConfigPtr )`	
Service ID[hex]:	0x01	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ConfigPtr	Pointer to post-build configuration data
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Initializes the AUTOSAR Communication Manager and restarts the internal state machines.	

⌋(SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)

[SWS_ComM_00793] ⌈Caveats of `ComM_Init()`: The NVRAM Manager module has to be initialized to have the possibility to "direct" access the ComM module's parameters.⌋()

[SWS_ComM_00864] ⌈In `ComM_Init()` ComM shall read non-volatile parameters specified in SWS_ComM_00103 from NVRAM. If no parameters are available, ComM shall use the default values in the ComM configuration.⌋()

8.3.2 ComM_DeInit

[SWS_ComM_00147]

⌈

Service name:	ComM_DeInit	
Syntax:	`void ComM_DeInit( void )`	
Service ID[hex]:	0x02	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	This API de-initializes the AUTOSAR Communication Manager.	

⌋(SRS_BSW_00336)

[SWS_ComM_00794] ⌈De-initialization in `ComM_DeInit()` shall only be performed if all channels controlled by the ComM module are in `COMM_NO_COMMUNICATION` mode.

Rationale for SWS_ComM_00794: Since the `ComM_DeInit()` API cannot return an error message, it must be assured that all channels are in `COMM_NO_COMMUNICATION` mode and `COMM_NO_COM_NO_PENDING_REQUEST` sub-state before `ComM_DeInit()` is called. E.g. the state should be checked with `ComM_GetState(Channel,…)` and `ComM_CommunicationAllowed(Channel,TRUE)` cannot be called before `ComM_DeInit()` has been called.⌋()

[SWS_ComM_00865] ⌈In `ComM_DeInit` ComM shall store non-volatile parameters specified in SWS_ComM_00103 to NVRAM.⌋()

8.3.3 ComM_GetState

[SWS_ComM_00872]

⌈

Service name:	ComM_GetState	
Syntax:	`Std_ReturnType ComM_GetState( NetworkHandleType Channel, ComM_StateType* State )`	
Service ID[hex]:	0x34	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	The Network Channel for the requested state of ComM state machine.
Parameters (inout):	None	
Parameters (out):	State	State of the ComM channel state machine: COMM_NO_COM_NO_PENDING_REQUEST COMM_NO_COM_REQUEST_PENDING COMM_FULL_COM_NETWORK_REQUESTED COMM_FULL_COM_READY_SLEEP COMM_SILENT_COM
Return value:	Std_ReturnType	E_OK: Successfully return current state of ComM state machine E_NOT_OK: Return of current state of ComM state machine

		failed
Description:	Return current state, including sub-state, of the ComM channel state machine. Usage of function only valid if EcuM/Fixed is used: To leave RUN: state/sub-state need to be COMM_NO_COM_NO_PENDING_REQUEST (No communication and no pending request to start communication) In POST RUN to return to RUN: state/sub-state need to be in COMM_NO_COM_REQUEST_PENDING (No communication, but a pending request to start communication) If EcuM/Flex and BswM is used, BswM instead use received mode indications from ComM (BswM_ComM_RequestedMode(..)).	

⌋()

8.3.4 ComM_GetStatus

[SWS_ComM_00242]

⌈

Service name:	ComM_GetStatus	

Syntax:	`Std_ReturnType ComM_GetStatus(`
	`  ComM_InitStatusType* Status`
	`)`	

Service ID[hex]:	0x03	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Status	COMM_UNINIT: The ComM is not initialized or not usable. Default value after startup or after ComM_DeInit() is called. COMM_INIT: The ComM is initialized and usable.
Return value:	Std_ReturnType	E_OK: Successfully return of initialization status E_NOT_OK: Return of initialization status failed
Description:	Returns the initialization status of the AUTOSAR Communication Manager. After a call to ComM_DeInit() ComM should have status COMM_UNINIT, and a new call to ComM_Init needed to make sure ComM restart internal state machines to defailt values.	

⌋(SRS_BSW_00406)

8.3.5 ComM_GetInhibitionStatus

[SWS_ComM_00619]

⌈

Service name:	ComM_GetInhibitionStatus

Syntax:	`Std_ReturnType ComM_GetInhibitionStatus(`
	`  NetworkHandleType Channel,`
	`  ComM_InhibitionStatusType* Status`
	`)`

Service ID[hex]:	0x04
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant

Parameters (in):	Channel	See NetworkHandleType
Parameters (inout):	None	
Parameters (out):	Status	See ComM_InhibitionStatusType
Return value:	Std_ReturnType	E_OK: Successfully returned Inhibition Status E_NOT_OK: Return of Inhibition Status failed
Description:	Returns the inhibition status of a ComM channel.	

⌋()

8.3.6 ComM_RequestComMode

[SWS_ComM_00110]

⌈

Service name:	ComM_RequestComMode	

Syntax:	`Std_ReturnType ComM_RequestComMode(`
	`  ComM_UserHandleType User,`
	`  ComM_ModeType ComMode`
	`)`	

Service ID[hex]:	0x05	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	User	Handle of the user who requests a mode
	ComMode	COMM_FULL_COMMUNICATION COMM_NO_COMMUNICATION
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed to the new mode E_NOT_OK: Changing to the new mode failed COMM_E_MODE_LIMITATION: Mode can not be granted because of mode inhibition.
Description:	Requesting of a Communication Mode by a user. Note: Internally mode COMM_SILENT_COMMUNICATION is not a valid request for a user, mode used for synchronization at shutdown. Valid modes are COMM_NO_COMMUNICATION and COMM_FULL_COMMUNICATION	

⌋(SRS_ModeMgm_09081)

[SWS_ComM_00795] ⌈Configuration of ComM_RequestComMode: Relationship between users and channels. A user is statically mapped to one or more channels.⌋()

8.3.7 ComM_GetMaxComMode

[SWS_ComM_00085]

⌈

Service name:	ComM_GetMaxComMode
Syntax:	`Std_ReturnType ComM_GetMaxComMode(` `  ComM_UserHandleType User,`

	ComM_ModeType* ComMode)	
Service ID[hex]:	0x06	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	User	Handle of the user who requests a mode
Parameters (inout):	None	
Parameters (out):	ComMode	See ComM_ModeType
Return value:	Std_ReturnType	E_OK: Successfully returned maximum allowed Communication Mode E_NOT_OK: Return of maximum allowed Communication Mode failed
Description:	Function to query the maximum allowed Communication Mode of the corresponding user.	

」()

Use Case: This function provides the possibility to request the maximum possible mode (e.g. user wants to check if it is possible to get "Full Communication" mode or if a limitation/inhibition is active). This is needed for diagnosis/debugging..

[SWS_ComM_00374] 「If more than one channel is linked to one user request and the maximum allowed modes of the channels are different, then the function ComM_GetMaxComMode shall return the lowest mode (see SWS_ComM_00867 and SWS_ComM_00868).」()

[SWS_ComM_00796] 「Configuration of `ComM_GetMaxComMode`: Relationship between users and channels. A user is statically mapped to one or more channels.」()

8.3.8 ComM_GetRequestedComMode

[SWS_ComM_00079]

「

Service name:	ComM_GetRequestedComMode	
Syntax:	Std_ReturnType ComM_GetRequestedComMode(ComM_UserHandleType User, ComM_ModeType* ComMode)	
Service ID[hex]:	0x07	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	User	Handle of the user who requests a mode
Parameters (inout):	None	
Parameters (out):	ComMode	Name of the requested mode
Return value:	Std_ReturnType	E_OK: Successfully returned requested Communication Mode E_NOT_OK: Return of requested Communication Mode failed

Description:	Function to query the currently requested Communication Mode of the corresponding user.

⌋(SRS_ModeMgm_09149)

Rationale for SWS_ComM_00079: The requested user "Communication Mode" has to be stored volatile within the Communication Manager Module itself, to prevent redundant storage of status information by the users.

Comment: If the Communication Manager Module would not have this service every user has to store the status on its own --> redundant and possibly inconsistent storage of the same data.

ComM797: Configuration of `ComM_GetRequestedComMode`: Relationship between users and channels. A user is statically mapped to one or more channels.

8.3.9 ComM_GetCurrentComMode

[SWS_ComM_00083]

⌈

Service name:	ComM_GetCurrentComMode	
Syntax:	`Std_ReturnType ComM_GetCurrentComMode( ComM_UserHandleType User, ComM_ModeType* ComMode )`	
Service ID[hex]:	0x08	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	User	Handle of the user who requests a mode
Parameters (inout):	None	
Parameters (out):	ComMode	See ComM_ModeType
Return value:	Std_ReturnType	E_OK: Successfully returned Communication Mode from Bus State Manager E_NOT_OK: Return of Communication Mode from Bus State Manager failed
Description:	Function to query the current Communication Mode. ComM shall use the corresponding interfaces of the Bus State Managers to get the current Communication Mode of the network. (Call to Bus State Manager API: XXXSM _GetCurrentComMode(...))	

⌋(SRS_ModeMgm_09084)

[SWS_ComM_00176] ⌈If more than one channel is linked to one user request and the modes of the channels are different, the function `ComM_GetCurrentComMode` shall return the lowest mode (see SWS_ComM_00867 and SWS_ComM_00868). ⌋(SRS_ModeMgm_09172)

[SWS_ComM_00798] ⌈Configuration of ComM_GetCurrentComMode: Relationship between users and channels. A user is statically mapped to one or more channels.⌋()

8.3.10 ComM_PreventWakeUp

[SWS_ComM_00156]

⌈

Service name:	ComM_PreventWakeUp	
Syntax:	`Std_ReturnType ComM_PreventWakeUp( NetworkHandleType Channel, boolean Status )`	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	See NetworkHandleType
	Status	FALSE: Wake up inhibition is switched off TRUE: Wake up inhibition is switched on
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed wake up status for the channel E_NOT_OK: Changed of wake up status for the channel failed
Description:	Changes the inhibition status COMM_NO_WAKEUP for the corresponding channel.	

⌋(SRS_ModeMgm_09157)

[SWS_ComM_00799] ⌈Configuration of ComM_PreventWakeUp: Configurable with COMM_WAKEUP_INHIBITION_ENABLED (see ECUC_ComM_00559).⌋()

8.3.11 ComM_LimitChannelToNoComMode

[SWS_ComM_00163]

⌈

Service name:	ComM_LimitChannelToNoComMode	
Syntax:	`Std_ReturnType ComM_LimitChannelToNoComMode( NetworkHandleType Channel, boolean Status )`	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	See NetworkHandleType
	Status	FALSE: Limit channel to COMM_NO_COMMUNICATION disabled TRUE: Limit channel to COMM_NO_COMMUNICATION enabled
Parameters (inout):	None	

Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed inhibition status for the channel E_NOT_OK: Changed of inhibition status for the channel failed
Description:	Changes the inhibition status for the channel for changing from COMM_NO_COMMUNICATION to a higher Communication Mode. (See also ComM_LimitECUToNoComMode, same functionality but for all channels)	

⌋(SRS_ModeMgm_09157)

[SWS_ComM_00800] ⌈Configuration of ComM_LimitChannelToNoComMode: Configurable with ComMModeLimitationEnabled (see ECUC_ComM_00560) and COMM_RESET_AFTER_FORCING_NO_COMM (see ComM558_Conf).⌋()

8.3.12 ComM_LimitECUToNoComMode

[SWS_ComM_00124]

⌈

Service name:	ComM_LimitECUToNoComMode	

Syntax:	`Std_ReturnType ComM_LimitECUToNoComMode(`
	`  boolean Status`
	`)`	

Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Status	FALSE: Limit ECU to COMM_NO_COMMUNICATION disabled TRUE: Limit ECU to COMM_NO_COMMUNICATION enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed inhibition status for the ECU E_NOT_OK: Changed of inhibition status for the ECU failed
Description:	Changes the inhibition status for the ECU (=all channels) for changing from COMM_NO_COMMUNICATION to a higher Communication Mode. (See also ComM_LimitChannelToNoComMode, same functionality but for a specific channels)	

⌋(SRS_ModeMgm_09157)

[SWS_ComM_00801] ⌈Configuration of ComM_LimitECUToNoComMode: Configurable with ComMModeLimitationEnabled (see ECUC_ComM_00560) and COMM_RESET_AFTER_FORCING_NO_COMM (see ComM558_Conf).⌋()

8.3.13 ComM_ReadInhibitCounter

[SWS_ComM_00224]

⌈

Service name:	ComM_ReadInhibitCounter	
Syntax:	`Std_ReturnType ComM_ReadInhibitCounter( uint16* CounterValue )`	
Service ID[hex]:	0x0d	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	CounterValue	Amount of rejected COMM_FULL_COMMUNICATION user requests.
Return value:	Std_ReturnType	E_OK: Successfully returned Inhibition Counter E_NOT_OK: Return of Inhibition Counter failed
Description:	This function returns the amount of rejected COMM_FULL_COMMUNICATION user requests.	

」(SRS_ModeMgm_09156)

[SWS_ComM_00802] 「Configuration of ComM_ReadInhibitCounter: Configurable with ComMModeLimitationEnabled (see ECUC_ComM_00560). Function will only be available if ComMModeLimitationEnabled (see ECUC_ComM_00560) is enabled.」()

8.3.14 ComM_ResetInhibitCounter

[SWS_ComM_00108]

「

Service name:	ComM_ResetInhibitCounter	
Syntax:	`Std_ReturnType ComM_ResetInhibitCounter( void )`	
Service ID[hex]:	0x0e	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully reset of Inhibit COMM_FULL_COMMUNICATION Counter E_NOT_OK: Reset of Inhibit COMM_FULL_COMMUNICATION Counter failed
Description:	This function resets the Inhibited COMM_FULL_COMMUNICATION request Counter.	

」(SRS_ModeMgm_09156)

[SWS_ComM_00803] 「Configuration of ComM_ResetInhibitCounter: Configurable with ComMModeLimitationEnabled (see ECUC_ComM_00560). Function will only be available if ComMModeLimitationEnabled (see ECUC_ComM_00560) is enabled.」()

8.3.15 ComM_SetECUGroupClassification

[SWS_ComM_00552]

⌈

Service name:	ComM_SetECUGroupClassification	
Syntax:	`Std_ReturnType ComM_SetECUGroupClassification( ComM_InhibitionStatusType Status )`	
Service ID[hex]:	0x0f	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Status	See ComM_InhibitionStatusType
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully change the ECU Group Classification Status E_NOT_OK: Change of the ECU Group Classification Status failed
Description:	Changes the ECU Group Classification status (see chapter 10.2.2)	

⌋()

8.3.16 ComM_GetVersionInfo

[SWS_ComM_00370]

⌈

Service name:	ComM_GetVersionInfo	
Syntax:	`void ComM_GetVersionInfo( Std_VersionInfoType* Versioninfo )`	
Service ID[hex]:	0x10	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	None	
Parameters (inout):	None	
Parameters (out):	Versioninfo	See Std_VersionInfoType
Return value:	None	
Description:	This function returns the published information (for details refer to table 10.3)	

⌋(SRS_BSW_00407)

8.4 Callback notifications

This is a list of functions provided for other modules. The function prototypes of the callback functions shall be provided in the header files `ComM_Nm.h`, `ComM_EcuMBswM.h`, `ComM_Dcm.h`, `ComM_BusSM.h` and `ComM_Com.h`.

[SWS_ComM_00620] ⌈All the provided indication functions shall be implemented pre-compile time.⌋(SRS_BSW_00387)

8.4.1 AUTOSAR Network Management Interface

8.4.1.1 ComM_Nm_NetworkStartIndication
[SWS_ComM_00383]

⌈

Service name:	ComM_Nm_NetworkStartIndication	
Syntax:	`void ComM_Nm_NetworkStartIndication( NetworkHandleType Channel )`	
Service ID[hex]:	0x15	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	See NetworkHandleType
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication that a NM-message has been received in the Bus Sleep Mode, what indicates that some nodes in the network have already entered the Network Mode.	

⌋()

[SWS_ComM_00805] ⌈Caveats of `ComM_Nm_NetworkStartIndication`: The ComM module is initialized correctly.⌋()

8.4.1.2 ComM_Nm_NetworkMode
[SWS_ComM_00390]

⌈

Service name:	ComM_Nm_NetworkMode	

Syntax:	`void ComM_Nm_NetworkMode(`
	`  NetworkHandleType Channel`
	`)`	

Service ID[hex]:	0x18	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has entered Network Mode.	

⌋()

[SWS_ComM_00806] ⌈Caveats of `ComM_Nm_NetworkMode`: The Communication Manager Module is initialized correctly.⌋()

8.4.1.3 ComM_Nm_PrepareBusSleepMode
[SWS_ComM_00391]

⌈

Service name:	ComM_Nm_PrepareBusSleepMode	

Syntax:	`void ComM_Nm_PrepareBusSleepMode(`
	`  NetworkHandleType Channel`
	`)`	

Service ID[hex]:	0x19	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has entered Prepare Bus-Sleep Mode. Reentrancy: Reentrant (but not for the same NM-Channel)	

⌋()

[SWS_ComM_00808] ⌈Caveats of `ComM_Nm_PrepareBusSleepMode`: The Communication Manager Module is initialized correctly.⌋()

8.4.1.4 ComM_Nm_BusSleepMode
[SWS_ComM_00392]

⌜

Service name:	ComM_Nm_BusSleepMode	

Syntax:	`void ComM_Nm_BusSleepMode(`
	`  NetworkHandleType Channel`
	`)`	

Service ID[hex]:	0x1a	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Notification that the network management has entered Bus-Sleep Mode. This callback function should perform a transition of the hardware and transceiver to bus-sleep mode.	

⌟()

[SWS_ComM_00810] ⌜Caveats of `ComM_Nm_BusSleepMode`: The Communication Manager Module is initialized correctly.⌟()

8.4.1.5 ComM_Nm_RestartIndication
[SWS_ComM_00792]

⌜

Service name:	ComM_Nm_RestartIndication	

Syntax:	`void ComM_Nm_RestartIndication(`
	`  NetworkHandleType Channel`
	`)`	

Service ID[hex]:	0x1b	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	If NmIf has started to shut down the coordinated busses, AND not all coordinated busses have indicated bus sleep state, AND on at least on one of the coordinated busses NM is restarted, THEN the NM Interface shall call the callback function ComM_Nm_RestartIndication with the nmNetworkHandle of the channels which have already indicated bus sleep state.	

⌟()

[SWS_ComM_00812] ⌜Caveats of `ComM_Nm_RestartIndication`: The ComM module is initialized correctly.⌟()

8.4.2 AUTOSAR Diagnostic Communication Manager Interface

8.4.2.1 ComM_DCM_ActiveDiagnostic [SWS_ComM_00873]

⌈

Service name:	ComM_DCM_ActiveDiagnostic	
Syntax:	`void ComM_DCM_ActiveDiagnostic( NetworkHandleType Channel )`	
Service ID[hex]:	0x1f	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel needed for Diagnostic communication
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of active diagnostic by the DCM.	

⌋()

8.4.2.2 ComM_DCM_InactiveDiagnostic [SWS_ComM_00874]

⌈

Service name:	ComM_DCM_InactiveDiagnostic	
Syntax:	`void ComM_DCM_InactiveDiagnostic( NetworkHandleType Channel )`	
Service ID[hex]:	0x20	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel no longer needed for Diagnostic communication
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	Indication of inactive diagnostic by the DCM.	

⌋()

8.4.3 AUTOSAR ECU State Manager Interface

8.4.3.1 ComM_EcuM_WakeUpIndication [SWS_ComM_00275]

⌈

Service name:	ComM_EcuM_WakeUpIndication	
Syntax:	`void ComM_EcuM_WakeUpIndication( NetworkHandleType Channel )`	
Service ID[hex]:	0x2a	
Sync/Async:	Synchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	Channel

Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	Notification of a wake up on the corresponding channel.

⌋()

[SWS_ComM_00814] ⌈Caveats of `ComM_EcuM_WakeUpIndication`: The Communication Manager Module is initialized correctly.⌋()

8.4.4 AUTOSAR ECU State Manager and Basic Software Mode Manager Interface

8.4.4.1 ComM_CommunicationAllowed

[SWS_ComM_00871]

⌈

Service name:	ComM_CommunicationAllowed	
Syntax:	`void ComM_CommunicationAllowed( NetworkHandleType Channel, boolean Allowed )`	
Service ID[hex]:	0x35	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	Channel	Channel
	Allowed	TRUE: Communication is allowed FALSE: Communication is not allowed
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	EcuM or BswM shall indicate to ComM when communication is allowed. If EcuM/Fixed is used: EcuM/Fixed. If EcuM/Flex is used: BswM	

⌋()

8.4.5 Bus State Manager Interface

8.4.5.1 ComM_BusSM_ModeIndication

[SWS_ComM_00675]

⌈

Service name:	ComM_BusSM_ModeIndication
Syntax:	`void ComM_BusSM_ModeIndication( NetworkHandleType Channel, ComM_ModeType* ComMode )`

Service ID[hex]:	0x33	
Sync/Async:	Asynchronous	
Reentrancy:	Reentrant	
Parameters (in):	Channel	See NetworkHandleType
	ComMode	See ComM_ModeType
Parameters (inout):	None	
Parameters (out):	None	
Return value:	None	
Description:	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.	

」()

[SWS_ComM_00816] 「Caveats of `ComM_BusSM_ModeIndication(…)`: The Communication Manager Module is initialized correctly.」()

8.4.6 COM Interface

[SWS_ComM_00819]

「

Service name:	ComM_COMCbk_<sn>
Syntax:	`void ComM_COMCbk_<sn>( void )`
Service ID[hex]:	0x36
Sync/Async:	Synchronous
Reentrancy:	Non Reentrant
Parameters (in):	None
Parameters (inout):	None
Parameters (out):	None
Return value:	None
Description:	This callback is called when the EIRA or ERA was updated in COM. The call only informs the ComM about ERA and EIRA changes. The actual handling is done in the next call to ComM_MainFunction_<Channel_Id> with changing the corresponding PN State machine.

」()

8.5 Scheduled functions

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

8.5.1 ComM_MainFunction

[SWS_ComM_00429]

⌈

Service name:	ComM_MainFunction_<Channel_Id>

Syntax:	`void ComM_MainFunction_<Channel_Id>(`
	`  void`
	`)`

Service ID[hex]:	0x60
Description:	This function shall perform the processing of the AUTOSAR ComM activities that are not directly initiated by the calls e.g. from the RTE. There shall be one dedicated Main Function for each channel of ComM. Precondition: ComM shall be initialized

⌋(SRS_BSW_00373, SRS_BSW_00376)

[SWS_ComM_00818] ⌈Configuration of `ComM_MainFunction_<Channel_Id>`: See section 10.2.2.⌋()

8.6 Expected interfaces

In this chapter all interfaces required from other modules are shown. An overview of the required interfaces is shown in Figure 1.

8.6.1 Mandatory Interfaces

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

[SWS_ComM_00828] ⌈

API function	***Module***	***Description***
Nm_PassiveStartUp	Nm	This function calls the <BusNm>_PassiveStartUp function (e.g. CanNm_PassiveStartUp function is called if channel is configured as CAN).
Nm_NetworkRequest	Nm	This function calls the <BusNm>_NetworkRequest (e.g. CanNm_NetworkRequest function is called if channel is configured as CAN).
Nm_NetworkRelease	Nm	This function calls the <BusNm>_NetworkRelease bus specific function (e.g. CanNm_NetworkRelease function is called if channel is configured as CAN).
Dcm_ComM_NoComModeEntered	Dcm	This call informs the Dcm module about a ComM mode change to `COMM_NO_COMMUNICATION`.
Dcm_ComM_SilentComModeEntered	Dcm	This call informs the Dcm module about a ComM mode change to

		COMM_SILENT_COMMUNICATION.
Dcm_ComM_FullComModeEntered	Dcm	This call informs the Dcm module about a ComM mode change to COMM_FULL_COMMUNICATION.
Rte_Ports_UserMode_P()[n].Switch_currentMode(RTE_MODE_ComMMode_COMM_NO_COMMUNICATION)	Rte	Indicate COMM_NO_COMMUNICATION mode to RTE
Rte_Ports_UserMode_P()[n].Switch_currentMode(RTE_MODE_ComMMode_COMM_SILENT_COMMUNICATION)	Rte	Indicate COMM_SILENT_COMMUNICATION mode to RTE
Rte_Ports_UserMode_P()[n].Switch_currentMode(RTE_MODE_ComMMode_COMM_FULL_COMMUNICATION)	Rte	Indicate COMM_FULL_COMMUNICATION mode to RTE
BswM_ComM_CurrentMode	BswM	Indicate Communication Mode to BswM
NvM_ReadBlock	NvM	NVRAM manager API for Read block
NvM_WriteBlock	NvM	NVRAM manager API for Write block
NvM_GetErrorStatus	NvM	NVRAM manager API for Get status
<BusSM>_GetCurrentComMode	<BusSM>	Function to query the actual communication mode from the <Bus> State Manager.
<BusSM>_RequestComMode	<BusSM>	Function to request a communication mode from the <Bus> State Manager.

」()

8.6.1.1 AUTOSAR NVRAM Manager module

[SWS_ComM_00103] 「The ComM module shall use the corresponding standardized services of the NVRAM Manager module (see SWS_ComM_00828) for storing and reading non-volatile configuration data ComMNoWakeup (see ECUC_ComM_00569), ComMEcuGroupClassification(see ECUC_ComM_00563), inhibition status (see SWS_ComM_00157) and the Inhibit counter (see SWS_ComM_00140).」()

Comment: See SWS_ComM_00864 and SWS_ComM_00865 when configuration data shall be read and stored

For details refer to the AUTOSAR NVRAM Manager module Specification [7].

8.6.1.2 AUTOSAR Bus State Manager

[SWS_ComM_00962]「 The prefix for the StateManager APIs ("<BusSm>") shall be CanSM, LinSM, FrSM, EthSM if the Parameter ComMBusType is COMM_BUS_TYPE_CAN, COMM_BUS_TYPE_LIN, COMM_BUS_TYPE_FR or COMM_BUS_TYPE_ETH accordingly.」()

[SWS_ComM_00957]「 If ComMBusType = "COMM_BUS_TYPE_CDD" the API prefix ("<BusSm>") shall be configured in the Parameter "ComMCDDBusPrefix".」()

[SWS_ComM_00963]「 The Communication Manager module shall use <BusSm>_GetCurrentComMode() from the State Manager to query the current communication mode if necessary.」()

[SWS_ComM_00958]⌈ The Communication Manager module shall use <BusSm>_RequestComMode() from the State Manager to request a dedicated communication mode.⌋()

When it is necessary to request a dedicated communication mode depends on the current status of each instance of the channel state machine (see above).

For details of the functionality of the Bus State Manager modules refer to their Specification [23], [24], [25], [28].

Comment: Those APIs can be called re-entrant, as long as different channel & controller numbers are used.

8.6.1.3 AUTOSAR Network Management Interface

[SWS_ComM_00261] ⌈The ComM module shall use the corresponding functions to synchronize the bus start-up and shutdown of the Network Management (see SWS_ComM_00828).

For details refer to the AUTOSAR NM Interface Specification [9].⌋()

8.6.1.4 AUTOSAR Diagnostic Communication Manager Module

[SWS_ComM_00266] ⌈The ComM module shall use the corresponding functions provided by DCM (see SWS_ComM_00828) to control the communication capabilities of the DCM module.⌋()

Comment: DCM provides no functions to start/stop transmission and reception. DCM ensures to control communication according the indicated Communication Manager Module states.

For details refer to the AUTOSAR DCM Specification [11].

8.6.1.5 AUTOSAR RTE interface provided by RTE to ComM for the SW-C

[SWS_ComM_00091] ⌈The ComM module shall use the corresponding function provided by RTE to indicate modes to the users. There shall be one indication per user. Fan-out in case of a mode indication related to more than one user shall be done by the Communication Manager Module.⌋(SRS_ModeMgm_09085)

[SWS_ComM_00663] ⌈If more than one channel is linked to one user request and the modes of the channels are different, the ComM module shall indicate the lowest mode to the user.⌋()

[SWS_ComM_00662] ⌈The sequence of users shall start with user 0 up to user N and the name of the mode ports shall be UM000, UM001, ... UM<N>.⌋()

Rationale for SWS_ComM_00662: It shall be possible to use the port based API also to address specific users directly.

Comment: Within the array of ports, the ports are named alphabetically.

[SWS_ComM_00778] ⌈The ComM module shall explicitly indicate changes in modes to each individual user, to which a SW-C is connected. The ComM module shall do this by calling the right API on the RTE through the ports "UMnnn".⌋()

Comment: There is one such port per configured user to which a SW-C is connected. For users not used by SW-Cs (e.g. the users created due to **ECUC_ComM_00840 :**) no mode port will be created.

Implementation Hint: An implementation of the ComM module could use any of the normal RTE-mechanisms to signal changes in the mode to the users. Given the specific configurability of the Communication Manager Module, using the RTE "Indirect API" seems most appropriate. This works as follows (consult the RTE specification for details).

An implementation of the Communication Manager Module can use the "Rte_Ports" API to obtain an array of the "UMnnn" ports at run-time.

```
/* Return an array of all ports that provide the interface ComM_CurrentMode.
   Because of the specific naming conventions chosen, the element n in this
   array of ports will reference to the port UM<nnn>. For example
   userModePorts[1] will be a handle on port UM001 */
   userModePorts = Rte_Ports_ComM_CurrentMode_P();
```

The number of such userModePorts can be obtained through the call Rte_NPorts_ComM_CurrentMode_P. This value corresponds to the size of the COMM_USER_LIST array.

To signal that a user n is in a new mode, the Communication Manager Module should: userModePorts[n].Switch_currentMode(newMode)

For details refer to the AUTOSAR RTE specification [8] and AUTOSAR Services Mode Management specification [21].

8.6.1.6 Basic Software Mode Manager (BswM)

[SWS_ComM_00861] ⌈The ComM module shall use the corresponding function provided by BswM to report the states of Communication Manager Module channels (see SWS_ComM_00828).⌋()

For details refer to AUTOSAR Basic Software Mode Manager module [29] .

8.6.2 Optional Interfaces

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

[SWS_ComM_00829] ⌈

API function	*Module*	*Description*
Det_ReportError	Det	Service to report development errors
BswM_ComM_CurrentPNCMode	BswM	Function called by ComM to indicate the current mode of the PNC

⌋()

8.6.2.1 AUTOSAR DET

The Communication Manager module shall use Det_ReportError from the Development Error Tracer Module to report development errors.

8.6.3 Configurable Interfaces

None.

8.6.4 AUTOSAR COM

Service name:	Com_SendSignal

Syntax:	`uint8 Com_SendSignal(`
	`  Com_SignalIdType SignalId,`
	`  const void* SignalDataPtr`

	)	
Service ID[hex]:	0x0a	
Sync/Async:	Asynchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	
Parameters (in):	SignalId	Id of signal to be sent.
	SignalDataPtr	Reference to the signal data to be transmitted.
Parameters (inout):	None	
Parameters (out):	None	
Return value:	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description:	The service Com_SendSignal updates the signal object identified by SignalId with the signal referenced by the SignalDataPtr parameter.	

Service name:	Com_ReceiveSignal	

Syntax:	`uint8 Com_ReceiveSignal(`
	`  Com_SignalIdType SignalId,`
	`  void* SignalDataPtr`
	`)`	

Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant for the same signal. Reentrant for different signals.	
Parameters (in):	SignalId	Id of signal to be received.
Parameters (inout):	None	
Parameters (out):	SignalDataPtr	Reference to the location where the received signal data shall be stored
Return value:	uint8	E_OK: service has been accepted COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped (or service failed due to development error) COM_BUSY: in case the TP-Buffer is locked for large data types handling
Description:	Com_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.	

8.7 Service Interfaces

8.7.1 Sender-Receiver-interfaces

8.7.1.1 ComM_CurrentChannelRequest

[SWS_ComM_00904]⌈

Name	ComM_CurrentChannelRequest_{channel_name}
Comment	Array of ComMUserIdentifier, that currently hold FULL_COM requests for this channel. The size of the attribute fullComRequestors.handleArray is NUM_COMM_USER_PER_CHANNEL

IsService	true	
Variation	{ecuc(ComM/ComMConfigSet/ComMChannel/ ComMFullCommRequestNotificationEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}	
Data Elements	fullComRequestors	
	Type	ComM_UserHandleArrayType_{channel_name}
	Variation	channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}

」()

8.7.2 Client-Server-interfaces

8.7.2.1 ComM_ChannelLimitation

[SWS_ComM_00743]「

Name	ComM_ChannelLimitation	
Comment	A SW-C playing the role of a "Mode Manager" can use this interface to configure the Communication Manager Module to inhibit communication mode for a given channel.	
IsService	true	
Variation	{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

GetInhibitionStatus		
Comments	returns the inhibition status of a channel	
Variation	--	
Parameters	Status	
	Comment	--
	Type	ComM_InhibitionStatusType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

LimitChannelToNoComMode		
Comments	Changes the inhibition status for the channel for changing from COMM_NO_COMMUNICATION to a higher Communication Mode. (See also ComM_LimitECUToNoComMode, same functionality but for all channels)	
Variation	--	
Parameters	Status	
	Comment	FALSE: Limit channel to COMM_NO_COMMUNICATION disabled TRUE: Limit channel to COMM_NO_COMMUNICATION enabled
	Type	boolean
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」()

8.7.2.2 ComM_ChannelWakeup

[SWS_ComM_00742]「

Name	ComM_ChannelWakeup	
Comment	A SW-C playing the role of a "Mode Manager" can use this interface to configure the Communication Manager Module to take precautions against awakening other ECU's by starting the communication.	
IsService	true	
Variation	{ecuc(ComM/ComMGeneral.ComMWakeupInhibitionEnabled)} == true	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

GetInhibitionStatus		
Comments	returns the inhibition status of a channel	
Variation	--	
Parameters	Status	
	Comment	--
	Type	ComM_InhibitionStatusType

	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

PreventWakeUp		
Comments	Changes the inhibition status COMM_NO_WAKEUP for the corresponding channel.	
Variation	--	
Parameters	Status	
	Comment	--
	Type	boolean
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」()

8.7.2.3 ComM_ECUModeLimitation

[SWS_ComM_00741]「

Name	ComM_ECUModeLimitation	
Comment	A SW-C which plays the role of a "Mode Manager" can use this interface to change the behavior of the entire ECU.	
IsService	true	
Variation	{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true	
Possible Errors	0	E_OK
	1	E_NOT_OK

Operations

LimitECUToNoComMode	
Comments	Changes the inhibition status for the ECU (=all channels) for changing from COMM_NO_COMMUNICATION to a higher Communication Mode. (See also ComM_LimitChannelToNoComMode, same functionality but for a specific channels)
Variation	--

Parameters	Status	
	Comment	FALSE: Limit ECU to COMM_NO_COMMUNICATION disabled TRUE: Limit ECU to COMM_NO_COMMUNICATION enabled
	Type	boolean
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

ReadInhibitCounter		
Comments	returns the value of the 'inhibited full communication request counter'	
Variation	--	
Parameters	CounterValue	
	Comment	--
	Type	uint16
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

ResetInhibitCounter		
Comments	reset the "inhibited full communication request counter"	
Variation	--	
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

SetECUGroupClassification	
Comments	changes the ECU group classification status
Variation	--
Parameters	Status

	Comment	--
	Type	ComM_InhibitionStatusType
	Variation	--
	Direction	IN
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

」()

8.7.2.4 ComM_UserRequest

[SWS_ComM_01000]「

Name	ComM_UserRequest	
Comment	A SW-C that wants to explicitly direct the local Communication Manager Module of the ECU towards a certain state requires the client-server interface ComM_UserRequest. Through this interface the SW-C can set the desired state of all communication channels that are relevant for that component to "No Communication" or "Full Communication".	
IsService	true	
Variation	--	
Possible Errors	0	E_OK
	1	E_NOT_OK
	2	E_MODE_LIMITATION

Operations

GetCurrentComMode		
Comments	Returns the current Communication Manager Module mode for the SW-C	
Variation	--	
Parameters	ComMode	
	Comment	--
	Type	ComM_ModeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

GetMaxComMode		
Comments	Returns the current Communication Manager Module mode for the SW-C	
Variation	--	
Parameters	ComMode	
	Comment	--
	Type	ComM_ModeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

GetRequestedComMode		
Comments	Returns that last Communication Manager Module Mode requested by the SW-C	
Variation	--	
Parameters	ComMode	
	Comment	--
	Type	ComM_ModeType
	Variation	--
	Direction	OUT
Possible Errors	E_OK	Operation successful
	E_NOT_OK	--

RequestComMode		
Comments	The SW-C requests that all communication channels it needs are in the provided Communication Manager Module mode	
Variation	--	
Parameters	ComMode	
	Comment	--
	Type	ComM_ModeType
	Variation	--
	Direction	IN

Possible Errors	E_OK	Operation successful
	E_NOT_OK	--
	E_MODE_LIMITATION	ComMMode cannot be granted because of ComMMode inhibition

⌋()

8.7.3 Mode-Switch-Interfaces

8.7.3.1 ComM_CurrentMode

[SWS_ComM_01001]⌈

Name	ComM_CurrentMode	
Comment	A SW-C that wants to get informed about its current Communication Manager Module Mode requires the ModeSwitchInterface ComM_CurrentMode.	
IsService	true	
Variation	--	
ModeGroup	currentMode	ComMMode

⌋()

8.7.4 Implementation Data Types

8.7.4.1 ComM_InhibitionStatusType

[SWS_ComM_01002]⌈

Name	ComM_InhibitionStatusType			
Kind	Structure (Bitfield)			
Derived from	uint8			
Elements	Kind	Name	Mask	Description
	bit	WakeupInhibitionActive	0x01	Bit 0 (LSB): Wake Up inhibition active
	bit	LimitedToNoCom	0x02	Bit 1: Limit to COMM_NO_COMMUNICATION mode
Description	Defines whether a mode inhibition is active or not. Inhibition status of ComM. e.g. status=00000011 -> Wake up inhibition and limitation to COMM_NO_COMMUNICATION mode active			

⌋()

8.7.4.2 ComM_ModeType

[SWS_ComM_01003]⌈

Name	ComM_ModeType		
Kind	Type		
Derived from	uint8		
Description	Current mode of the Communication Manager (main state of the state machine).		
Range	COMM_NO_COMMUNICATION	0	ComM state machine is in "No Communication" mode. Configured channel shall have no transmission or reception capability.
	COMM_SILENT_COMMUNICATION	1	ComM state machine is in "Silent Communication" mode. Configured channel shall have only reception capability, no transmission capability.
	COMM_FULL_COMMUNICATION	2	ComM state machine is in "Full Communication" mode. Configured channel shall have both transmission and reception capability.
Variation	--		

⌋()

8.7.4.3 ComM_UserHandleType

[SWS_ComM_01004]⌈

Name	ComM_UserHandleType
Kind	Type
Derived from	uint8
Description	Handle to identify a user. For each user, a unique value must be defined at system generation time. Maximum number of users is 255. Legal user IDs are in the range 0 .. 254; user ID 255 is reserved and shall have the symbolic representation COMM_NOT_USED_USER_ID.
Variation	--

⌋()

8.7.4.4 ComM_UserHandleArrayType

[SWS_ComM_00906]⌈

Name	ComM_UserHandleArrayType_{channel_name}		
Kind	Structure		
Elements	numberOfRequesters	uint8	

	handleArray	ComM_UserHandleSubArrayType_{channel_name}	
Description	numberOfRequesters contains the number of valid user handle entries in the "handleArray" member. If no user keeps the channel requested, this is zero {LOWER-LIMIT=0, UPPER-LIMIT= MAX_CHANNEL_REQUESTER }		
Variation	channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}		

⌋()

8.7.4.5 ComM_UserHandleSubArrayType

[SWS_ComM_01005]⌈

Name	ComM_UserHandleSubArrayType_{channel_name}		
Kind	Array	Element type	ComM_UserHandleType
Size	COUNT{ecuc(ComM/ComMConfigSet/ComMChannel/ComMUserPerChannel)} Elements		
Description	This element contains the user handles of the users which keep the channel requested (if any), starting in its first entries. The size of the array MAX_CHANNEL_REQUESTERS is the maximum of the number of users requesting a channel.		
Variation	channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}		

⌋()

8.7.5 Ports

8.7.5.1 ComM_CL

[SWS_ComM_01006]⌈

Name	CL_{channel_name}		
Kind	ProvidedPort	Interface	ComM_ChannelLimitation
Description	The numeric value for the port defined argument for ComMChannelId shall be derived from Configuration.		
Variation	{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel)}		

⌋()

8.7.5.2 ComM_CR

[SWS_ComM_01007]⌈

Name	CR_{channel_name}		
Kind	ProvidedPort	Interface	ComM_CurrentChannelRequest_{channel_name}
Description	The numeric value for the port defined argument for ComMChannelId shall be derived from Configuration.		

Variation	{ecuc(ComM/ComMConfigSet/ComMChannel/ ComMFullCommRequestNotificationEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}

」()

8.7.5.3 ComM_CW

[SWS_ComM_01008]「

Name	CW_{channel_name}		
Kind	ProvidedPort	Interface	ComM_ChannelWakeup
Description	The numeric value for the port defined argument for ComMChannelId shall be derived from Configuration.		
Variation	{ecuc(ComM/ComMGeneral.ComMWakeupInhibitionEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel)}		

」()

8.7.5.4 ComM_modeLimitation

[SWS_ComM_01009]「

Name	modeLimitation		
Kind	ProvidedPort	Interface	ComM_ECUModeLimitation
Description	--		
Variation	{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true		

」()

8.7.5.5 ComM_UM

[SWS_ComM_01010]「

Name	UM_{user_name}		
Kind	ProvidedPort	Interface	ComM_CurrentMode
Description	The numeric value for the port defined argument for ComMUserIdentifier shall be derived from Configuration.		
Variation	user_name = {ecuc(ComM/ComMConfigSet/ComMUser.SHORT-NAME)}		

」()

8.7.5.6 ComM_UR

[SWS_ComM_01011]「

Name	UR_{user_name}		
Kind	ProvidedPort	Interface	ComM_UserRequest
Description	The numeric value for the port defined argument for ComMUserIdentifier shall be derived from Configuration.		

Variation	user_name = {ecuc(ComM/ComMConfigSet/ComMUser.SHORT-NAME)}

⌋()

8.7.6 ModeDeclarationGroups

8.7.6.1 ComMMode

[SWS_ComM_01012]⌈

Name	ComMMode
Kind	ModeDeclarationGroup
Initial mode	COMM_NO_COMMUNICATION
Modes	COMM_NO_COMMUNICATION
	COMM_SILENT_COMMUNICATION
	COMM_FULL_COMMUNICATION
Description	--

⌋()

9 Sequence diagrams

9.1 Transmission and Reception start (CAN)

Figure 9 shows the sequence for starting transmission and reception on CAN. The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.

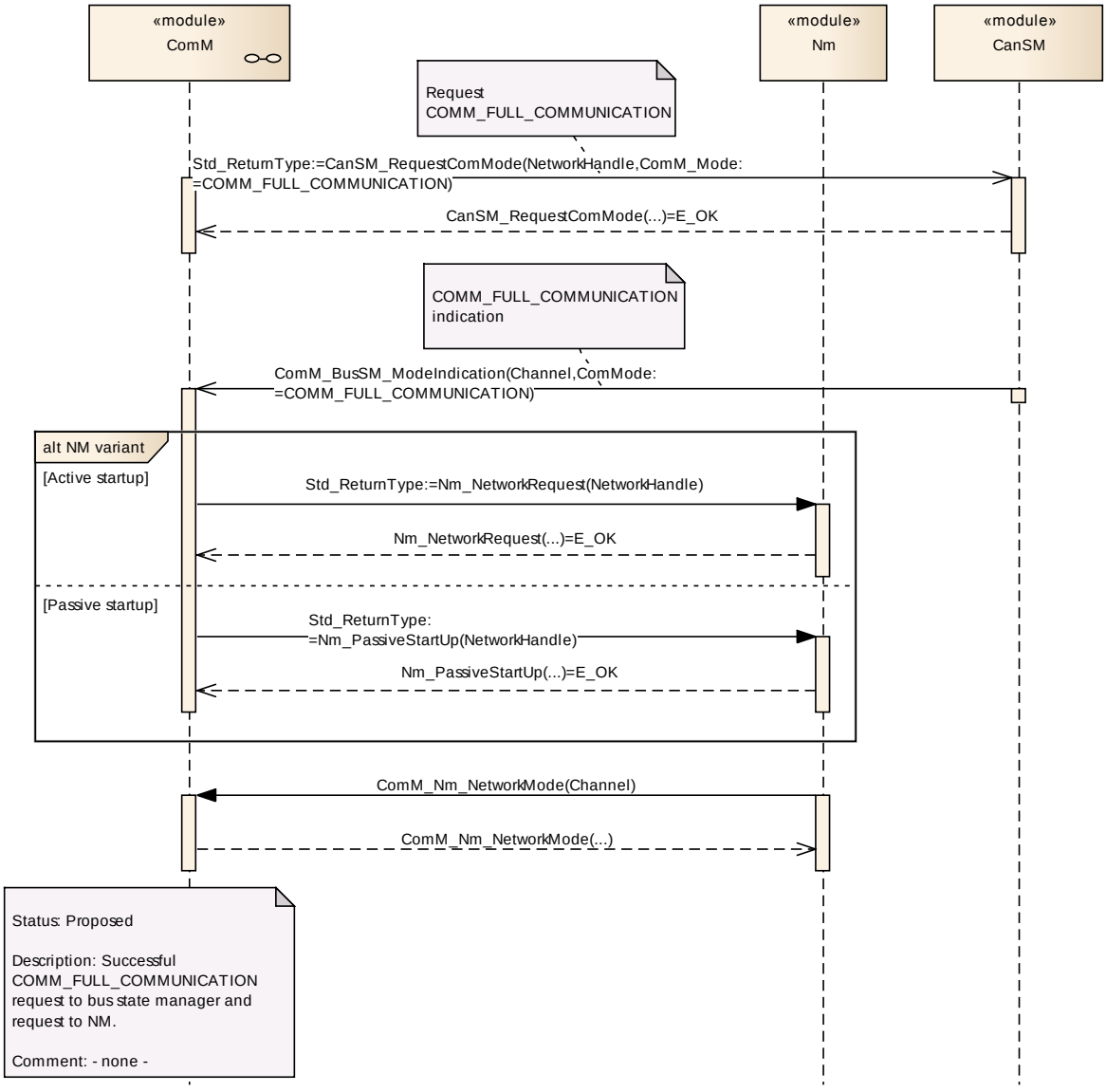

Figure 9: Starting transmission and reception on CAN

9.2 Passive Wake-up (CAN)

Figure 10 shows the behaviour after a wake-up indicated by the ECU State Manager module, or the Nm module for a CAN channel. The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.

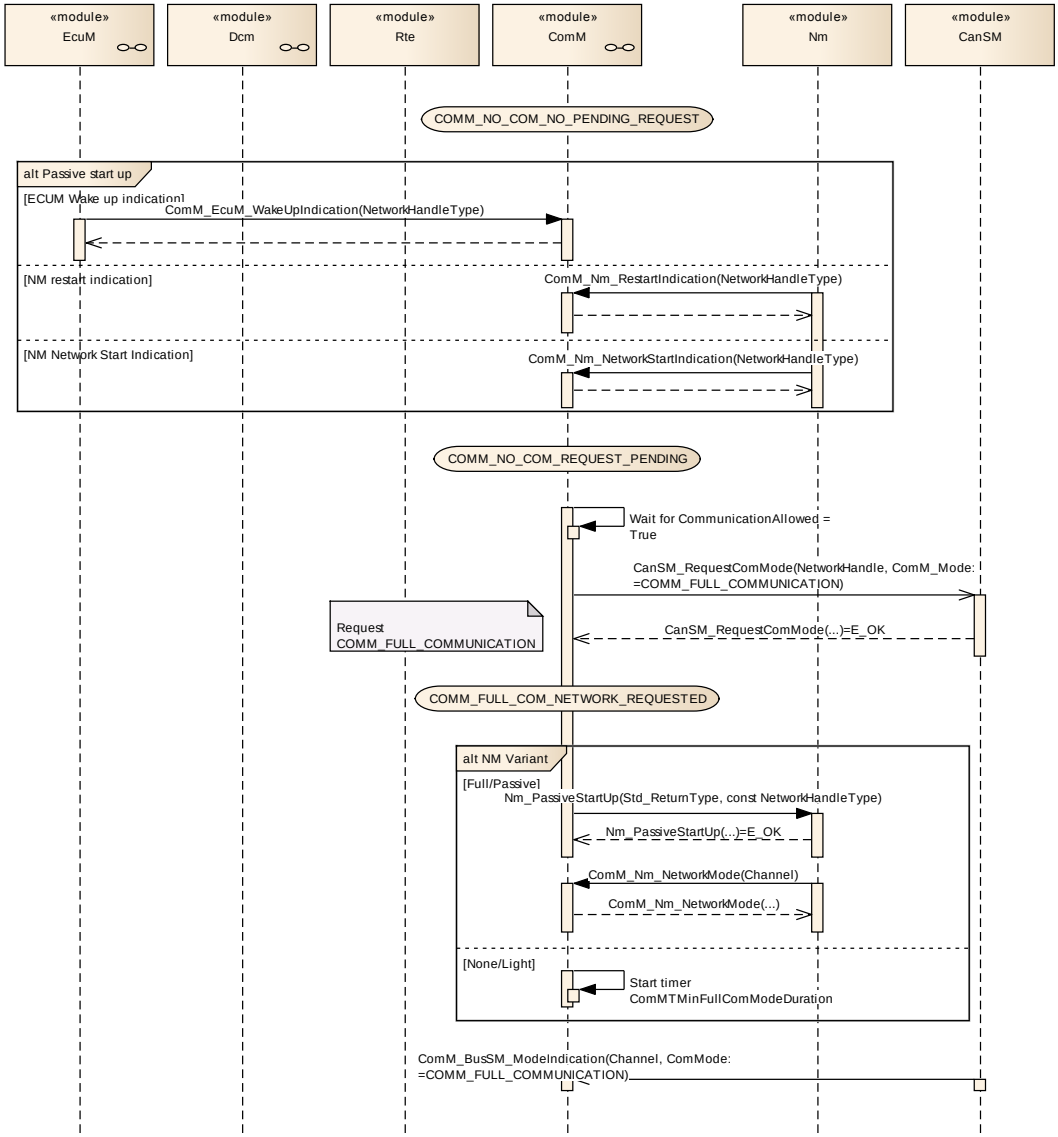

Figure 10: Reaction on a wake-up indicated by the ECU State Manager module

9.3 Network shutdown (CAN)

Figure 11 shows the possibilities to shutdown the CAN network. It can be either initiated if the last user releases his `COMM_FULL_COMMUNICATION` request or `ComM_LimitChannelToNoComMode(…)` (see SWS_ComM_00163) is called. The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.

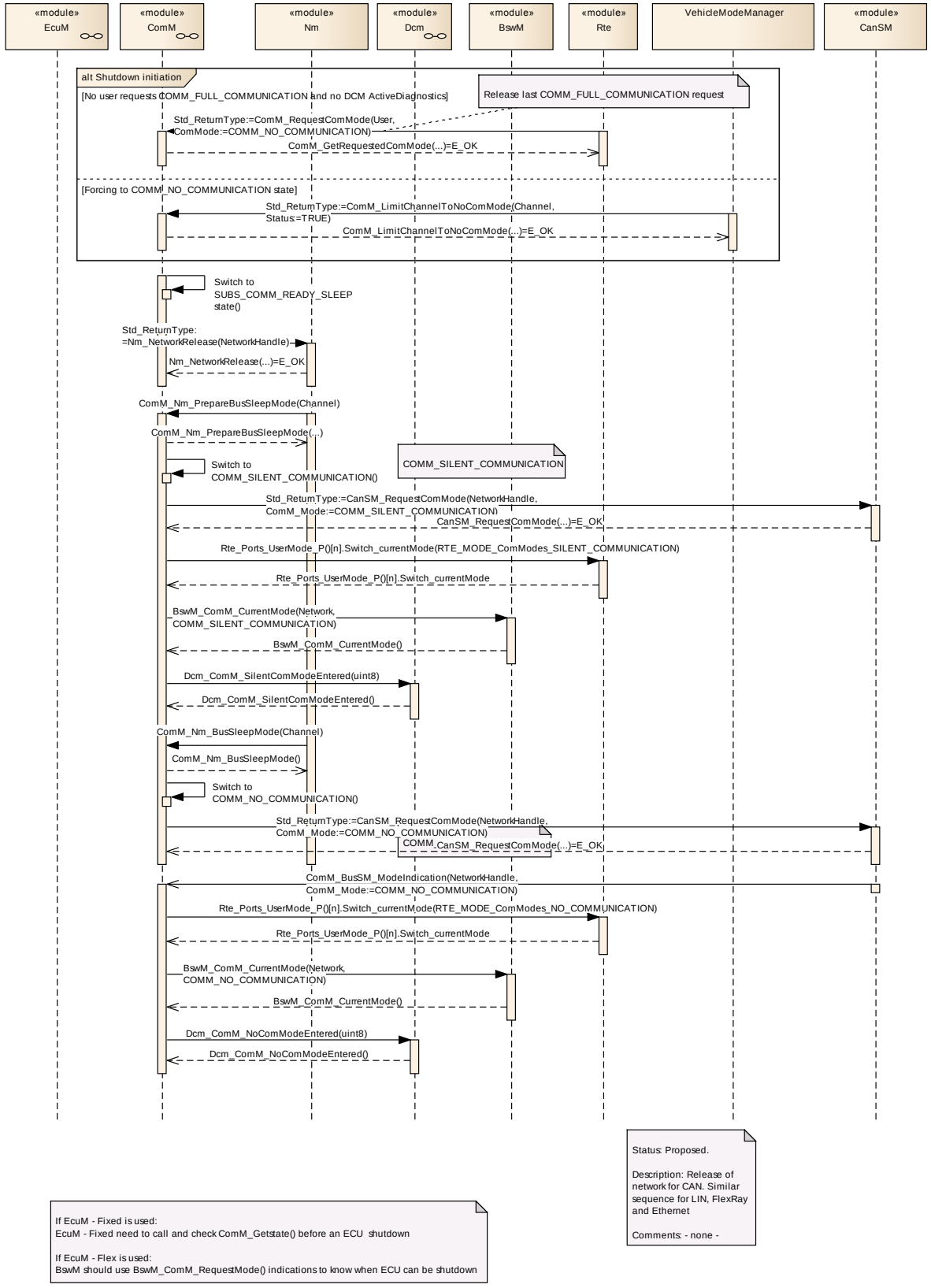

Figure 11: Network shutdown (CAN)

9.4 Communication request

Figure 12 shows the possibilities to start COMM_FULL_COMMUNICATION on CAN. It can be either initiated if a user requests COMM_FULL_COMMUNICATION request or DCM indicates *ComM_DCM_ActiveDiagnostic* (see SWS_ComM_00873). The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.

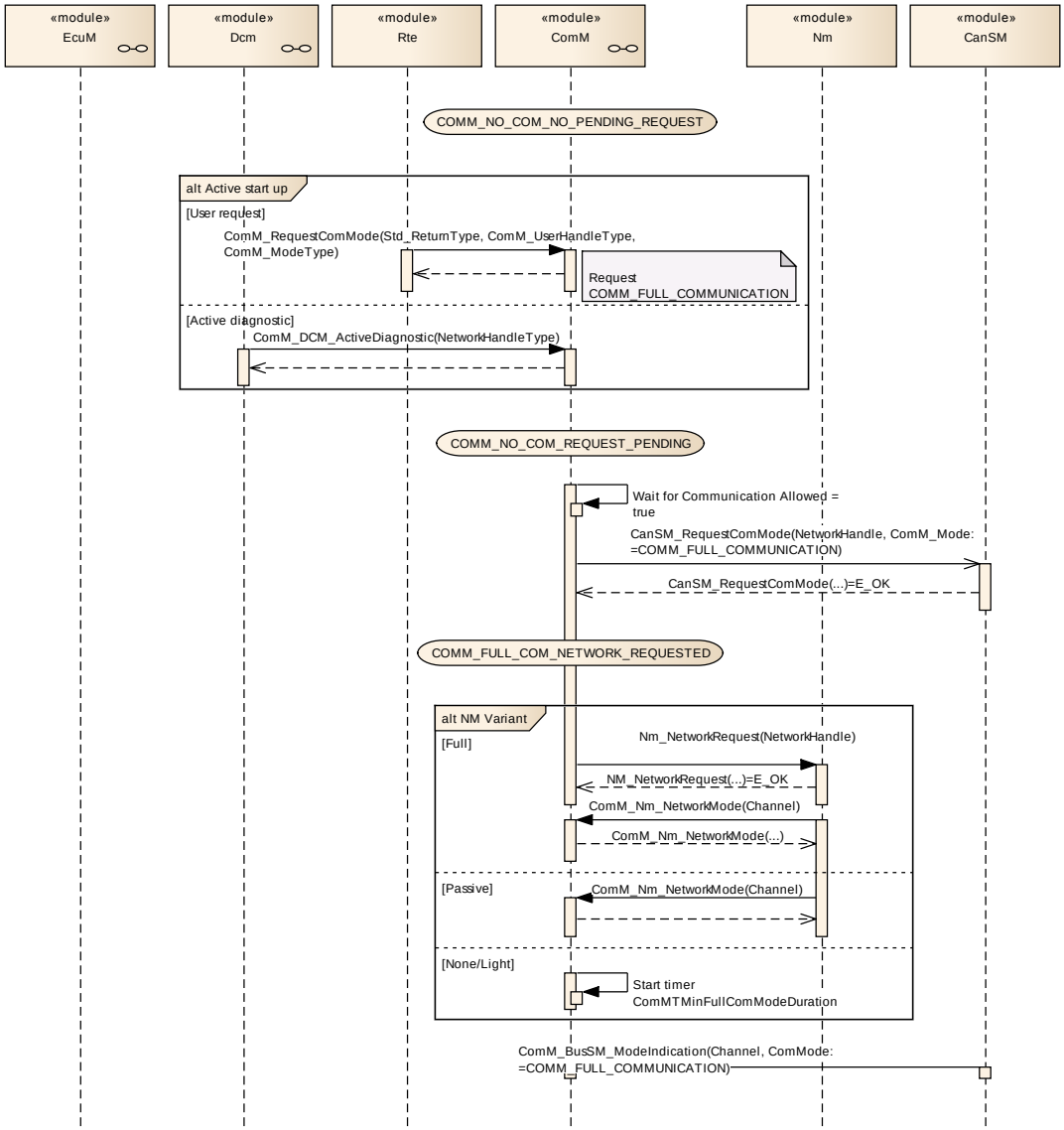

Figure 12: Request Communication

10 Configuration specification

In general, this chapter defines configuration parameters and their clustering into containers. In order to support the specification Chapter 10.1 describes fundamentals.

Chapter 10.2 specifies the structure (containers) and the parameters of the module Communication Manager Module.

Chapter 10.3 specifies published information of the Communication Manager Module.

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

[SWS_ComM_00419] ⌈The ComM module pre-compile time and link time configuration parameters shall be checked statically (at the latest during link time) for correctness.⌋(SRS_BSW_00167)

[SWS_ComM_00327] ⌈The ComM module configuration shall support the possibility to assign communication-channels to users by static configuration.⌋(SRS_ModeMgm_09133)

[SWS_ComM_00159] ⌈The ComM module configuration shall support to configure several communication channels to a user.⌋(SRS_ModeMgm_09090)

Rationale for SWS_ComM_00159: In a multi channel system each user can be assigned to one or more channels. If the user requests a mode, all channels assigned to this user, shall switch to the corresponding mode. All other channels shall not be affected.

[SWS_ComM_00160] ⌈ComMUsers shall be assignable to ComMChannels in combination with all ComMNmVariants except ComMNmVariant = PASSIVE.⌋()

[SWS_ComM_00161]⌈ ComMUsers shall be assignable to PNCs, which refer to ComMChannels in combination with all ComMNmVariants except ComMNmVariant = PASSIVE.⌋()

[SWS_ComM_00322] ⌈The ComM module configuration shall support configuration of bus type for each channel.⌋()

Rationale for SWS_ComM_00322: Interfaces for controlling the communication stack depends on the bus type.

[SWS_ComM_00464] ⌈The ComM module shall strictly separate configuration from implementation.⌋(SRS_BSW_00158)

Rationale for *SWS_ComM_00464*: Easy and clear configuration.

[SWS_ComM_00456] ⌈The ComM module pre-compile time and published configuration data, shall group and export the configuration data to a static configuration interface. The name of the interface shall be `ComM_Cfg.h`.⌋(SRS_BSW_00345)

10.2.1 VARIANT POST-BUILD

[SWS_ComM_00998] 「ComM shall support a variant called VARIANT POST-BUILD. The supported parameter shall be:

1. ComMPncEnabled

」()

10.2.2 VARIANT-PRE-COMPILE

[SWS_ComM_00549] 「The ComM module shall support a variant called VARIANT-PRE-COMPILE.」(SRS_BSW_00388)

10.2.3 ComM

Module Name	*ComM*
Module Description	Configuration of the ComM (Communications Manager) module.

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ComMConfigSet	1	This container is the base for a multiple configuration set.
ComMGeneral	1	General configuration parameters of the Communication Manager.

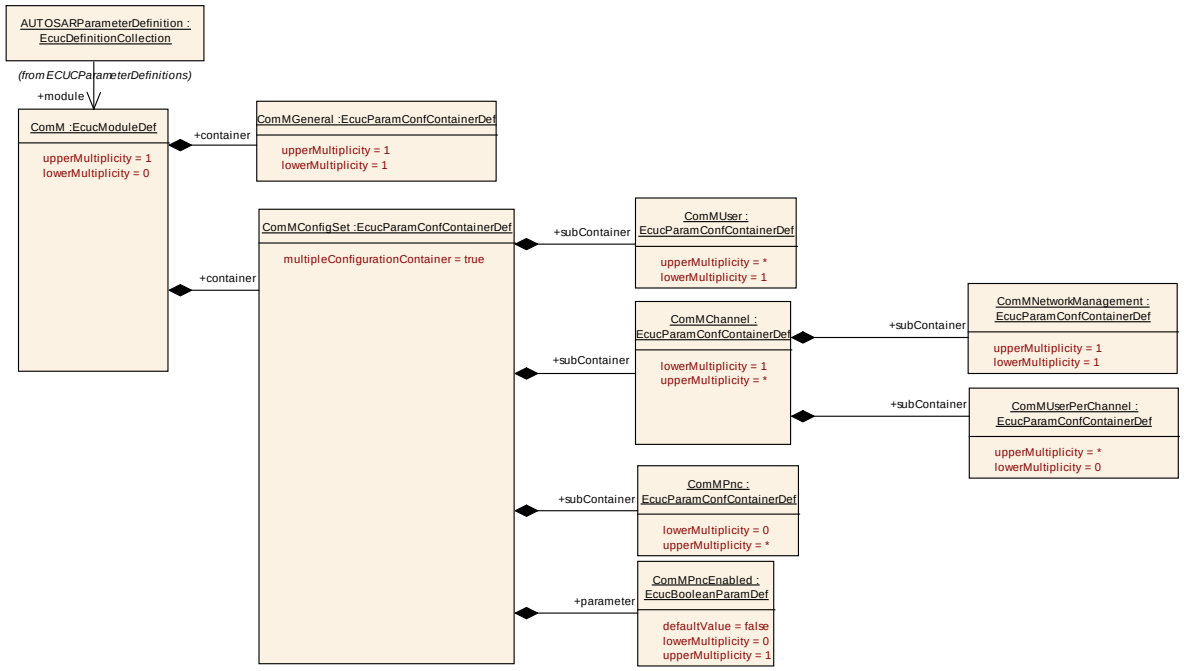

Figure 13: Configuration ComM

10.2.4 ComMGeneral

SWS Item	**ECUC_ComM_00554 :**
Container Name	ComMGeneral{CommunicationManagerConfiguration}
Description	General configuration parameters of the Communication Manager.
Configuration Parameters	

SWS Item	**ECUC_ComM_00555 :**		
Name	ComMDevErrorDetect {COMM_DEV_ERROR_DETECT}		
Description	Switches the Development Error Detection and Notification ON or OFF. 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: local		

SWS Item	**ECUC_ComM_00840 :**
Name	ComMDirectUserMapping {COMM_DIRECT_USER_MAPPING}
Description	If this parameter is set to true the configuration tool shall automatically create a ComMUser per ComMPnc and a ComMUser per ComMChannel. The shortName of the generated ComMUsers shall follow the following naming convention: PNCUser_ComMPncId, e.g. PNCUser_13 ChannelUser_ComMChannelId, e.g. ChannelUser_25 Restriction: ComMUser, which are created due to this configuration parameter, shall not be used by SWCs (only available for BswM).
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_ComM_00563 :**		
Name	ComMEcuGroupClassification {COMM_ECU_GROUP_CLASSIFICATION}		
Description	Defines whether a mode inhibition affects the ECU or not. Examples: 000: No mode inhibition can be activated 001: Wake up inhibition can be enabled Forcing into COMM_NO_COMMUNICATION mode shall be switched on if ComMNmVariant=PASSIVE.		
Multiplicity	1		
Type	EcucIntegerParamDef		
Range	0 .. 255		
Default value	3		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Shall be stored none volatile (value must be kept during a reset.). Can be changed during runtime with ComM_SetECUGroupClassification() thus the default values shall be set only once (first ECU initialization).		

SWS Item	**ECUC_ComM_00560 :**		
Name	ComMModeLimitationEnabled {COMM_MODE_LIMITATION_ENABLED}		
Description	true if mode limitation functionality shall be enabled. 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: local dependency: Shall be true if ComMNmVariant=COMM_PASSIVE		

SWS Item	**ECUC_ComM_00889 :**		
Name	ComMNmPassiveModeEnable {COMM_NM_PASSIVE_MODE_ENABLE}		
Description	Enables support of Passive Mode.		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00887 :**
Name	ComMPncGatewayEnabled {COMM_PNC_GW_ENABLED}
Description	Enables or disables support of Partial Network Gateway. False: Partial Networking Gateway is disabled True: Partial Networking Gateway 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		

SWS Item	**ECUC_ComM_00841 :**		
Name	ComMPncPrepareSleepTimer {COMM_T_PNC_PREPARE_SLEEP}		
Description	Time in seconds the PNC state machine shall wait in PNC_PREPARE_SLEEP.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. 63		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00839 :**		
Name	ComMPncSupport {COMM_PNC_SUPPORT}		
Description	Enables or disables support of partial networking. False: Partial Networking is disabled True: Partial Networking is enabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00558 :**		
Name	ComMResetAfterForcingNoComm {COMM_RESET_AFTER_FORCING_NO_COMM}		
Description	ComM shall perform a reset after entering "No Communication" mode because of an active mode limitation to "No Communication" mode. 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: local		

SWS Item	**ECUC_ComM_00695 :**		
Name	ComMSynchronousWakeUp {COMM_SYNCHRONOUS_WAKE_UP}		
Description	Wake up of one channel shall lead to a wake up of all channels if true. true: Enabled false: Disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00557 :**		
Name	ComMTMinFullComModeDuration {COMM_T_MIN_FULL_COM_MODE_DURATION}		
Description	Minimum time duration in seconds, spent in the COMM_FULL_COMMUNICATION sub-state COMM_FULL_COM_NETWORK_REQUESTED.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0.001 .. 65		
Default value	5		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00622 :**		
Name	ComMVersionInfoApi {COMM_VERSION_INFO_API}		
Description	Switches the possibility to read the published information with the service ComM_GetPublishedInformation(). true: Enabled false: Disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	true		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00559 :**		
Name	ComMWakeupInhibitionEnabled {COMM_WAKEUP_INHIBITION_ENABLED}		
Description	true if wake up inhibition functionality enabled. 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: local		

SWS Item	**ECUC_ComM_00783 :**		
Name	ComMGlobalNvMBlockDescriptor {COMM_GlobalNvMBlockDescriptor}		
Description	Reference to NVRAM block containing the none volatile data. If this parameter is not configured it means that no NVRam is used at all.		
Multiplicity	0..1		
Type	Symbolic name reference to [NvMBlockDescriptor]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Derived from NvM configuration		

No Included Containers

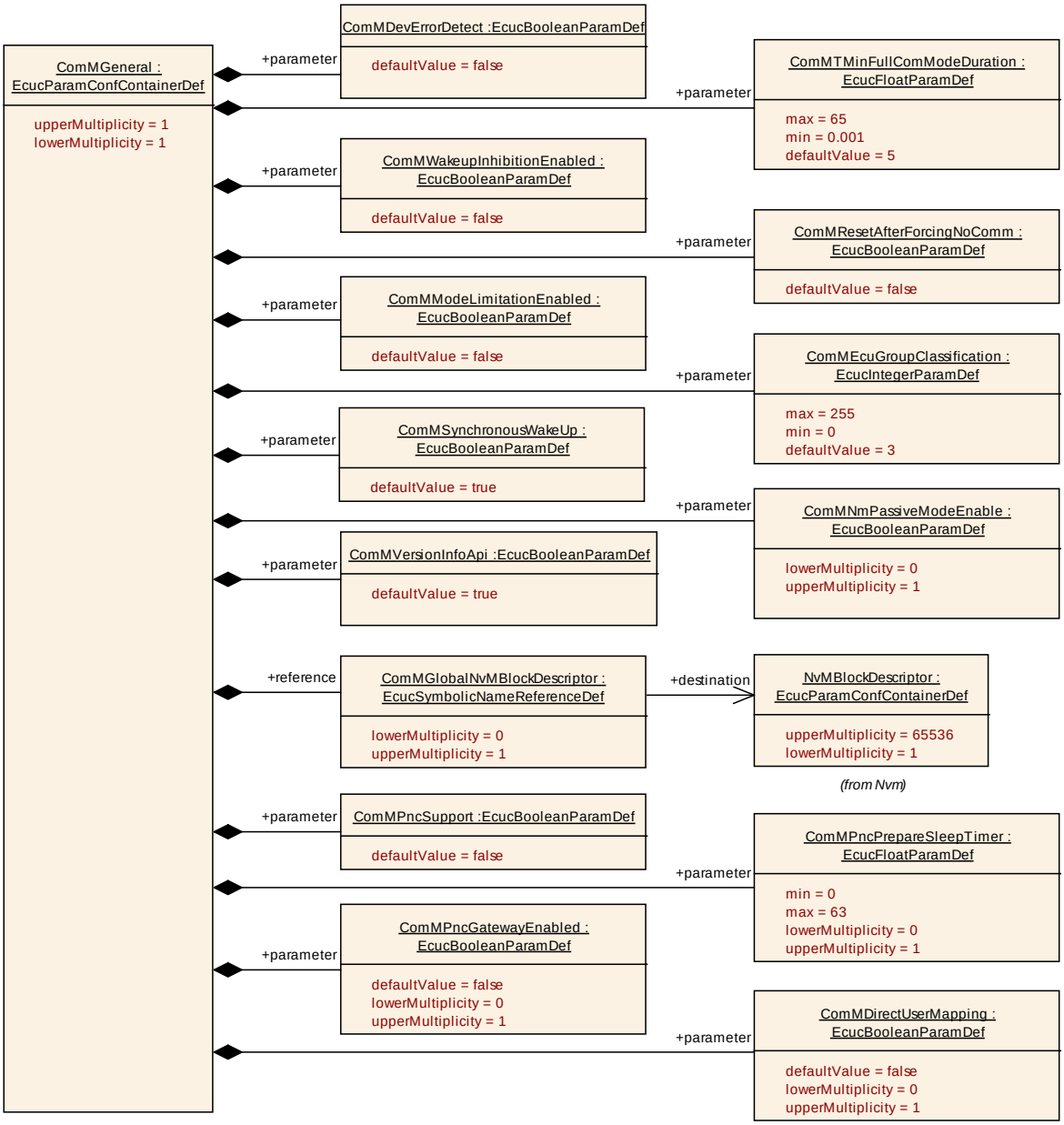

Figure 14: Configuration ComMGeneral

10.2.5 ComMConfigSet

SWS Item	**ECUC_ComM_00879 :**
Container Name	ComMConfigSet [Multi Config Container]
Description	This container is the base for a multiple configuration set.
Configuration Parameters	

SWS Item	**ECUC_ComM_00878 :**		
Name	ComMPncEnabled {COMM_PNC_ENABLED}		
Description	Defines whether in this configuration set the partial networking is enabled. true: Enabled false: Disabled		
Multiplicity	0..1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: ECU dependency: ComMPncSupport		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ComMChannel	1..*	This container contains the configuration (parameters) of the bus channel(s). The channel parameters shall be harmonized within the whole communication stack.
ComMPnc	0..*	This container contains the configuration of the partial network cluster (PNC).
ComMUser	1..*	This container contains a list of identifiers that are needed to refer to a user in the system which is designated to request Communication modes.

10.2.6 ComMUser

SWS Item	**ECUC_ComM_00653 :**
Container Name	ComMUser{CommunicationManagerUser}
Description	This container contains a list of identifiers that are needed to refer to a user in the system which is designated to request Communication modes.
Configuration Parameters	

SWS Item	**ECUC_ComM_00654 :**		
Name	ComMUserIdentifier {COMM_USER}		
Description	An identifier that is needed to refer to a user in the system which is designated to request Communication Modes. ImplementationType: ComM_UserHandleType		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: EcuMUser: The concept of users is very similar to the concept of requestors in the ECU State Manager specification. These two parameters shall be harmonized during the configuration process.		

SWS Item	**ECUC_ComM_00786 :**
Name	ComMUserEcucPartitionRef
Description	Denotes in which "EcucPartition" the requester is executed. When the partition is stopped, the communication request shall be cancelled in the ComM

	to avoid a stay-awake situation of the bus due to a stopped partition.		
Multiplicity	0..1		
Type	Reference to [EcucPartition]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

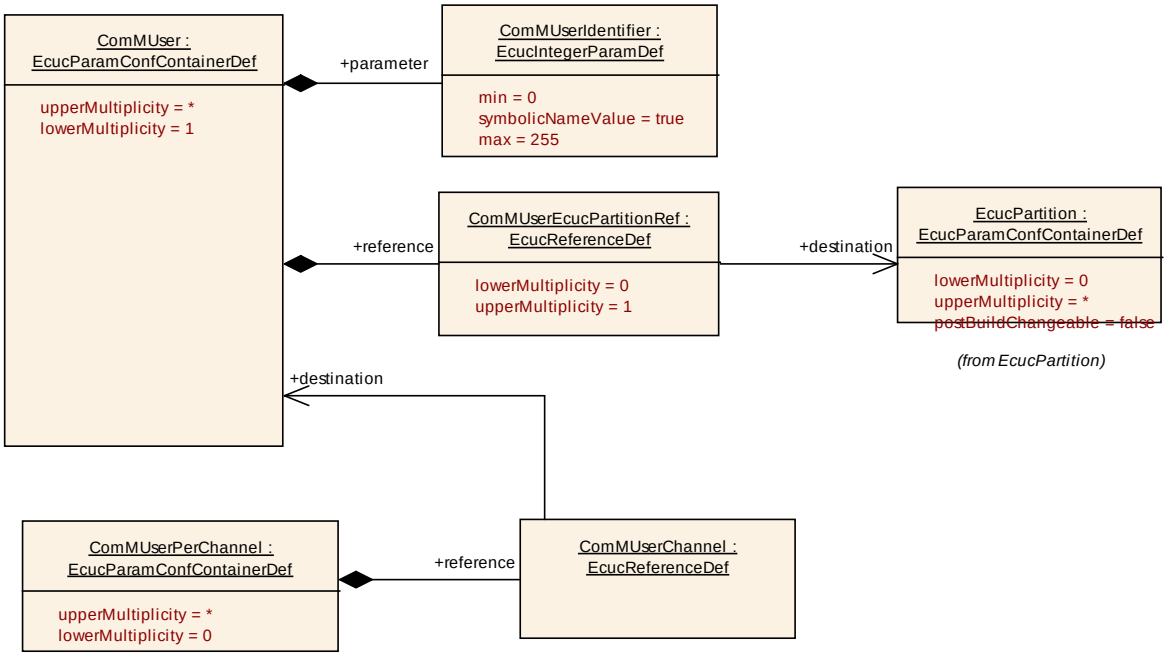

Figure 15: Configuration ComMUser

10.2.7 ComMChannel

SWS Item	**ECUC_ComM_00565 :**
Container Name	ComMChannel{Channel}
Description	This container contains the configuration (parameters) of the bus channel(s). The channel parameters shall be harmonized within the whole communication stack.
Configuration Parameters	

SWS Item	**ECUC_ComM_00567 :**	
Name	ComMBusType {COMM_BUS_TYPE}	
Description	Identifies the bus type of the channel.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	COMM_BUS_TYPE_CAN	--
	COMM_BUS_TYPE_CDD	--

	COMM_BUS_TYPE_ETH	--	
	COMM_BUS_TYPE_FR	--	
	COMM_BUS_TYPE_INTERNAL	--	
	COMM_BUS_TYPE_LIN	--	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_ComM_00888 :**		
Name	ComMCDDBusPrefix {COMM_CDD_BUS_PREFIX}		
Description	Prefix to be used for API calls to CDD.		
Multiplicity	0..1		
Type	EcucStringParamDef		
Default value	--		
maxLength	--		
minLength	--		
regularExpression	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Only applicable if ComMBusType equals COMM_BUS_TYPE_CDD.		

SWS Item	**ECUC_ComM_00635 :**		
Name	ComMChannelId {COMM_CHANNEL_ID}		
Description	Channel identification number of the corresponding channel.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	0 .. 255		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU dependency: Shall be harmonized with channel IDs of networkmanagement and the bus interfaces.		

SWS Item	**ECUC_ComM_00787 :**		
Name	ComMFullCommRequestNotificationEnabled {COMM_FULL_COMM_REQUEST_NOTIFICATION_ENABLED}		
Description	Defines if the optional SenderReceiver Port of Interface ComM_CurrentChannelRequest will be provided for this channel. True means enabled. False means disabled		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Shall be stored none volatile (value must be kept during a reset).		

SWS Item	**ECUC_ComM_00789 :**

Name	ComMGlobalNvmBlockDescriptor {COMM_NO_WAKEUP_INHIBITION_NVM_STORAGE}		
Description	If this parameter is set to "true", the NoWakeUp inhibition state of the channel shall be stored (in some implementation specific way) in the block pointed to by ComMGlobalNvmBlockDescriptor.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: If the parameter is set to true, a valid Nvm block reference must be given in the (existing, i.e. multiplicity 1) ComMGlobalNvmBlockDescriptor pointing to a sufficiently big Nvm block.		

SWS Item	**ECUC_ComM_00556 :**		
Name	ComMMainFunctionPeriod {COMM_MAIN_FUNCTION_PERIOD}		
Description	Specifies the period in seconds that the MainFunction has to be triggered with. Comment: ComM scheduling shall be at least as fast as the communication stack and a schedule longer than 100ms makes no sense for communication.		
Multiplicity	1		
Type	EcucFloatParamDef		
Range	0.001 .. 0.1		
Default value	0.02		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_ComM_00571 :**		
Name	ComMNoCom {COMM_NO_COM}		
Description	Not allowed to change state of ComM channel to COMM_SILENT_COMMUNICATION or COMM_FULL_COMMUNICATION. true: Enabled - Not allowed to switch to Communication Modes above. false: Disabled - Allowed to switch Communication Modes above. Shall be possible to change parameter during runtime with ComM API's. ECU/All channels: ComM_LimitECUToNoComMode(). Separate channels: ComM_LimitChannelToNoComMode().		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: ComMModeLimitationEnabled		

SWS Item	**ECUC_ComM_00569 :**
Name	ComMNoWakeup {COMM_NO_WAKEUP}
Description	Defines if an ECU is not allowed to wake-up the channel. true: Enabled (not allowed to wake-up)) false: Disabled This is the default/init value of a runtime variable that can be changed during runtime using ComM_PreventWakeUp().

Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Shall be stored none volatile (value must be kept during a reset).		

SWS Item	**ECUC_ComM_00842 :**		
Name	ComMPncGatewayType {COMM_PNC_GW_TYPE}		
Description	Identifies the Partial Network Gateway behaviour of a ComMChannel.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	COMM_GATEWAY_TYPE_ACTIVE	--	
	COMM_GATEWAY_TYPE_PASSIVE	--	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

Included Containers		
Container Name	***Multiplicity***	***Scope / Dependency***
ComMNetworkManagement	1	This container contains the configuration parameters of the networkmanagement.
ComMUserPerChannel	0..*	This container contains a list of identifiers that are needed to refer to a user in the system which is linked to a channel.

[SWS_ComM_00690] ⌈Configuration parameter `ComMNoCom` (see ECUC_ComM_00571) need not to be evaluated in case `ComMModeLimitationEnabled` = FALSE = Disabled (see ECUC_ComM_00560) thus it can be removed in that case to reduce/optimize the configuration.⌋()

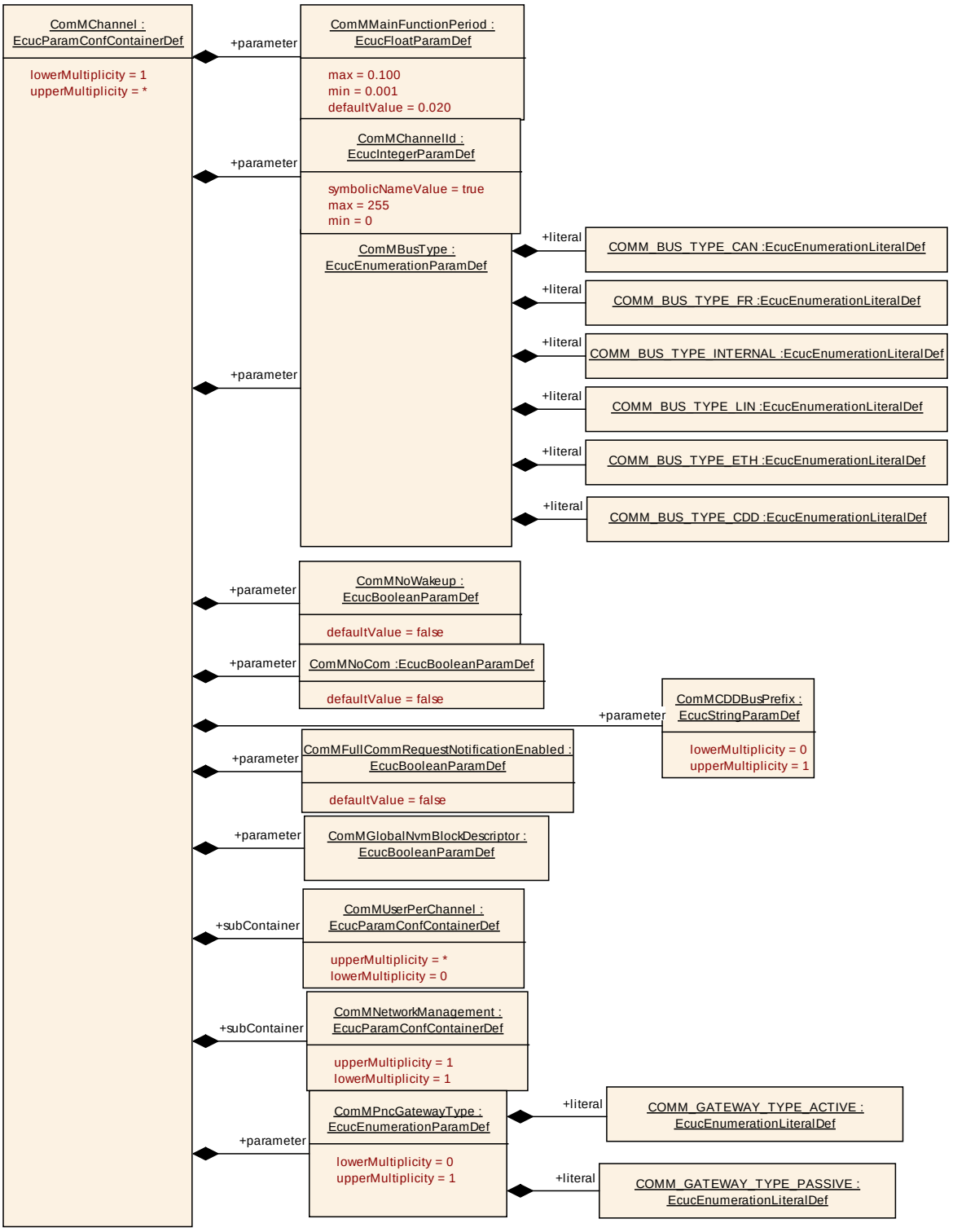

Figure 16: Configuration ComMChannel

10.2.8 ComMNetworkManagement

SWS Item	**ECUC_ComM_00607 :**

Container Name	ComMNetworkManagement{Networkmanagement}
Description	This container contains the configuration parameters of the networkmanagement.
Configuration Parameters	

SWS Item	**ECUC_ComM_00606 :**		
Name	ComMNmLightTimeout {COMM_NM_LIGHT_TIMEOUT}		
Description	Defines the timeout (in seconds) after COMM_FULL_COMMUNICATION sub-state COMM_FULL_COM_READY_SLEEP is left. The range shall be greater than 0.0 and less or equal to 255.0.		
Multiplicity	0..1		
Type	EcucFloatParamDef		
Range	0 .. 255		
Default value	10		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: Only used if ComMNmVariant is configured as ComMLight		

SWS Item	**ECUC_ComM_00568 :**		
Name	ComMNmVariant {COMM_NM_VARIANT}		
Description	Defines the functionality of the networkmanagement. Shall be harmonized with NM configuration.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	FULL	AUTOSAR NM available (default). (default)	
	LIGHT	No AUTOSAR NM available but functionality to shut down a channel.	
	NONE	No NM available	
	PASSIVE	AUTOSAR NM running in passive mode available.	
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: If ComMNmPassiveModeEnable is enabled the values are limited to LIGHT, NONE, PASSIVE; If ComMNmPassiveModeEnable is disabled the values are limited to LIGHT, NONE, FULL.; ComMNmVariant shall be NONE if ComMBusType = COMM_BUS_TYPE_INTERNAL		

SWS Item	**ECUC_ComM_00886 :**		
Name	ComMPncNmRequest {COMM_PNC_NM_REQUEST}		
Description	If this parameter equals true then every time a FULL Communication is requested due to a change in the PNC state machine to PNC_REQUESTED Nm shall be called using the API Nm_NetworkRequest.		
Multiplicity	1		
Type	EcucBooleanParamDef		
Default value	false		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: It shall only be possible to set ComMPncNmRequest to TRUE, if ComMNmVariant is FULL.		

No Included Containers

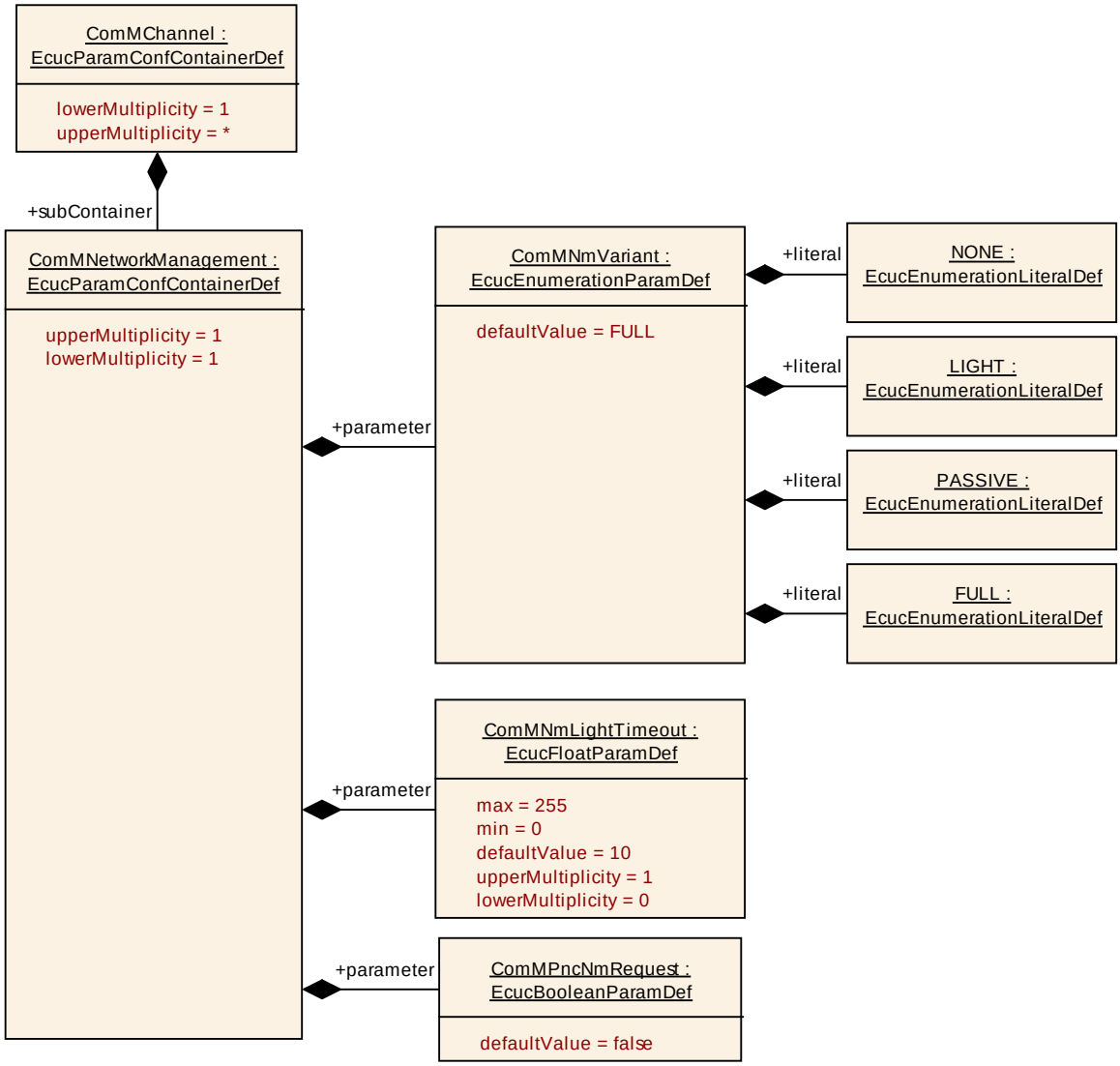

Figure 17: Configuration ComMNetworkManagement

10.2.9 ComMUserPerChannel

SWS Item	**ECUC_ComM_00657 :**
Container Name	ComMUserPerChannel{UserPerChannel}
Description	This container contains a list of identifiers that are needed to refer to a user in the system which is linked to a channel.
Configuration Parameters	

SWS Item	**ECUC_ComM_00658 :**
Name	ComMUserChannel

Description	Reference to the ComMUser that corresponds to this channel user. ImplementationType: COMM_UserHandleType		
Multiplicity	1		
Type	Reference to [ComMUser]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

No Included Containers

10.2.10 ComMPnc

SWS Item	**ECUC_ComM_00843 :**
Container Name	ComMPnc
Description	This container contains the configuration of the partial network cluster (PNC).
Configuration Parameters	

SWS Item	**ECUC_ComM_00874 :**		
Name	ComMPncId {COMM_PNC_ID}		
Description	Partial network cluster identification number.		
Multiplicity	1		
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)		
Range	8 .. 63		
Default value	--		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

SWS Item	**ECUC_ComM_00880 :**		
Name	ComMChannelPerPnc		
Description	Reference to the ComMChannel that is required for this PNC. ImplementationType: NetworkHandleType		
Multiplicity	1..*		
Type	Reference to [ComMChannel]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

SWS Item	**ECUC_ComM_00876 :**		
Name	ComMUserPerPnc		
Description	Reference to the ComMUsers that correspond to this PNC. ImplementationType: COMM_UserHandleType		
Multiplicity	0..*		
Type	Reference to [ComMUser]		
ConfigurationClass	***Pre-compile time***	X	VARIANT-PRE-COMPILE
	Link time	--	
	Post-build time	X	VARIANT-POST-BUILD
Scope / Dependency	scope: local		

Included Containers

Container Name	Multiplicity	Scope / Dependency
ComMPncComSignal	0..*	Represents the PncComSignals which are used to communicate the EIRA and ERA status of this PNC.

10.2.11 ComMPncComSignal

SWS Item	ECUC_ComM_00881 :
Container Name	ComMPncComSignal
Description	Represents the PncComSignals which are used to communicate the EIRA and ERA status of this PNC.
Configuration Parameters	

SWS Item	ECUC_ComM_00885 :		
Name	ComMPncComSignalDirection		
Description	Indicates the communication direction of this PncComSignal.		
Multiplicity	1		
Type	EcucEnumerationParamDef		
Range	RX	--	
	TX	--	
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local		

SWS Item	ECUC_ComM_00883 :		
Name	ComMPncComSignalKind		
Description	Indicates whether this PncComSignal represents EIRA or ERA PNC information. This parameter ComMPncComSignalKind is optional and shall be ignored when ComMPncComSignalDirection equals TX.		
Multiplicity	0..1		
Type	EcucEnumerationParamDef		
Range	EIRA	--	
	ERA	--	
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: local dependency: This parameter ComMPncComSignalKind shall be ignored when ComMPncComSignalDirection equals TX.		

SWS Item	ECUC_ComM_00884 :		
Name	ComMPncComSignalChannelRef		
Description	Reference to the ComMChannel which is used to determine whether this PncComSignal shall participate in the active or passive role (via the parameter ComMPncGatewayType of the ComMChannel). This information may be available by following the ComMPncComSignalRef and analyse the Com configuration as well.		
Multiplicity	0..1		
Type	Reference to [ComMChannel]		
ConfigurationClass	Pre-compile time	X	All Variants
	Link time	--	

	Post-build time	--	
Scope / Dependency	scope: local dependency: ComMPncGatewayEnabled		

SWS Item	**ECUC_ComM_00882 :**		
Name	ComMPncComSignalRef		
Description	Reference to the ComSignal which is used to transport the partial network channel request information.		
Multiplicity	1		
Type	Symbolic name reference to [ComSignal]		
ConfigurationClass	***Pre-compile time***	X	All Variants
	Link time	--	
	Post-build time	--	
Scope / Dependency	scope: ECU		

No Included Containers

10.3 Published information

[SWS_ComM_00418] ⌈The version information in the module header and source files shall be validated and consistent (e.g. by comparing the version information in the module header and source files with a pre-processor macro).⌋(SRS_BSW_00004)

11 Not applicable requirements

[SWS_ComM_00499]「These requirements are not applicable to this specification.」
(SRS_BSW_00005, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00314, SRS_BSW_00325, SRS_BSW_00326, SRS_BSW_00341, SRS_BSW_00343, SRS_BSW_00344, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00375, SRS_BSW_00378, SRS_BSW_00398, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00413, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, BSW00431, SRS_BSW_00432, SRS_BSW_00433, BSW00434, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00439)