Document Title	Specification of Communication Manager
Document Category	Software specification (SWS)
Document UID	79
Document Status	published
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	R26-11

Document Change History			
Date	**Release**	**Changed by**	**Description**
2026-11-26	R26-11	AUTOSAR Release Management	• Removed extended production errors • Removed ComM_GetVersionInfo API • Harmonized term for PNC request bit vector • Clarified handling for corrupted NV data
2025-11-27	R25-11	AUTOSAR Release Management	• Moved ComMPncEnabled from ComMConfigSet to ComMChannel • Reworked chapter of communication inhibition • Added missing SRS uptraces • Minor bug fixes
2024-11-27	R24-11	AUTOSAR Release Management	• Introduced validation findings of concept "ReworkOfPNCrelatedComMandNM handling (part2) • Harmonized names of possible erros at service interfaces • Minor bug fixes
2023-11-23	R23-11	AUTOSAR Release Management	• Document structure rework • Introduced validation findings of concept "ReworkOfPNCrelatedComMandNM handling (part2)"

△

2022-11-24	R22-11	AUTOSAR Release Management	• Introduced validation findings of concept "ReworkOfPNCrelatedComMandNM handling (part2)" • Introduced validation findings of concept "VNSM (part1)" • Removed obsolete marked requirements of concept "ReworkOfPNCrelatedComMandNM handling (part1/part2)" and "EthernetWakeOnDataLine (part1)"
2021-11-25	R21-11	AUTOSAR Release Management	• Introduced dedicated APIs to synchronize the PNC status with Nm and set the usage of ComSignals to obsolete • Introduced ComMChannelPerTxOnlyPnc to support transmission-only PNCs • Set requirements to valid which relates to forward an wake up request if an PNC is actively requested • Re-worked the service interfaces to support the Pn learning phase
2020-11-30	R20-11	AUTOSAR Release Management	• Added optional feature for dynamic PNC-to-channel mapping • Added optional handling to transfer kind of communication request (either active or passive) to lower layers • Extend ComM service interface ComM_GetCurrentComMode, to obtain the PNC state of the mapped ComMUser • Added restriction for ComM users according the assignment of managed and managing channels
2019-11-28	R19-11	AUTOSAR Release Management	• Introduce handling of PNC coordinator if serval ComM channels have the same PNC assignment but PncGatewayTypeEnum is set to "none." • Enabled ComM to be used for BSW distribution (multicore use case) • Minor corrections • Changed Document Status from Final to published

▽

△

2018-10-31	4.4.0	AUTOSAR Release Management	• Introduce "managing" and "managed" ComM channels • Remove relations to EcuMfixed completely • Minor corrections
2017-12-08	4.3.1	AUTOSAR Release Management	• Clarification regarding communication inhibition and bus wake up inhibition
2016-11-30	4.3.0	AUTOSAR Release Management	• Added the possiblity to switch ehternet switch ports according to ComM channel request / release • Added the wake up handling in case of a ECU which is controlling a Ethernet switch and using PNCs. • Minor corrections
2015-07-31	4.2.2	AUTOSAR Release Management	• Chapter added to explain partial network usecase • Minor corrections
2014-10-31	4.2.1	AUTOSAR Release Management	• Release of PNC related FULL_COM request already upon leaving PNC_REQUESTED • Several clarifications • Minor corrections
2014-03-31	4.1.3	AUTOSAR 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"
2013-10-31	4.1.2	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

▽

△

2013-03-15	4.1.1	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
2011-12-22	4.0.3	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
2009-12-18	4.0.1	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
2010-02-02	3.1.4	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
2008-08-13	3.1.1	AUTOSAR Administration	• Legal disclaimer revised

▽

△

2007-07-24	2.1.18	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
2007-01-24	2.1.19	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
2005-05-31	1.0	AUTOSAR Administration	• Initial Release

Disclaimer

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

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

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

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

The word AUTOSAR and the AUTOSAR logo are registered trademarks.

Table of Contents

1 Introduction and functional overview 14
2 Acronyms and definitions 15
3 Related documentation 17
3.1 Input documents & related standards and norms . . . 17
3.2 Related specification . . . 18
4 Constraints and assumptions 19
4.1 Limitations . . . 19
4.2 Applicability to car domains . . . 19
5 Dependencies to other modules 20
5.1 File structure . . . 20
5.1.1 Code file structure . . . 20
5.2 AUTOSAR Runtime Environment (RTE) . . . 21
5.3 ECU State Manager (EcuM) . . . 21
5.4 Basic Software Mode Manager (BswM) . . . 21
5.5 NVRAM Manager . . . 21
5.6 Diagnostic Communication Manager (DCM) . . . 22
5.7 LIN State Manager . . . 22
5.8 CAN State Manager . . . 22
5.9 FlexRay State Manager . . . 22
5.10 Ethernet State Manager . . . 22
5.11 Network Management (NM) . . . 23
5.12 Default Error Tracer (DET) . . . 23
6 Requirements Tracing 24
7 Functional specification 30
7.1 Partial Network Cluster Management . . . 32
7.1.1 Overview . . . 32
7.1.2 Partial Network Cluster Management Functionality . . . 33
7.1.3 ComM PNC state machine . . . 35
7.1.3.1 Behavior in PNC main state COMM_PNC_NO_COMMUNICATION 38
7.1.3.2 On entry of PNC main state COMM_PNC_NO_COMMUNICATION from PowerOff . . . 39
7.1.3.3 Behavior in PNC main state COMM_PNC_FULL_COMMUNICATION 40
7.1.3.4 On entry of PNC sub state COMM_PNC_REQUESTED . . . 40
7.1.3.5 Behavior in PNC sub state COMM_PNC_REQUESTED . . . 42
7.1.3.6 On entry PNC sub state COMM_PNC_READY_SLEEP . . . 47
7.1.3.7 Behavior in PNC sub state COMM_PNC_READY_SLEEP . . . 47
7.1.3.8 On entry of PNC sub state COMM_PNC_PREPARE_SLEEP . . . 48
7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP . . . 49

7.1.4 PNC Gateway . . . 50
7.1.4.1 Support for not coordinated PNCs assigned to multiple channels 51
7.1.4.2 Active PNC Gateway . . . 52
7.1.4.3 Passive PNC Gateway . . . 53
7.1.4.4 Synchronized PNC shutdown . . . 53
7.1.4.5 Support for multiple top-level PNC coordinators . . . 55
7.1.5 Dynamic PNC-to-channel-mapping (optional) . . . 58
7.1.5.1 Update PNC-to-channel-mapping . . . 59
7.1.5.2 PNC Membership Forwarding . . . 60
7.1.6 Partial Networking Configuration Hints . . . 60
7.1.6.1 Bi-directional PNC handling . . . 61
7.1.6.2 Uni-directional PNC handling . . . 61
7.1.6.3 Transmission only PNC handling . . . 62
7.2 ComM channel state machine . . . 63
7.2.1 ComM managed and managing channels . . . 69
7.2.2 Behavior in state COMM_NO_COMMUNICATION . . . 69
7.2.2.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state . . . 71
7.2.2.2 COMM_NO_COM_REQUEST_PENDING sub-state . . . 73
7.2.3 Behavior in state COMM_SILENT_COMMUNICATION . . . 73
7.2.4 Behavior in state COMM_FULL_COMMUNICATION . . . 74
7.2.4.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state . . . 76
7.2.4.2 COMM_FULL_COM_READY_SLEEP sub-state . . . 79
7.3 ComM User to PNC Relations . . . 81
7.4 Extended functionality . . . 84
7.4.1 Communication inhibition . . . 84
7.4.1.1 Manage communication inhibition . . . 85
7.4.1.2 Bus wake up inhibition . . . 87
7.4.1.3 Limit to COMM_NO_COMMUNICATION mode . . . 89
7.4.1.4 Examples for communication inhibition function . . . 90
7.5 Bus communication management . . . 94
7.6 Network management dependencies . . . 94
7.7 Bus error management . . . 96
7.7.1 Network Start Indication . . . 96
7.8 Test support requirements . . . 96
7.8.1 Inhibited Full Communication Request Counter . . . 96
7.9 Error Classification . . . 97
7.9.1 Development Errors . . . 97
7.9.2 Runtime Errors . . . 98
7.9.3 Production Errors . . . 98
7.10 Communication Manager Module Services . . . 98
7.10.1 Architecture . . . 98
7.10.2 Use Cases . . . 99
7.10.2.1 SW-Cs does not care about the ComM module at all . . . 99

7.10.2.2 SW-Cs only cares about the state of its communication system 99
7.10.2.3 SW-Cs explicitly wants to take influence on its communication state . . . 101
7.10.2.4 SW-C wants to interact directly with physical channels activate ECU Mode Limitation . . . 102
7.10.3 Specification of Ports and Port Interfaces . . . 103
7.10.3.1 Types used by the interfaces . . . 103
7.10.3.2 Ports and Port Interface for User Requests . . . 103
7.10.3.3 Ports and Port Interfaces for the current mode of the Communication Manager Module . . . 104
7.10.3.4 Ports and Port Interfaces for the ComM users currently requesting COMM_FULL_COMMUNICATION . . . 104
7.10.3.5 Ports and Port Interface for ECU Mode Limitation . . . 105
7.10.3.6 Ports and Port Interface for Channel Wake up . . . 106
7.10.3.7 Ports and Port Interface for interface Channel Limitation . . . 106
7.10.3.8 Definition of the Service of the Communication Manager Module 107
7.10.4 Runnables and Entry points . . . 108
7.10.4.1 Internal behaviour . . . 108
7.10.4.2 Header file to be included by the Communication Manager Module . . . 110
7.11 Multicore Distribution . . . 110
7.12 Non functional requirements . . . 111
7.13 Security Events . . . 111
8 API specification 112
8.1 Imported types . . . 112
8.1.1 Standard types . . . 112
8.2 Type definitions . . . 113
8.2.1 ComM_InitStatusType . . . 113
8.2.2 ComM_PncModeType . . . 113
8.2.3 ComM_StateType . . . 114
8.2.4 ComM_ConfigType . . . 114
8.3 Function definitions . . . 115
8.3.1 ComM_Init . . . 115
8.3.2 ComM_DeInit . . . 116
8.3.3 ComM_GetStatus . . . 117
8.3.4 ComM_GetInhibitionStatus . . . 118
8.3.5 ComM_RequestComMode . . . 118
8.3.6 ComM_GetMaxComMode . . . 119
8.3.7 ComM_GetRequestedComMode . . . 120
8.3.8 ComM_GetCurrentComMode . . . 121
8.3.9 ComM_GetCurrentPNCComMode . . . 122
8.3.10 ComM_GetPncToChannelMapping . . . 123
8.3.11 ComM_UpdatePncToChannelMapping . . . 125

8.3.12 ComM_ResetPncToChannelMapping . . . 126
8.3.13 ComM_PnLearningRequest . . . 127
8.3.14 ComM_UpdatePncMembership . . . 129
8.3.15 ComM_PreventWakeUp . . . 130
8.3.16 ComM_LimitChannelToNoComMode . . . 131
8.3.17 ComM_LimitECUToNoComMode . . . 131
8.3.18 ComM_ReadInhibitCounter . . . 132
8.3.19 ComM_ResetInhibitCounter . . . 133
8.3.20 ComM_SetECUGroupClassification . . . 134
8.3.21 AUTOSAR Network Management Interface . . . 134
8.3.21.1 ComM_Nm_NetworkStartIndication . . . 134
8.3.21.2 ComM_Nm_NetworkMode . . . 135
8.3.21.3 ComM_Nm_PrepareBusSleepMode . . . 135
8.3.21.4 ComM_Nm_BusSleepMode . . . 136
8.3.21.5 ComM_Nm_RestartIndication . . . 136
8.3.21.6 ComM_Nm_RepeatMessageLeftIndication . . . 137
8.3.21.7 ComM_Nm_PncLearningBitIndication . . . 137
8.3.21.8 ComM_Nm_ForwardSynchronizedPncShutdown . . . 138
8.3.21.9 ComM_Nm_UpdateEIRA . . . 138
8.3.21.10 ComM_Nm_UpdateERA . . . 139
8.3.22 AUTOSAR Diagnostic Communication Manager Interface . . . 139
8.3.22.1 ComM_DCM_ActiveDiagnostic . . . 139
8.3.22.2 ComM_DCM_InactiveDiagnostic . . . 140
8.3.23 AUTOSAR ECU State Manager Interface . . . 140
8.3.23.1 ComM_EcuM_WakeUpIndication . . . 140
8.3.23.2 ComM_EcuM_PNCWakeUpIndication . . . 141
8.3.24 AUTOSAR ECU State Manager and Basic Software Mode Manager Interface . . . 141
8.3.24.1 ComM_CommunicationAllowed . . . 141
8.3.25 Bus State Manager Interface . . . 142
8.3.25.1 ComM_<Bus>SM_ModeIndication . . . 142
8.3.25.2 ComM_<Bus>SM_BusSleepMode . . . 143
8.4 Scheduled functions . . . 143
8.4.1 ComM_MainFunction . . . 143
8.5 Expected interfaces . . . 144
8.5.1 Mandatory interfaces . . . 144
8.5.1.1 AUTOSAR NVRAM Manager module . . . 145
8.5.1.2 AUTOSAR Bus State Manager . . . 146
8.5.1.3 AUTOSAR Network Management Interface . . . 146
8.5.1.4 AUTOSAR Diagnostic Communication Manager Module . . . 147
8.5.1.5 AUTOSAR RTE interface provided by RTE to ComM for the SW-C . . . 147
8.5.1.6 Basic Software Mode Manager (BswM) . . . 148

8.5.2 Optional interfaces . . . 148
8.5.3 Callout interfaces . . . 149
8.5.4 Configurable interfaces . . . 149
8.6 Service Interfaces . . . 150
8.6.1 Sender-Receiver-Interfaces . . . 150
8.6.1.1 ComM_CurrentChannelRequest . . . 150
8.6.2 Client-Server-Interfaces . . . 150
8.6.2.1 ComM_ChannelLimitation . . . 150
8.6.2.2 ComM_ChannelWakeup . . . 151
8.6.2.3 ComM_ECUModeLimitation . . . 152
8.6.2.4 ComM_UserRequest . . . 153
8.6.2.5 ComM_PncToChannelMapping . . . 155
8.6.2.6 ComM_DynamicPncToChannelMapping . . . 157
8.6.3 Mode-Switch-Interfaces . . . 158
8.6.3.1 ComM_CurrentMode . . . 158
8.6.4 Implementation Data Types . . . 159
8.6.4.1 ComM_InhibitionStatusType . . . 159
8.6.4.2 ComM_ModeType . . . 159
8.6.4.3 ComM_UserHandleType . . . 160
8.6.4.4 ComM_UserHandleArrayType . . . 161
8.6.4.5 ComM_UserHandleSubArrayType . . . 161
8.6.5 Ports . . . 162
8.6.5.1 ComM_CL . . . 162
8.6.5.2 ComM_CR . . . 162
8.6.5.3 ComM_CW . . . 163
8.6.5.4 ComM_modeLimitation . . . 163
8.6.5.5 ComM_UM . . . 163
8.6.5.6 ComM_UR . . . 164
8.6.5.7 ComM_PncToChannelMapping . . . 164
8.6.5.8 ComM_DynamicPncToChannelMapping . . . 164
8.6.6 ModeDeclarationGroups . . . 165
8.6.6.1 ComMMode . . . 165
9 Sequence diagrams 166
9.1 Transmission and Reception start (CAN) . . . 166
9.2 Passive Wake-up (CAN) . . . 166
9.3 Network shutdown (CAN) . . . 168
9.4 Communication request . . . 170
9.5 Synchronized PNC shutdown . . . 171
10 Configuration specification 175
10.1 How to read this chapter . . . 175
10.2 Containers and configuration parameters . . . 175
10.2.1 ComM . . . 175

10.2.2 ComMGeneral . . . 177
10.2.3 ComMConfigSet . . . 183
10.2.4 ComMUser . . . 184
10.2.5 ComMChannel . . . 186
10.2.6 ComMNetworkManagement . . . 194
10.2.7 ComMUserPerChannel . . . 196
10.2.8 ComMPnc . . . 197
10.3 Published Information . . . 200
A Not applicable requirements 201
B Change history of AUTOSAR traceable items 202
B.1 Traceable item history of this document according to AUTOSAR Release R22-11 . . . 202
B.2 Traceable item history of this document according to AUTOSAR Release R23-11 . . . 202
B.2.1 Added Specification Items in R23-11 . . . 202
B.2.2 Changed Specification Items in R23-11 . . . 204
B.2.3 Deleted Specification Items in R23-11 . . . 204
B.2.4 Added Constraints in R23-11 . . . 204
B.2.5 Changed Constraints in R23-11 . . . 205
B.2.6 Deleted Constraints in R23-11 . . . 205
B.3 Traceable item history of this document according to AUTOSAR Release R24-11 . . . 205
B.3.1 Added Specification Items in R24-11 . . . 205
B.3.2 Changed Specification Items in R24-11 . . . 205
B.3.3 Deleted Specification Items in R24-11 . . . 206
B.3.4 Added Constraints in R24-11 . . . 206
B.3.5 Changed Constraints in R24-11 . . . 206
B.3.6 Deleted Constraints in R24-11 . . . 206
B.4 Traceable item history of this document according to AUTOSAR Release R25-11 . . . 206
B.4.1 Added Specification Items in R25-11 . . . 206
B.4.2 Changed Specification Items in R25-11 . . . 207
B.4.3 Deleted Specification Items in R25-11 . . . 207
B.4.4 Added Constraints in R25-11 . . . 208
B.4.5 Changed Constraints in R25-11 . . . 208
B.4.6 Deleted Constraints in R25-11 . . . 208
B.5 Traceable item history of this document according to AUTOSAR Release R26-11 . . . 208
B.5.1 Added Specification Items in R26-11 . . . 208
B.5.2 Changed Specification Items in R26-11 . . . 208
B.5.3 Deleted Specification Items in R26-11 . . . 211
B.5.4 Added Constraints in R26-11 . . . 212
B.5.5 Changed Constraints in R26-11 . . . 212

B.5.6 Deleted Constraints in R26-11 212

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:

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

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.

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.

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.

Offering the possibility to force an ECU that keeps the bus awake to the 'No Communication' mode (see section 7.4.1.3 for details).

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.

Further, the PNC extension allows users to request and keep awake a logical group of ECUs all over the network, a so-called "partial network cluster". The "PNC gateway" allows to span these (logical) network clusters over different, hierarchically structured physical busses and networks

2 Acronyms and definitions

The glossary below includes acronyms and abbreviations relevant to the Communication Manager module that are not included in the [1].

Abbreviation / Acronym:	Description:
BSW	Basic Software
BswM	Basic Software Mode Manager
ComM	Communication Manager
DCM	Diagnostic Communication Manager
Det	Default 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
OA TC10	Open Alliance TC10 specification (see [2])
IRA	Internal Request Array. This is a bit vector which contains the aggregated internal PNC requests per channel. (see also chapter 8.5.2 "Nm_UpdateIRA")
EIRA	External and Internal Request Array. This is a bit vector which contains the aggregated external and internal PNC requests
ERA	External Request Array. This is a bit vector which contains the aggregated external PNC requests. Each ComMChannel which has a `ComMPncGatewayType` set is has one corresponding ERA
ERAn	All External Request Arrays which are available in ComM, i.e. "n" ComMChannels where `ComMPncGatewayType` is set, result in "n" External Request Arrays in ComM

Table 2.1: Acronyms and abbreviations used in the scope of this Document

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 [3].
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
Managed channel	A ComM channel that is referenced exclusively from one other channel by ECUC parameter ComMManageReference.
Managing channel	A ComM channel that references 1..n other channels by ECUC parameter ComMManageReference.

Table 2.2: Definitions used in the scope of this Document

3 Related documentation

3.1 Input documents & related standards and norms

[1] Glossary
AUTOSAR_FO_TR_Glossary

[2] OPEN Sleep/Wake-up Specification for Automotive Ethernet
http://www.opensig.org/Automotive-Ethernet-Specifications/

[3] Specification of ECU State Manager
AUTOSAR_CP_SWS_ECUStateManager

[4] Basic Software Module Description Template
AUTOSAR_CP_TPS_BSWModuleDescriptionTemplate

[5] General Specification of Basic Software Modules
AUTOSAR_CP_SWS_BSWGeneral

[6] Specification of RTE Software
AUTOSAR_CP_SWS_RTE

[7] Specification of Default Error Tracer
AUTOSAR_CP_SWS_DefaultErrorTracer

[8] General Requirements on Basic Software Modules
AUTOSAR_CP_RS_BSWGeneral

[9] Requirements on Mode Management
AUTOSAR_CP_RS_ModeManagement

[10] Guide to Mode Management
AUTOSAR_CP_EXP_ModeManagementGuide

[11] System Template
AUTOSAR_CP_TPS_SystemTemplate

[12] Guide to BSW Distribution
AUTOSAR_CP_EXP_BSWDistributionGuide

[13] Specification of NVRAM Manager
AUTOSAR_CP_SWS_NVRAMManager

[14] Specification of LIN State Manager
AUTOSAR_CP_SWS_LINStateManager

[15] Specification of CAN State Manager
AUTOSAR_CP_SWS_CANStateManager

[16] Specification of FlexRay State Manager
AUTOSAR_CP_SWS_FlexRayStateManager

[17] Specification of Ethernet State Manager

AUTOSAR_CP_SWS_EthernetStateManager

[18] Specification of Network Management Interface
AUTOSAR_CP_SWS_NetworkManagementInterface

[19] Specification of Diagnostic Communication Manager
AUTOSAR_CP_SWS_DiagnosticCommunicationManager

[20] Specification of Basic Software Mode Manager
AUTOSAR_CP_SWS_BSWModeManager

3.2 Related specification

AUTOSAR provides a General Specification on Basic Software modules [5], which is also valid for Communication Manager.

Thus, the specification SWS BSW General shall be considered as additional and required specification for Communication 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 5.1:

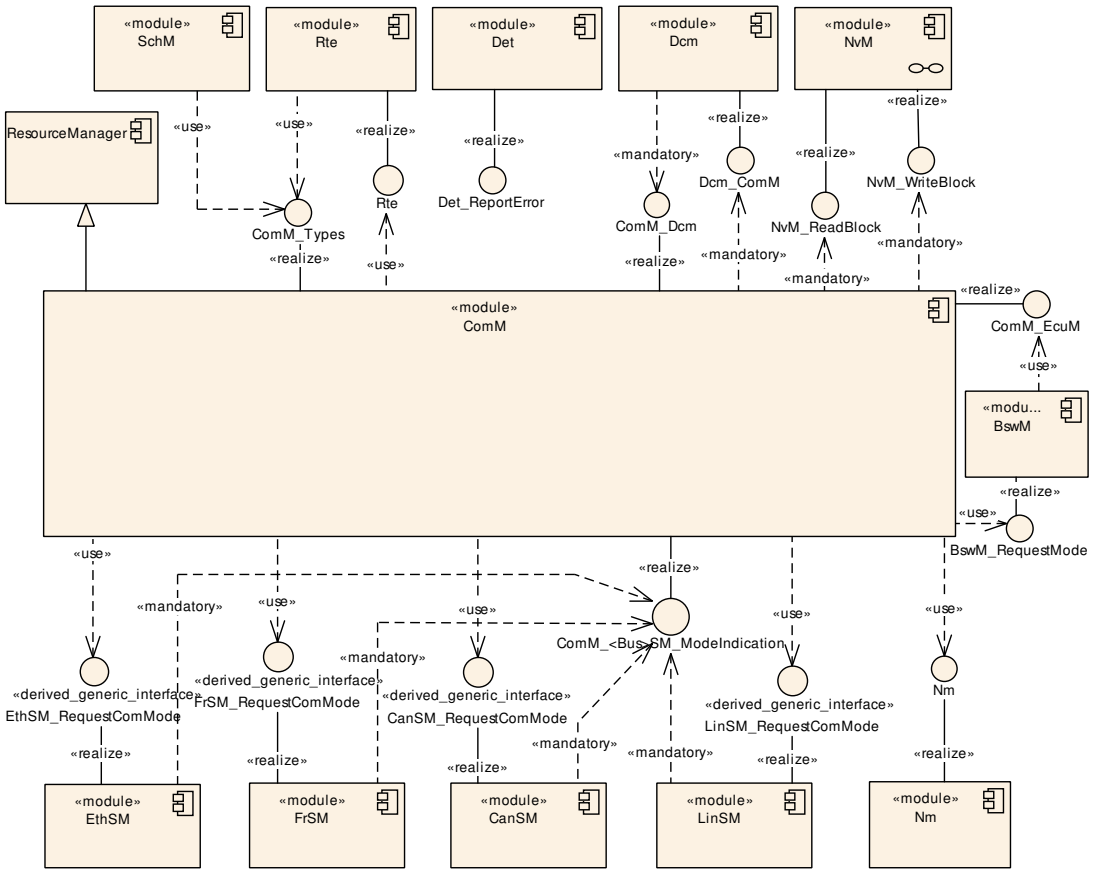


Figure 5.1: Communication Manager Module context view

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 Code file structure

For details refer to the Chapter 5.1.6 "Code file structure" in [5, SWS_BSWGeneral].

The code file structure is not defined within this specification completely. However to allow integration to other modules the following structure is needed.

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 [6]).

5.3 ECU State Manager (EcuM)

EcuM is responsible to validate wake-up events and send an indication to ComM if a wake-up is validated.

Communication allowed and shutdown of ECU is handled by EcuM together with BswM. (see [3] for details)

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]

Upstream requirements: SRS_ModeMgm_09251

⌈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). Additionally the status information about PNCs is exchanged via dedicated APIs between ComM and Nm.

5.12 Default Error Tracer (DET)

The DET (see [7])provides services for reporting development, runtime, and transient errors. (see Section 7.9)

6 Requirements Tracing

The following tables reference the requirements specified in [8] and [9] and links to the fulfillment of these. Please note that if column “Satisfied by” is empty for a specific requirement this means that this requirement is not fulfilled by this document.

Requirement	Description	Satisfied by
[SRS_BSW_00004]	All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files	[SWS_ComM_00418]
[SRS_BSW_00101]	The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function	[SWS_ComM_00146] [SWS_ComM_00668] [SWS_ComM_00793] [SWS_ComM_00864] [SWS_ComM_01098] [SWS_ComM_01099] [SWS_ComM_01100] [SWS_ComM_01101]
[SRS_BSW_00159]	All modules of the AUTOSAR Basic Software shall support a tool based configuration	[SWS_ComM_00464]
[SRS_BSW_00167]	All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks	[SWS_ComM_00419] [SWS_ComM_00464] [SWS_ComM_00690]
[SRS_BSW_00323]	All AUTOSAR Basic Software Modules shall check passed API parameters for validity	[SWS_ComM_00234]
[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_91027]
[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_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_00345]	BSW Modules shall support pre-compile configuration	[SWS_ComM_00464] [SWS_ComM_00620]
[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_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_00369]	All AUTOSAR Basic Software Modules shall not return specific development error codes via the API	[SWS_ComM_91027]
[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_00377]	A Basic Software Module can return a module specific types	[SWS_ComM_91027]

△

Requirement	Description	Satisfied by
[SRS_BSW_00385]	List possible error notifications	[SWS_ComM_00234]
[SRS_BSW_00386]	The BSW shall specify the configuration and conditions for detecting an error	[SWS_ComM_00234]
[SRS_BSW_00404]	BSW Modules shall support post-build configuration	[SWS_ComM_00162]
[SRS_BSW_00405]	BSW Modules shall support multiple configuration sets	[SWS_ComM_00162]
[SRS_BSW_00406]	API handling in uninitialized state	[SWS_ComM_00242] [SWS_ComM_00612] [SWS_ComM_00668] [SWS_ComM_00858] [SWS_ComM_00865]
[SRS_BSW_00414]	Init functions shall have a pointer to a configuration structure as single parameter	[SWS_ComM_00146]
[SRS_BSW_00441]	Naming convention for type, macro and function	[SWS_ComM_00863] [SWS_ComM_91027]
[SRS_BSW_00497]	Handling of corrupted NV data.	[SWS_ComM_00864] [SWS_ComM_01101]
[SRS_ModeMgm_-00049]	The Communication Manager shall initiate the wake-up and keep awake physical channels	[SWS_ComM_00261] [SWS_ComM_00383] [SWS_ComM_00390] [SWS_ComM_00391] [SWS_ComM_00392] [SWS_ComM_00402] [SWS_ComM_00792] [SWS_ComM_00869] [SWS_ComM_00870] [SWS_ComM_00929] [SWS_ComM_01069] [SWS_ComM_01071] [SWS_ComM_01086]
[SRS_ModeMgm_-09071]	It shall be possible to limit communication modes independently for each physical channel	[SWS_ComM_00066] [SWS_ComM_00085] [SWS_ComM_00124] [SWS_ComM_00163] [SWS_ComM_00215] [SWS_ComM_00301] [SWS_ComM_00303] [SWS_ComM_00355] [SWS_ComM_00470] [SWS_ComM_00488] [SWS_ComM_00552] [SWS_ComM_00669] [SWS_ComM_00740] [SWS_ComM_00741] [SWS_ComM_00743] [SWS_ComM_00744] [SWS_ComM_00745] [SWS_ComM_00752] [SWS_ComM_00800] [SWS_ComM_00801] [SWS_ComM_00840] [SWS_ComM_00842] [SWS_ComM_00890] [SWS_ComM_01006] [SWS_ComM_01009] [SWS_ComM_01011] [SWS_ComM_01102] [SWS_ComM_01103] [SWS_ComM_01104] [SWS_ComM_01105]
[SRS_ModeMgm_-09078]	The Communication Manager shall coordinate multiple communication requests	[SWS_ComM_00079] [SWS_ComM_00083] [SWS_ComM_00085] [SWS_ComM_00092] [SWS_ComM_00110] [SWS_ComM_00582] [SWS_ComM_00670] [SWS_ComM_00686] [SWS_ComM_00736] [SWS_ComM_00744] [SWS_ComM_00745] [SWS_ComM_00848] [SWS_ComM_01011] [SWS_ComM_01107] [SWS_ComM_91002]
[SRS_ModeMgm_-09080]	Each physical channel shall be controlled by an independent communication mode	[SWS_ComM_00051] [SWS_ComM_00092] [SWS_ComM_00744] [SWS_ComM_00745]
[SRS_ModeMgm_-09081]	The Communication Manager shall provide an API allowing collecting communication requests	[SWS_ComM_00079] [SWS_ComM_00083] [SWS_ComM_00085] [SWS_ComM_00110] [SWS_ComM_00672] [SWS_ComM_00675] [SWS_ComM_01000] [SWS_ComM_01011] [SWS_ComM_91002]

▽

△

Requirement	Description	Satisfied by
[SRS_ModeMgm_-09083]	The Communication Manager shall support two communication modes for each physical channel	[SWS_ComM_00073] [SWS_ComM_00092] [SWS_ComM_00402] [SWS_ComM_00485] [SWS_ComM_00637] [SWS_ComM_00794] [SWS_ComM_00845] [SWS_ComM_00846] [SWS_ComM_00866] [SWS_ComM_00867] [SWS_ComM_00868] [SWS_ComM_00875] [SWS_ComM_00876] [SWS_ComM_00879] [SWS_ComM_00880] [SWS_ComM_00881] [SWS_ComM_00897] [SWS_ComM_00899]
[SRS_ModeMgm_-09084]	The Communication Manager shall provide an API which allows application to query the current communication mode	[SWS_ComM_00083] [SWS_ComM_00734] [SWS_ComM_00744] [SWS_ComM_00745] [SWS_ComM_91002]
[SRS_ModeMgm_-09085]	The Communication Manager shall provide an indication of communication mode changes	[SWS_ComM_00091] [SWS_ComM_00313] [SWS_ComM_00472] [SWS_ComM_00663] [SWS_ComM_00733] [SWS_ComM_00778] [SWS_ComM_00847] [SWS_ComM_00876] [SWS_ComM_01001] [SWS_ComM_01010] [SWS_ComM_01012]
[SRS_ModeMgm_-09087]	The Minimum duration of communication request after wakeup shall be configurable	[SWS_ComM_00886] [SWS_ComM_00890] [SWS_ComM_00893] [SWS_ComM_00894]
[SRS_ModeMgm_-09089]	The Communication Manager shall be able to prevent waking up physical channels	[SWS_ComM_00156] [SWS_ComM_00157] [SWS_ComM_00302] [SWS_ComM_00470] [SWS_ComM_00742] [SWS_ComM_00747] [SWS_ComM_00799] [SWS_ComM_01008] [SWS_ComM_01106]
[SRS_ModeMgm_-09090]	Relationship between users and physical channels shall be configurable at pre compile time	[SWS_ComM_00662] [SWS_ComM_00795] [SWS_ComM_00796] [SWS_ComM_00798] [SWS_ComM_00995] [SWS_ComM_01025]
[SRS_ModeMgm_-09132]	It shall be possible to assign Network Management to physical channels	[SWS_ComM_00133] [SWS_ComM_00288] [SWS_ComM_00383] [SWS_ComM_00390] [SWS_ComM_00391] [SWS_ComM_00392] [SWS_ComM_00583] [SWS_ComM_00599] [SWS_ComM_00602] [SWS_ComM_00665] [SWS_ComM_00667] [SWS_ComM_00792] [SWS_ComM_00902] [SWS_ComM_00903]
[SRS_ModeMgm_-09133]	It shall be possible to assign physical channels to the Communication Manager	[SWS_ComM_00322] [SWS_ComM_00995]
[SRS_ModeMgm_-09141]	The Communication Manager shall be able to configure the physical channel wake-up prevention	[SWS_ComM_00066] [SWS_ComM_00157] [SWS_ComM_00218] [SWS_ComM_00219] [SWS_ComM_00302] [SWS_ComM_00625] [SWS_ComM_01106]
[SRS_ModeMgm_-09149]	The Communication Manager shall provide an API for querying the requested communication mode	[SWS_ComM_00079] [SWS_ComM_00084] [SWS_ComM_00191] [SWS_ComM_00374] [SWS_ComM_00744] [SWS_ComM_00745] [SWS_ComM_00904] [SWS_ComM_00906] [SWS_ComM_01005] [SWS_ComM_01007] [SWS_ComM_01022] [SWS_ComM_01023] [SWS_ComM_01024]
[SRS_ModeMgm_-09155]	The Communication Manager shall provide a counter for inhibited communication requests	[SWS_ComM_00138] [SWS_ComM_00140] [SWS_ComM_00141] [SWS_ComM_00142] [SWS_ComM_00470] [SWS_ComM_00625] [SWS_ComM_00803] [SWS_ComM_00839] [SWS_ComM_00840] [SWS_ComM_00962]

▽

△

Requirement	Description	Satisfied by
[SRS_ModeMgm_-09156]	It shall be provided an API to retrieve the number of inhibited "Full Communication" mode requests	[SWS_ComM_00108] [SWS_ComM_00143] [SWS_ComM_00224] [SWS_ComM_00741] [SWS_ComM_00802] [SWS_ComM_01009]
[SRS_ModeMgm_-09157]	It shall be possible to revoke a communication mode limitation, independently for each physical channel	[SWS_ComM_00085] [SWS_ComM_00124] [SWS_ComM_00156] [SWS_ComM_00163] [SWS_ComM_00182] [SWS_ComM_00470] [SWS_ComM_00552] [SWS_ComM_00669] [SWS_ComM_00741] [SWS_ComM_00743] [SWS_ComM_00744] [SWS_ComM_00745] [SWS_ComM_00840] [SWS_ComM_01006] [SWS_ComM_01008] [SWS_ComM_01009] [SWS_ComM_01011] [SWS_ComM_01102] [SWS_ComM_01103] [SWS_ComM_01104] [SWS_ComM_01105]
[SRS_ModeMgm_-09168]	The Communication Manager shall support users that are connected to no physical channel	[SWS_ComM_00664] [SWS_ComM_00744] [SWS_ComM_00745]
[SRS_ModeMgm_-09172]	It shall be possible to evaluate the current communication mode	[SWS_ComM_00092] [SWS_ComM_00176] [SWS_ComM_00191] [SWS_ComM_00744] [SWS_ComM_00745] [SWS_ComM_01001] [SWS_ComM_01010] [SWS_ComM_01012]
[SRS_ModeMgm_-09207]	ComM shall allow for additional bus specific state managers	[SWS_ComM_00957]
[SRS_ModeMgm_-09243]	The Communication Manager shall be able to handle the Partial Networks on Flexray, CAN and Ethernet	[SWS_ComM_00825] [SWS_ComM_00827] [SWS_ComM_00910] [SWS_ComM_00911] [SWS_ComM_00926] [SWS_ComM_00953] [SWS_ComM_00979] [SWS_ComM_00980] [SWS_ComM_00982] [SWS_ComM_00987]
[SRS_ModeMgm_-09245]	Enabling or disabling the Partial Network Cluster management in Com M shall be post-build selectable.	[SWS_ComM_00911]
[SRS_ModeMgm_-09246]	The communication manager shall arbitrate and coordinate requests from users on physical channel and users on PNCs	[SWS_ComM_00151] [SWS_ComM_00500] [SWS_ComM_00827] [SWS_ComM_00877] [SWS_ComM_00912] [SWS_ComM_00932] [SWS_ComM_00938] [SWS_ComM_00948] [SWS_ComM_00950] [SWS_ComM_00972] [SWS_ComM_00991] [SWS_ComM_00996] [SWS_ComM_01025] [SWS_ComM_01075] [SWS_ComM_01087] [SWS_ComM_01094]
[SRS_ModeMgm_-09247]	For each configured PNC an independent state machine shall be instantiated	[SWS_ComM_00673] [SWS_ComM_00898] [SWS_ComM_00907] [SWS_ComM_00909] [SWS_ComM_00920] [SWS_ComM_00924] [SWS_ComM_00978] [SWS_ComM_01087]
[SRS_ModeMgm_-09248]	it shall be possible to distinguish between internal and external PNC activation requests	[SWS_ComM_00694] [SWS_ComM_00940] [SWS_ComM_01014] [SWS_ComM_01015] [SWS_ComM_01060] [SWS_ComM_01061] [SWS_ComM_01062] [SWS_ComM_01065] [SWS_ComM_01068] [SWS_ComM_01072] [SWS_ComM_01085] [SWS_ComM_01087] [SWS_ComM_01088] [SWS_ComM_01089] [SWS_ComM_91028] [SWS_ComM_91029]
[SRS_ModeMgm_-09249]	PNC gateway and coordination functionality	[SWS_ComM_01061] [SWS_ComM_01066] [SWS_ComM_01068] [SWS_ComM_01072] [SWS_ComM_01075] [SWS_ComM_01083] [SWS_ComM_01089]

▽

△

Requirement	Description	Satisfied by
[SRS_ModeMgm_-09250]	PNC activation requests shall be exchanged with the Network Management via a `PNC Request Bit Vector`	[SWS_ComM_01060] [SWS_ComM_01061] [SWS_ComM_01062] [SWS_ComM_01079] [SWS_ComM_01080] [SWS_ComM_01081] [SWS_ComM_01085] [SWS_ComM_01092] [SWS_ComM_01093] [SWS_ComM_91028] [SWS_ComM_91029]
[SRS_ModeMgm_-09251]	PNC communication state shall be forwarded to the BswM	[SWS_ComM_00861] [SWS_ComM_00908] [SWS_ComM_00976]
[SRS_ModeMgm_-09256]	PNC Gateway Functionality shall consider systems with more than one gateways connected to the same network	[SWS_ComM_01073] [SWS_ComM_01074] [SWS_ComM_01076] [SWS_ComM_01077] [SWS_ComM_01078] [SWS_ComM_01079] [SWS_ComM_01080] [SWS_ComM_01081] [SWS_ComM_01084]
[SRS_ModeMgm_-09257]	ComM shall forward PNC-Clusters also to busses that are currently not awake	[SWS_ComM_01066]
[SRS_ModeMgm_-09258]	Optional Dynamic Extension of PNC Gateway	[SWS_ComM_01034] [SWS_ComM_01037] [SWS_ComM_01041] [SWS_ComM_01044] [SWS_ComM_01047] [SWS_ComM_01091]
[SRS_ModeMgm_-09259]	ComM API shall provide interfaces to access PNC Mapping (optional)	[SWS_ComM_01035] [SWS_ComM_01036] [SWS_ComM_01038] [SWS_ComM_01039] [SWS_ComM_01040] [SWS_ComM_01042] [SWS_ComM_01043] [SWS_ComM_91013] [SWS_ComM_91015] [SWS_ComM_91017] [SWS_ComM_91102] [SWS_ComM_91107]
[SRS_ModeMgm_-09260]	ComM API shall provide an interface to start PNC Learning mechanism for PNC Mapping (optional)	[SWS_ComM_01026] [SWS_ComM_01045] [SWS_ComM_01046] [SWS_ComM_01048] [SWS_ComM_01049] [SWS_ComM_01058] [SWS_ComM_91019] [SWS_ComM_91108] [SWS_ComM_91109]
[SRS_ModeMgm_-09261]	ComM shall forward the information for Partial Networking Learning (optional)	[SWS_ComM_01028] [SWS_ComM_01090] [SWS_ComM_01093] [SWS_ComM_91026]
[SRS_ModeMgm_-09262]	ComM shall set all its assigned PNCs when partial networking learning is requested (optional)	[SWS_ComM_01092]
[SRS_ModeMgm_-09263]	ComM API shall provide an interface to set PNC-membership on Host-ECU (optional)	[SWS_ComM_91021] [SWS_ComM_91108] [SWS_ComM_91109]
[SRS_ModeMgm_-09265]	ComM shall send the information for Partial Networking Learning (optional)	[SWS_ComM_01029] [SWS_ComM_91024]
[SRS_ModeMgm_-09266]	ComM shall support communication channels that act as communication slaves with wake-up capability	[SWS_ComM_01017] [SWS_ComM_01018]
[SRS_ModeMgm_-09267]	ComM shall support communication channels which act as communication slaves without wake-up capability	[SWS_ComM_00915] [SWS_ComM_01018]
[SRS_ModeMgm_-09268]	ComM shall support the possibility to forward the information if the communication request is active or passive to it's lower layer layer	[SWS_ComM_00069] [SWS_ComM_01056] [SWS_ComM_01057] [SWS_ComM_01067] [SWS_ComM_01070] [SWS_ComM_01071]
[SRS_ModeMgm_-09269]	The Communication Manager shall support synchronized PNC shutdown	[SWS_ComM_01082] [SWS_ComM_01083] [SWS_ComM_01097] [SWS_ComM_91030]

▽

△

Requirement	Description	Satisfied by
[SRS_ModeMgm_-09278]	The Communication Manager shall support synchronous and asynchronous request upon a indicated wakeup	[SWS_ComM_00990] [SWS_ComM_01063] [SWS_ComM_01064]
[SRS_ModeMgm_-09279]	The Communication Manager shall support a coordinated release of PNCs	[SWS_ComM_00947] [SWS_ComM_00952]

Table 6.1: Requirements Tracing

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]

Upstream requirements: SRS_ModeMgm_09083

⌈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.⌋

[SWS_ComM_00151]

Upstream requirements: SRS_ModeMgm_09246

⌈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.
- The Communication Mode COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST is only necessary to request the lower layer to trigger a wake-up on the network (e.g. Ethernet hardware compliant to [2, OA TC10]). This mode could not be requested by a user.

[SWS_ComM_00868]

Upstream requirements: SRS_ModeMgm_09083

⌈The Communication Mode COMM_SILENT_COMMUNICATION shall only be used for network synchronization.⌋

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]).
- The ComM module allows querying the actual Communication Mode of a channel if the user is assigned to channel(see ComM_GetCurrentComMode(), [SWS_ComM_00083])

- The ComM module allows querying for the current PNC mode if the user is assigned to a PNC (see ComM_GetCurrentPNCComMode(), [SWS_ComM_91002])

[SWS_ComM_00845]

Upstream requirements: SRS_ModeMgm_09083

⌈In COMM_FULL_COMMUNICATION mode, the ComM module shall allow transmission and reception on the affected physical channel.⌋

[SWS_ComM_00846]

Upstream requirements: SRS_ModeMgm_09083

⌈In COMM_NO_COMMUNICATION mode, the ComM module shall prevent transmission and reception on the affected physical channel.⌋

[SWS_ComM_00686]

Upstream requirements: SRS_ModeMgm_09078

⌈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.⌋

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]

Upstream requirements: SRS_ModeMgm_09246

⌈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]

Upstream requirements: SRS_ModeMgm_09083

⌈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]

Upstream requirements: SRS_ModeMgm_09078, SRS_ModeMgm_09080, SRS_ModeMgm_09083, SRS_ModeMgm_09172

⌈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]

Upstream requirements: SRS_ModeMgm_09149

⌈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 is set to 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

The ComM offers users the option to wake and keep awake so-called "partial network cluster" (PNC). A PNC is a (logical) group of ECUs which have to be active at the same time to realize some distributed functionality. If PNC-enabled gateways are used, a PNC can span the whole network (different busses on different topology levels of the network hierarchy). Without the PN functionality, NM messages can only wake and keep awake whole busses.

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 within the so-called PNC Request Bit Vector via a network management message (NM message).

Additional information regarding the partial network cluster functionality can be found in document Guide to Mode Management [10].

7.1.2 Partial Network Cluster Management Functionality

[SWS_ComM_00910]

Upstream requirements: SRS_ModeMgm_09243

⌈PNC functionality shall only exist if the parameter `ComMPncSupport` is set to TRUE. (see [ECUC_ComM_00839]).⌋

[SWS_ComM_00911] Enabling or disabling of the PNC functionality shall be post-build configurable

Upstream requirements: SRS_ModeMgm_09243, SRS_ModeMgm_09245

⌈Enabling or disabling of the PNC functionality shall be post-build configurable using the parameter `ComMPncEnabled` (see [ECUC_ComM_00901]).⌋

[SWS_ComM_CONSTR_00007] Enabling or disabling of the PNC functionality at post-build time shall be exclusively allowed for CAN and FlexRay ⌈Enabling or disabling of the PNC functionality at post-build time shall be allowed for `ComMChannel`s that have `ComMBusType` set to `COMM_BUS_TYPE_CAN` or `COMM_BUS_TYPE_FR`.⌋

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]

Upstream requirements: SRS_ModeMgm_09243

⌈For exchanging PNC status information between ComM and Nm, bit vectors shall be used. Such a bit vector is called "PNC Request Bit Vector" and contain a maximum of 504 bits.⌋

Comment: The PNC Request Bit Vector is provided as a reference to an array of type uint8 to the ComM within the dedicated APIs. Each bit in the PNC Request Bit Vector represents the status of a particular PNC. The bit is called "PNC bit".

[SWS_ComM_00825]

Upstream requirements: SRS_ModeMgm_09243

⌈The byteIndex and bitIndex, in which a PNC bit corresponding to one `ComMPncId` resides, shall be determined as follows:

- byteIndex=(`ComMPncId` div 8) - <PNC Vector Offset>

- bitIndex=(ComMPncId mod 8)

⌋

Hint: The value of the PNC Request Bit Vector length of the corresponding channel can

be obtained from the configuration of the Network Management module.

Comment: [SWS_ComM_00825] defines only the calculation of the byteIndex and bitIndex, not how it shall be implemented.

ComM receives the aggregated state of internal and external PNC requests as PNC Request Bit Vector via the callback function ComM_Nm_UpdateEIRA(<PNC Request Bit Vector of internal and external PNC requests>).

[SWS_ComM_01060]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09250

⌈If ComM_Nm_UpdateEIRA(<PNC Request Bit Vector of EIRA>) is called, then ComM shall transfer the content of the given PNC Request Bit Vector to the EIRA of ComM with respect to the PNC Request Bit Vector length configured in NmIf.⌋

Note for [SWS_ComM_01060]: It is assumed that one buffer for the EIRA PNC Request Bit Vector per PNC is implemented. The length of this buffer should have the maximum length of the configured PNC Request Bit Vectors associated with the channels which are referenced by this PNC. For example, if PNC refer to ComMChannel A and ComMChannel B and the PNC Request Bit Vector length configured for those channels deviate (ChannelA-PncRequestBitVectorLength = 10Byte and ChannelB-PncRequestBitVectorLength = 20), then one EIRA buffer for this PNC shall be available with PNC Request Bit Vector length set to 20 byte.

ComM receives the aggregated state of external PNC requests as PNC Request Bit Vector per channel via the callback function ComM_Nm_UpdateERA(<Channel>, <PNC Request Bit Vector of external PNC requests>).

[SWS_ComM_01061]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09250, SRS_ModeMgm_09249

⌈If the configuration parameter ComMPncGatewayEnabled (see [ECUC_ComM_00887]) is set to TRUE, ComM_Nm_UpdateERA(<channel>, <PNC Request Bit Vector of ERA>) is called and the parameter ComMPncGatewayType is set for the given channel, then ComM shall transfer the content of the given PNC Request Bit Vector to the ERA of ComM with respect to the given channel and the PNC Request Bit Vector length configured in NmIf.⌋

Note:

- ComM tranfers the EIRA PNC Request Bit Vector provided by Nm in one internal EIRA (see [SWS_ComM_01060]) and each ERA PNC Request Bit Vector in one ERA per ComMChannel (see [SWS_ComM_01061])
- Transferring the content of a PNC Request Bit Vector result in the internal EIRA / ERA of ComM by setting the PNC bit in the internal EIRA / ERA to '1' if the corresponding PNC bit in the PNC Request Bit Vector is set to '1' or setting the PNC bit in the internal EIRA / ERA to '0' if the corresponding PNC bit in the PNC Request Bit Vector is set to '0'

[SWS_ComM_01062]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09250

⌈The ComM module shall be able to distribute the status of a particular PNC (result of the PNC state machine) across the assigned ComM channels. Therefore ComM shall forward the aggregated state of internal PNC request per communication channel (e.g. bus or network) as PNC Request Bit Vector by calling the API Nm_UpdateIRA(<channel>, <PNC Request Bit Vector of aggregated internal PNC requests>). The IRA PNC Request Bit Vector designates the status of the internal PNC requests.⌋

Note:

- The meaning of the PNC bits is defined in [SWS_ComM_00825]
- Internal PNC requests are based on ComM user PNC requests and/or PNC requests, due to PNC gateway handling

7.1.3 ComM PNC state machine

[SWS_ComM_00953] Execute PNC functionality upfront to ComMChannel related actions

Upstream requirements: SRS_ModeMgm_09243

⌈If the PNC functionality is enabled using the configuration parameter ComMPnc-Enabled set to TRUE, all actions related to PNC changes shall be executed before the channel related actions (channel related actions, see Chapter 7.3).⌋

[SWS_ComM_00909]

Upstream requirements: SRS_ModeMgm_09247

⌈For every Partial Network Cluster, 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]

Upstream requirements: SRS_ModeMgm_09247

⌈The ComM module shall support up to 504 PNC state machines.⌋

[SWS_ComM_00924]

Upstream requirements: SRS_ModeMgm_09247

⌈ The PNC state machine shall consist of the two main states COMM_PNC_FULL_COMMUNICATION and COMM_PNC_NO_COMMUNICATION.⌋

[SWS_ComM_00907]

Upstream requirements: SRS_ModeMgm_09247

⌈The PNC main state COMM_PNC_FULL_COMMUNICATION shall consist of the sub states COMM_PNC_PREPARE_SLEEP, COMM_PNC_READY_SLEEP and COMM_PNC_REQUESTED.⌋

[SWS_ComM_00908]

Upstream requirements: SRS_ModeMgm_09251

⌈Every state change (listed within the ComM_PncModeType), excluding entering of the main state COMM_PNC_NO_COMMUNICATION coming from PowerOff, shall be notified by the API call BswM_ComM_CurrentPNCMode with the entered PNC state.⌋

[SWS_ComM_00978]

Upstream requirements: SRS_ModeMgm_09247

⌈State transitions of the PNC state machines in ComM, triggered by a call to ComM_RequestComMode() shall be executed in the ComM_MainFunction_<Channel.ShortName> only.⌋

Comment: Every PNC activation triggers sending of the PNC Request Bit Vector n-times, thus it would increase the busload without debouncing.

[SWS_ComM_00972]

Upstream requirements: SRS_ModeMgm_09246

⌈The trigger "ComMUser" represents a notification about a communication request of a ComMUser by calling the API ComM_RequestComMode().⌋

[SWS_ComM_00987]

Upstream requirements: SRS_ModeMgm_09243

⌈Within the ComM_MainFunction_<Channel.ShortName> 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_00827]

Upstream requirements: SRS_ModeMgm_09243, SRS_ModeMgm_09246

⌈ComM channel requests originating from a PNC state machine shall be treated like user requests for the according channels. This includes impact on ComM channel state machine which comes along with "Communication allowed" (see CommunicationAllowed boolean flag) and "Communication inhibition")⌋

Note: Please refer to (see Chapter 7.4.1 for details regarding communication inhibition

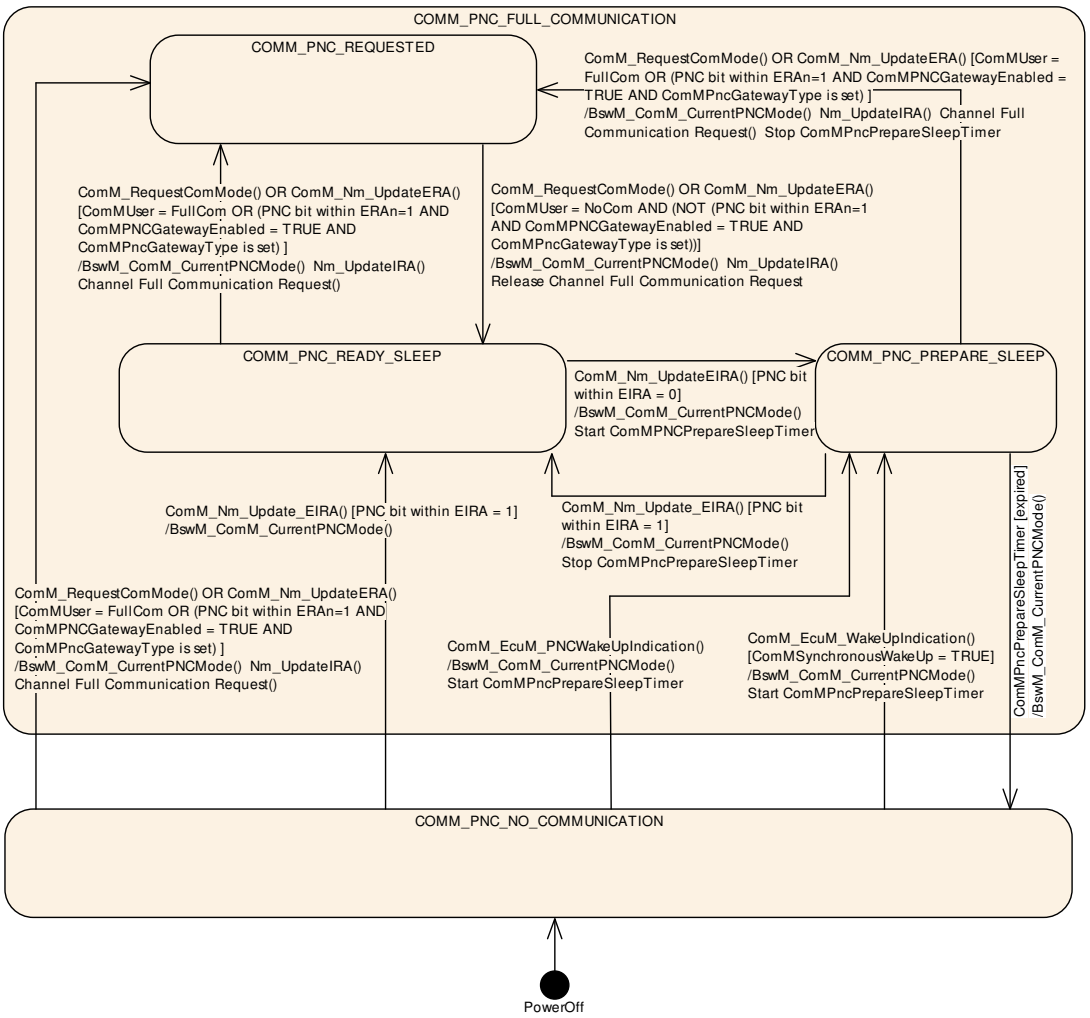


Figure 7.1: PNC State Machine

7.1.3.1 Behavior in PNC main state COMM_PNC_NO_COMMUNICATION

[SWS_ComM_00926]

Upstream requirements: SRS_ModeMgm_09243

⌈The PNC main state COMM_PNC_NO_COMMUNICATION shall be the default PNC state from power off.⌋

The main state COMM_PNC_NO_COMMUNICATION is the target state as long as the PNC is neither requested ECU internally nor requested externally.

[SWS_ComM_01063]

Upstream requirements: SRS_ModeMgm_09278

⌈If the API ComM_EcuM_WakeUpIndication() is called in PNC state COMM_PNC_NO_COMMUNICATION, the configuration switch ComMSynchronousWakeUp is set to TRUE (see [ECUC_ComM_00695]) and for all PNCs that reference at least one ComMChannel via ComMChannelPerPnc (see [ECUC_ComM_00880]), the PNC main state COMM_PNC_NO_COMMUNICATION shall be left and the PNC sub state COMM_PNC_PREPARE_SLEEP shall be entered.⌋

[SWS_ComM_00990]

Upstream requirements: SRS_ModeMgm_09278

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

Comment: In case of asynchronous wake up, the PNC state shall stay in COMM_PNC_NO_COMMUNICATION until the PNC request is received (PNC bit in EIRA/ERAn is set to '1').

[SWS_ComM_01064]

Upstream requirements: SRS_ModeMgm_09278

⌈If the API ComM_EcuM_PNCWakeUpIndication(<PNC>) (see [SWS_ComM_91001]) is called in PNC state COMM_PNC_NO_COMMUNICATION and the indicated PNC reference at least one ComMChannel via ComMChannelPerPnc (see [ECUC_ComM_00880]), the PNC main state COMM_PNC_NO_COMMUNICATION shall be left and the PNC sub state COMM_PNC_PREPARE_SLEEP shall be entered.⌋

[SWS_ComM_00932]

Upstream requirements: SRS_ModeMgm_09246

⌈When at least one ComMUser assigned to this PNC requests "Full Communication" in PNC main state COMM_PNC_NO_COMMUNICATION, this state shall be left and the sub state COMM_PNC_REQUESTED of the main state COMM_PNC_FULL_COMMUNICATION shall be entered.⌋

[SWS_ComM_01065]

Upstream requirements: SRS_ModeMgm_09248

⌈When in main state COMM_PNC_NO_COMMUNICATION at least one PNC bit representing this PNC in EIRA changes to '1' and this PNC reference at least one ComMChannel via ComMChannelPerPnc (see [ECUC_ComM_00880]), the main state COMM_PNC_NO_COMMUNICATION shall be left and the COMM_PNC_READY_SLEEP shall be entered.⌋

7.1.3.1.1 PNC gateway related requirements

[SWS_ComM_01066]

Upstream requirements: SRS_ModeMgm_09257, SRS_ModeMgm_09249

⌈When in main state COMM_PNC_NO_COMMUNICATION at least one PNC bit representing this PNC in ERAn changes to '1', then the main state COMM_PNC_NO_COMMUNICATION shall be left and the sub state COMM_PNC_REQUESTED shall be entered under the following conditions:

- the parameter ComMPncGatewayEnabled (see [ECUC_ComM_00887]) is set to TRUE
- this PNC references at least one channel via ComMChannelPerPnc (see [ECUC_ComM_00880]) and all referenced channels have the ComMPncGatewayType set

⌋

Note: All the channels shall have GW type set which are referred by the PNC irrespective of the type of the reference i.e ComMChannelPerPnc or ComMChannelPerTxOnlyPnc.

7.1.3.2 On entry of PNC main state COMM_PNC_NO_COMMUNICATION from PowerOff

Note: After switching on the power supply, main state COMM_PNC_NO_COMMUNICATION is entered from PowerOff (see [SWS_ComM_00926])

7.1.3.3 Behavior in PNC main state COMM_PNC_FULL_COMMUNICATION

[SWS_ComM_00929]

Upstream requirements: SRS_ModeMgm_00049

⌈As long as a specific PNC is in state COMM_PNC_FULL_COMMUNICATION all ComMChannels which are referenced by this PNC via ComMChannelPerPnc (see [ECUC_ComM_00880]) shall be in COMM_FULL_COMMUNICATION.⌋

7.1.3.4 On entry of PNC sub state COMM_PNC_REQUESTED

[SWS_ComM_01067]

Upstream requirements: SRS_ModeMgm_09268

⌈When entering the PNC sub state COMM_PNC_REQUESTED from COMM_PNC_NO_COM or COMM_PNC_PREPARE_SLEEP, this PNC reference at least one ComMChannel via ComMChannelPerPnc (see [ECUC_ComM_00880]) and ComMPncWakeupSleepRequestEnabled of this PNC is set to TRUE, BswM_ComM_CurrentPNCMode shall be called with COMM_PNC_REQUESTED_WITH_WAKEUP_REQUEST, instead of calling BswM_ComM_CurrentPNCMode with COMM_PNC_REQUESTED.⌋

Note: Notification towards the BswM with COMM_PNC_REQUESTED_WITH_WAKEUP_REQUEST is used for Ethernet switch port switching to trigger a wake-up on the network where the used Ethernet hardware is compatible to the OA TC10 (see [2])

[SWS_ComM_01068]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09249

⌈When entering the PNC sub state COMM_PNC_REQUESTED, then the ComM module shall set the PNC bit with value '1' of the PNC bit representing this PNC within the IRA and forward the aggregated internal PNC requests to each channel which is referenced this PNC by calling Nm_UpdateIRA(<channel>, <IRA>) under either of the following conditions:

- ComMPncGatewayEnabled is set to FALSE
- ComMPncGatewayType is not set on any of the ComMChannels referenced by this PNC

⌋

[SWS_ComM_01069]

Upstream requirements: SRS_ModeMgm_00049

⌈Every time the sub state COMM_PNC_REQUESTED is entered from other states, ComM shall request COMM_FULL_COMMUNICATION for all configured ComM channels which

are referenced by this PNC via parameter ComMChannelPerPnc (see [ECUC_ComM_00880]) and where ComMWakeupSleepRequestEnabled is set to FALSE or not available, even if the channel is already requested.⌋

[SWS_ComM_01070]

Upstream requirements: SRS_ModeMgm_09268

⌈Every time the sub state COMM_PNC_REQUESTED is entered from COMM_PNC_NO_COM or COMM_PNC_PREPARE_SLEEP, ComM shall request COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST for all configured ComM channels which are referenced by this PNC via parameter ComMChannelPerPnc (see [ECUC_ComM_00880]) and where ComMWakeupSleepRequestEnabled is set to TRUE, even if the channel is already requested.⌋

[SWS_ComM_01071]

Upstream requirements: SRS_ModeMgm_09268, SRS_ModeMgm_00049

⌈Every time the sub state COMM_PNC_REQUESTED is entered from COMM_PNC_READY_SLEEP, ComM shall request COMM_FULL_COMMUNICATION for all configured ComM channels which are referenced by this PNC via parameter ComMChannelPerPnc (see [ECUC_ComM_00880]) and where ComMWakeupSleepRequestEnabled is set to TRUE, even if the channel is already requested.⌋

Comment on [SWS_ComM_01071]: Entering from COMM_PNC_READY_SLEEP should not result in a wake-up on the network, since the PNC is already requested remotely by another ECU

7.1.3.4.1 PNC gateway related requirements

[SWS_ComM_01072]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09249

⌈When entering the PNC sub state COMM_PNC_REQUESTED and ComMPncGatewayEnabled is set to TRUE, then ComM shall set the PNC bit with value '1' of the PNC bit representing this PNC within the IRA on all referenced ComMChannels where ComMPncGatewayType is set to COMM_GATEWAY_TYPE_ACTIVE and forward the aggregated internal PNC request accordingly to those ComMChannels by calling Nm_UpdateIRA(<channel>, <IRA>)⌋

7.1.3.5 Behavior in PNC sub state COMM_PNC_REQUESTED

[SWS_ComM_00938]

Upstream requirements: SRS_ModeMgm_09246

⌈When all ComMUsers assigned to this PNC request "No Communication", the sub state COMM_PNC_REQUESTED shall be left and the sub state COMM_PNC_READY_SLEEP shall be entered, if ComMPncGatewayEnabled is set to FALSE or ComMPncGatewayType is not set on all channels which are referenced by this PNC.⌋

Note: As long as at least one ComMUser assigned to this PNC requests "Full Communication", COMM_PNC_REQUESTED will be the current PNC state. Please refer to the following requirements.

[SWS_ComM_01073]

Upstream requirements: SRS_ModeMgm_09256

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to FALSE and at least one ComMUser assigned to a specific PNC requests "Full Communication", then ComM shall request COMM_FULL_COMMUNICATION of those ComMChannels which are referenced via ComMChannelPerTxOnlyPnc by this PNC.⌋

[SWS_ComM_01074]

Upstream requirements: SRS_ModeMgm_09256

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to FALSE and all ComMUsers assigned to a specific PNC requests "No Communication", then ComM shall request COMM_NO_COMMUNICATION of those ComMChannels which are referenced via ComMChannelPerTxOnlyPnc by this PNC.⌋

7.1.3.5.1 PNC gateway related requirements

[SWS_ComM_00991]

Upstream requirements: SRS_ModeMgm_09246

⌈The sub state COMM_PNC_REQUESTED shall be left and the sub state COMM_PNC_READY_SLEEP shall be entered under the following conditions:

- all ComMUsers assigned to this PNC request "No Communication"
- the parameter ComMPncGatewayEnabled is set to TRUE
- at least one ComMChannel is referenced via ComMChannelPerPnc (see [ECUC_ComM_00880]) by this PNC
- all ComMChannels referenced by this PNC have ComMPncGatewayType parameter set
- the PNC bit representing this PNC equals to ’0’ in ERAn

⌋

[SWS_ComM_01075]

Upstream requirements: SRS_ModeMgm_09246, SRS_ModeMgm_09249

⌈The sub state COMM_PNC_REQUESTED shall be left and the sub state COMM_PNC_READY_SLEEP shall be entered under the following conditions:

- all ComMUsers assigned to this PNC request "No Communication"
- the parameter ComMPncGatewayEnabled is set to TRUE
- all ComMChannels referenced by this PNC have ComMPncGatewayType parameter NOT set

⌋

[SWS_ComM_01076]

Upstream requirements: SRS_ModeMgm_09256

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to TRUE and at least one ComMUser assigned to a specific PNC requests "Full Communication", then ComM shall set the PNC bit representing this specific PNC to value '1' within the IRA of those ComMChannels

- which have ComMPncGatewayType parameter set to COMM_GATEWAY_TYPE_PASSIVE and
- referenced either via ComMChannelPerPnc or via ComMChannelPerTxOnlyPnc by this PNC,

and forward the updated IRA with a call of Nm_UpdateIRA(<channel>, <IRA>).⌋

[SWS_ComM_01077]

Upstream requirements: SRS_ModeMgm_09256

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to TRUE and the PNC bit representing a specific PNC equals to '1' in at least one ERA, whose corresponding ComMChannel has the ComMPncGatewayType parameter set to COMM_GATEWAY_TYPE_ACTIVE, then ComM shall set the PNC bit representing this specific PNC to value '1' within the IRA of those ComMChannels

- which have ComMPncGatewayType parameter set to COMM_GATEWAY_TYPE_PASSIVE and
- referenced via ComMChannelPerPnc or via ComMChannelPerTxOnlyPnc by this PNC,

and forward the updated IRA with a call of Nm_UpdateIRA(<channel>, <IRA>).⌋

[SWS_ComM_01078]

Upstream requirements: SRS_ModeMgm_09256

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to TRUE and at least one ComMUser assigned to a specific PNC requests "Full Communication", then ComM shall request COMM_FULL_COMMUNICATION of those ComMChannels which are referenced via ComMChannelPerTxOnlyPnc by this PNC.⌋

[SWS_ComM_01079]

Upstream requirements: SRS_ModeMgm_09256, SRS_ModeMgm_09250

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to TRUE, if

- all ComMUsers assigned to a specific PNC request "No Communication" and
- the PNC bit representing this specific PNC equals to '0' in ERAn, whose corresponding ComMChannel has the ComMPncGatewayType parameter set to COMM_GATEWAY_TYPE_ACTIVE,

then ComM shall set the PNC bit representing this specific PNC to value '0' within

the IRA of those ComMChannels

- which have ComMPncGatewayType parameter set to COMM_GATEWAY_TYPE_PASSIVE and
- which are referenced via ComMChannelPerPnc or via ComMChannelPerTxOnlyPnc by this PNC,

and forward the updated IRA with a call of Nm_UpdateIRA(<channel>, <IRA>).⌋

[SWS_ComM_01080]

Upstream requirements: SRS_ModeMgm_09256, SRS_ModeMgm_09250

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to TRUE, if

- all ComMUsers assigned to a specific PNC request "No Communication" and
- the ComMChannels which are referenced by this PNC have the ComMPncGatewayType parameter not set,

then ComM shall set the PNC bit representing this specific PNC to value '0' within

the IRA of all ComMChannels which are referenced by this PNC and forward the updated IRA with a call of Nm_UpdateIRA(<channel>, <IRA>)⌋

[SWS_ComM_01081]

Upstream requirements: SRS_ModeMgm_09256, SRS_ModeMgm_09250

⌈In sub state COMM_PNC_REQUESTED when ComMPncGatewayEnabled is set to TRUE and all ComMUsers assigned to a specific PNC request "No Communication",

then ComM shall request COMM_NO_COMMUNICATION of those ComMChannels which are referenced via ComMChannelPerTxOnlyPnc by this PNC.⌋

[SWS_ComM_01082]

Upstream requirements: SRS_ModeMgm_09269

⌈When a request to forward a synchronized PNC shutdown has been indicated via a call of ComM_Nm_ForwardSynchronizedPncShutdown(<channel>, <PNC Request Bit Vector>) in sub-state COMM_PNC_REQUESTED and all following conditions apply:

- all ComM users assigned to this PNC request "No Com",
- all corresponding PNC bits are set to '0' in ERAn of all channels which are referenced by this PNC via ComMChannelPerPnc (see [ECUC_ComM_00880]) where the channel attribute ComMPncGatewayType is set to COMM_GATEWAY_TYPE_ACTIVE,
- the PNC is indicated for a shutdown (PNC bit set to '1' in the given PNC Request Bit Vector) within the call of ComM_Nm_ForwardSynchronizedPncShutdown
- the indicated channel has ComMPncGatewayType set to COMM_GATEWAY_TYPE_PASSIVE and the channel is referenced via ComMChannelPerPnc (see [ECUC_ComM_00880]),
- ComMSynchronizedPncShutdownEnabled is set to TRUE,

then the ComM module shall perform the following actions:

- ComM shall set the ERA bit to '0' of this PNC in the ERA of all channels which are referenced by this PNC via ComMChannelPerPnc (see [ECUC_ComM_00880]) where the channel attribute ComMPncGatewayType is set to COMM_GATEWAY_TYPE_PASSIVE
- ComM shall call Nm_RequestSynchronizedPncShutdown (<channel>, <PncId>) for each <channel> with <PncId> of the current handled PNC, where ComMPnc-GatewayType is set to "COMM_GATEWAY_TYPE_ACTIVE" and the channel is referenced via ComMChannelPerPnc (see [ECUC_ComM_00880])
- The sub state COMM_PNC_REQUESTED shall be left and the sub state COMM_PNC_READY_SLEEP shall be entered

⌋

Comment on [[SWS_ComM_01082]]:

- Every time an intermediate PNC coordinator (PNC coordinator which have at least one ComMChannel with ComMPncGatewayType set to COMM_GATEWAY_TYPE_PASSIVE) receive a Nm frame as PN shutdown message from the top-level PNC coordinator, ComM shall immediately release the PNC, forward the PNC Request Bit Vector of the PN shutdown message and request a synchronized PNC shutdown (request to transmit a PN shutdown message) on those

ComMChannels which are are assigned to the affected PNC and where ComMPncGatewayType is set to COMM_GATEWAY_TYPE_ACTIVE

- ComM has to ensure that the procedure upon the reception of Nm frame as PN shutdown message has to be performed as fast as possible, to minimize the delay of the synchronized PNC shutdown
- The forwarding of a synchronized PNC shutdown is not performed if a local user has indicated to request the affected PNC, or a PNC request was received via a ComM channel with ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE. The request for a PNC either local requested or remotely requested always overrule a request for a synchronized PNC shutdown.
- Synchronized PNC shutdown handling is only performed if the indicated PNCs (given within the PNC Request Bit Vector) reside in COMM_PNC_REQUESTED

[SWS_ComM_01097]

Upstream requirements: SRS_ModeMgm_09269

⌈If a request to forward a synchronized PNC shutdown has been indicated via a call of ComM_Nm_ForwardSynchronizedPncShutdown(<channel>) for this PNC, the PNC is qualified to be released and the precondition to forward the synchronized PNC request are not fulfilled (see [SWS_ComM_01082]), then the ComM module shall reject to perform the forwarding of a synchronized PNC shutdown and if ComMPncNmRequest is set to TRUE, then ComM shall request the network again by invoking Nm_NetworkRequest for all ComMChannels which are assigned to this PNC, even though the current state of an affected channel is already "Full communication"⌋

[SWS_ComM_01083]

Upstream requirements: SRS_ModeMgm_09269, SRS_ModeMgm_09249

⌈If ComMSynchronizedPncShutdownEnabled is set to TRUE and ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE on all ComM channels assigned to this PNC, the API Nm_RequestSynchronizedPncShutdown (<channel>, <PncId>) shall be called, whereat <channel> represent the current handled ComMChannel and <PncId> the ComMPncId of this PNC under the following conditions:

- corresponding PNC bit in ERAn is equal to "0"
- all ComMUsers assigned to this PNC request "No Communication"
- The channel is referenced via ComMChannelPerPnc (see [ECUC_ComM_00880]) by this PNC

⌋

Comment on [SWS_ComM_01083]: Everytime a PNC is released, synchronized PNC shutdown is configured and the ECU act as a top-level PNC coordinator for this PNC, a PN shutdown message has to be transmitted on the affected ComMChannels. Therefore ComM forward the PNC Request Bit Vector regarding the detection of a released PNC to NmIf by calling Nm_RequestSynchronizedPncShutdown

for each ComMChannel the PNC is assigned to. NmIf is forwarding the call to the affected <Bus>Nm. The PN shutdown message is transmitted within the <Bus>Nm_Mainfunction.

[SWS_ComM_01084]

Upstream requirements: SRS_ModeMgm_09256

⌈In sub state COMM_PNC_REQUESTED if ComM0PncVectorAvoidance is set to TRUE and all PNC bits in the calculated IRA of a ComMChannel referenced via ComMChannelPerPnc (see [ECUC_ComM_00880]) are set to '0', the ComM module shall release this ComMChannel. As soon as at least one bit in the IRA changes back to '1' again, the ComM module shall request this ComMChannel again.⌋

Comment on [SWS_ComM_01084]: As long as a PNC is requested remotely (i.e. at least one PNC bit within ERAn assigned to this PNC equals '1') and the configuration switch ComMPncGatewayEnabled is set to TRUE, COMM_PNC_REQUESTED will be the current PNC state.

7.1.3.6 On entry PNC sub state COMM_PNC_READY_SLEEP

[SWS_ComM_01085]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09250

⌈When entering the PNC sub state COMM_PNC_READY_SLEEP from COMM_PNC_REQUESTED, then the PNC bit representing this PNC within the IRA shall be set to value '0' and the aggregated internal PNC requests shall be forwarded to each channel which is referenced by this PNC by calling Nm_UpdateIRA(<channel>, <IRA>)⌋

[SWS_ComM_01086]

Upstream requirements: SRS_ModeMgm_00049

⌈When entering the PNC sub state COMM_PNC_READY_SLEEP from COMM_PNC_REQUESTED, ComM shall release the COMM_FULL_COMMUNICATION request for all configured ComM channels referenced via ComMChannelPerPnc (see [ECUC_ComM_00880]) by this PNC⌋

7.1.3.7 Behavior in PNC sub state COMM_PNC_READY_SLEEP

As long as the PNC is requested (i.e. the PNC bit representing this PNC within EIRA equals '1') and no ComMUser assigned to this PNC requests "Full Communication", COMM_PNC_READY_SLEEP will be the current state.

[SWS_ComM_00940]

Upstream requirements: SRS_ModeMgm_09248

⌈If the PNC is released (i.e. the PNC bit representing this PNC within EIRA equals '0'), the sub state COMM_PNC_READY_SLEEP shall be left and the sub state COMM_PNC_PREPARE_SLEEP shall be entered.⌋

[SWS_ComM_01087]

Upstream requirements: SRS_ModeMgm_09246, SRS_ModeMgm_09247, SRS_ModeMgm_09248

⌈The sub state COMM_PNC_READY_SLEEP shall be left and the sub state COMM_PNC_REQUESTED shall be entered if at least one ComMUser assigned to this PNC requests "Full Communication".⌋

7.1.3.7.1 PNC gateway related requirement

[SWS_ComM_01088]

Upstream requirements: SRS_ModeMgm_09248

⌈When in sub state COMM_PNC_READY_SLEEP at least one PNC bit representing this PNC in ERAn changes to '1', the sub state COMM_PNC_READY_SLEEP shall be left and the sub state COMM_PNC_REQUESTED shall be entered under the following conditions:

- the parameter ComMPncGatewayEnabled (see [ECUC_ComM_00887]) is set to TRUE,
- this PNC references at least one channel via ComMChannelPerPnc (see [ECUC_ComM_00880]) and the referenced channels have the ComMPnc-GatewayType set

⌋

7.1.3.8 On entry of PNC sub state COMM_PNC_PREPARE_SLEEP

[SWS_ComM_00952]

Upstream requirements: SRS_ModeMgm_09279

⌈If the sub state COMM_PNC_PREPARE_SLEEP is entered, the timer ComMPnc-PrepareSleepTimer (see [ECUC_ComM_00841]) shall be started with the configured initial value.⌋

7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP

As long as the timer ComMPncPrepareSleepTimer (see [ECUC_ComM_00841]) is running and no changes in ComMUser, EIRA or ERAn occur, COMM_PNC_PREPARE_SLEEP will be the current state.

[SWS_ComM_00947]

Upstream requirements: SRS_ModeMgm_09279

⌈When the timer ComMPncPrepareSleepTimer (see [ECUC_ComM_00841]) expires, the PNC sub state COMM_PNC_PREPARE_SLEEP shall be left and the PNC main state COMM_PNC_NO_COMMUNICATION shall be entered.⌋

[SWS_ComM_00948]

Upstream requirements: SRS_ModeMgm_09246

⌈When in COMM_PNC_PREPARE_SLEEP at least one ComMUser assigned to this PNC requests "Full Communication", the COMM_PNC_PREPARE_SLEEP state shall be left. The timer ComMPncPrepareSleepTimer shall be stopped and the sub state COMM_PNC_REQUESTED state shall be entered.⌋

[SWS_ComM_00950]

Upstream requirements: SRS_ModeMgm_09246

⌈When in COMM_PNC_PREPARE_SLEEP the PNC bit representing this PNC within EIRA changes to '1' and this PNC references at least one channel via ComMChannelPerPnc (see [ECUC_ComM_00880]), the sub state COMM_PNC_PREPARE_SLEEP shall be left. The timer ComMPncPrepareSleepTimer shall be stopped and the sub state COMM_PNC_READY_SLEEP shall be entered.⌋

7.1.3.9.1 PNC gateway related requirements

[SWS_ComM_01089]

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09249

⌈When in sub state COMM_PNC_PREPARE_SLEEP at least one PNC bit representing this PNC in ERAn changes to '1', then sub state COMM_PNC_PREPARE_SLEEP shall be left, COMM_PNC_REQUESTED shall be entered and timer ComMPncPrepareSleepTimer shall be stopped under the following conditions:

- the parameter ComMPncGatewayEnabled (see [ECUC_ComM_00887]) is set to TRUE,
- this PNC references at least one channel via ComMChannelPerPnc (see [ECUC_ComM_00880]) and the referenced channels have the ComMPncGatewayType set

⌋

7.1.4 PNC Gateway

The PNC Gateway feature is used to span (logical) partial network clusters across bus / communication channel boundaries, "gatewaying" PNC requests from one bus/network to the others. (Therefore, for a PNC gateway to exist, it needs to be connected to multiple physical channels.)

To do so, the PNC gateway configuration contains information for each PNC which physical channels are required to reach all members of that PNC (PNC-to-channel-mapping, see Figure 7.2).

The PNC gateway collects PNC requests from all of its multiple active channels (which are called active since it actively keeps them awake, if required) and aggregates them. The PNC gateway sends the aggregated PNC state in the network to all its active channels, which causes all nodes to have the same view on the global PNC request state as the gateway.

If the PNC gateway is not the topmost PNC gateway in the network hierarchy, the PNC gateway will also send the aggregated PNC request state of all subordinate nodes, plus its own internal request state, to its superior PNC coordinator, which is connected via the so-called "passive" channel (which is called passive because it's the opposite of active).

The superior PNC coordinators will aggregate the subordinate coordinators' PNC request states, so the top level coordinator will know about all active PNC requests in the network, and send that info to the subordinate nodes.

Subordinate PNC coordinators forward the PNC request information received on their passive channel to their active channels to distribute the top level coordinators holistic view of the PNC request state to all leaf nodes in the logical hierarchy, so every node in the system is on the same page regarding the PNC request state.

A PNC coordinator must never aggregate and send back the information received via its passive channel in order not to create an endless mirroring loop of "phantom PNC requests".

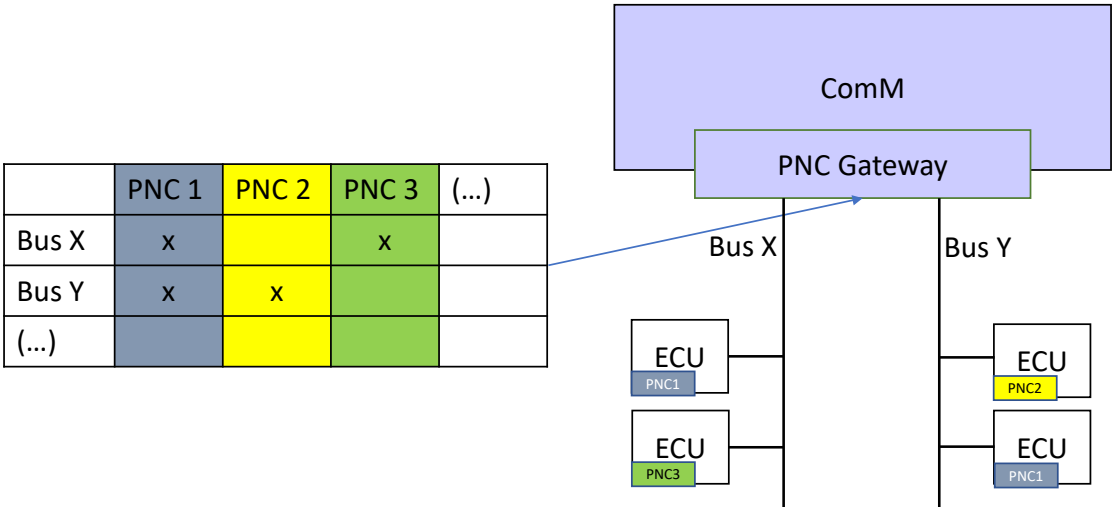


Figure 7.2: PNC-to-channel-mapping

The PNC to channel mapping is provided statically by configuration. Additionally, the optional feature Dynamic PNC-to-channel-mapping (see chapter 7.5) could be used to extend the PNC-to-channel mapping during run-time.

Note that when PNC Gateway is active and even if a PNC is only assigned to one channel, coordination might occur when request comes in from another channel where PNC is not assigned to. This is intended as there might be only PNC-requestor on the other channel which is not interested in being kept awake by this PNC.

7.1.4.1 Support for not coordinated PNCs assigned to multiple channels

Comment: When a Partial Network is assigned to more than one `ComMChannel` than this PNC is coordinated either on all affected `ComMChannel`s or not at all (see [11, System Template] [constr_5094]).

Note: If PNCs are assigned to different `ComMChannel`s and those `ComMChannel`s are not coordinated by a PNC gateway, then the network topology and communication design has to ensure, that the affected `ComMChannel`s are requested and released to the same point in time. If PNCs are used, an application should not care about `ComMChannel` states, and additionally, ComM will not take care about `ComMChannel` states for this use case, since the PNC coordination for those `ComMChannel`s is not performed. Or in other words, if a PNC is requested (passively) then also all referenced `ComMChannel`s shall be requested (passively), because an application expects that all `ComMChannel`s assigned to this PNCs reside in `COMM_FULL_COMMUNICATION`.

Figure 7.3 depict an example for a PNC gateway (Node2) with not coordinated `ComMChannel`s

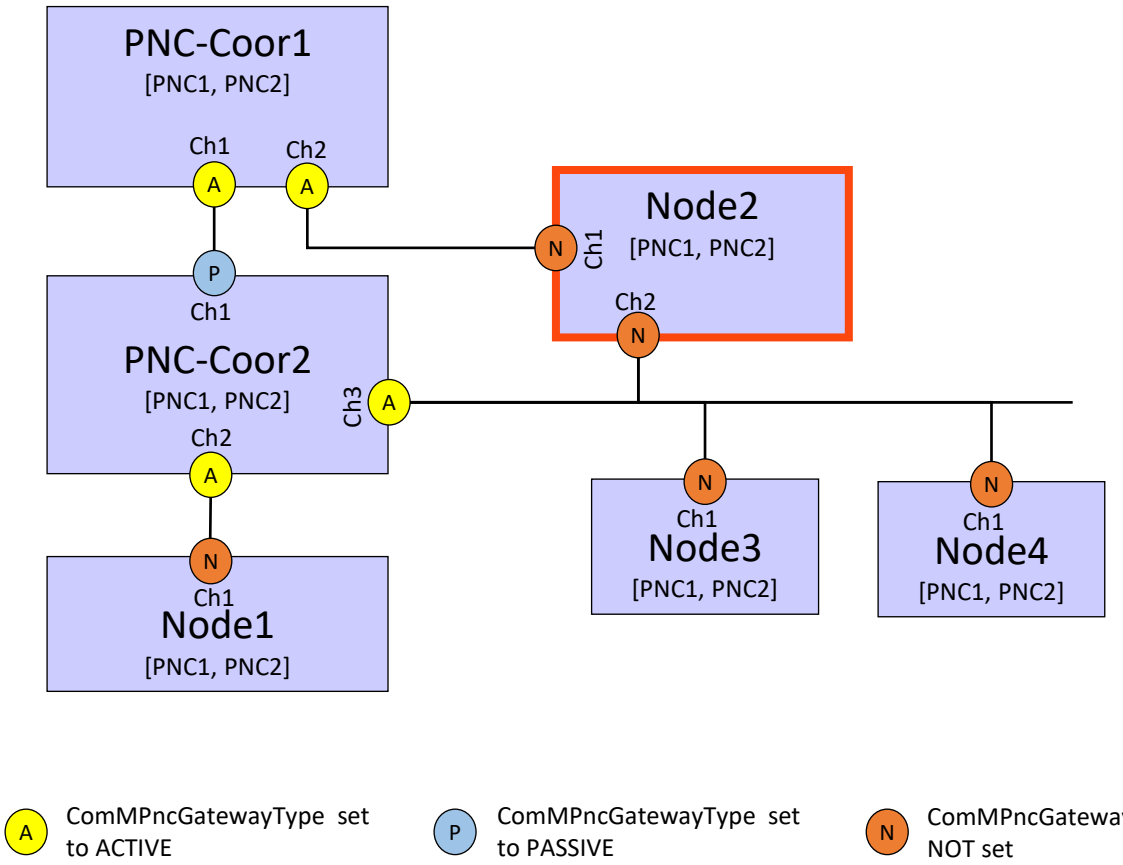


Figure 7.3: Example for a PNC gateway with not coordinated `ComMChannel`s (see Node2)

7.1.4.2 Active PNC Gateway

Note: Even if the configuration parameter `ComMPncGatewayEnabled` (see [ECUC_ComM_00887]) is TRUE and the parameter `ComMPncGatewayType` is set to `COMM_GATEWAY_TYPE_ACTIVE` for a `ComMChannel` (see [ECUC_ComM_00842]), the active PNC gateway still behaves as shown in Figure 7.1 “PNC State Machine”.

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

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

7.1.4.3 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` is set to TRUE) ComM channels mapped to this PNC 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.

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.

Comment to [SWS_ComM_01079] and [SWS_ComM_01080]: A PNC gateway calculates the PNCs bit value in the ERA Tx PNC Request Bit Vectors to be sent for a passively coordinated channel, in the same manner as the PNC bit value in ERA for an actively coordinated channel, but sets the PNC's bit to '0' according to the rules of [SWS_ComM_01079] and [SWS_ComM_01080].

7.1.4.4 Synchronized PNC shutdown

A PN topology always reflects a hierarchical topology, where the so-called top-level PNC coordinator is located on the highest level. On the subordinated levels multiple so-called intermediate PNC coordinators and PNC leaf nodes could reside.

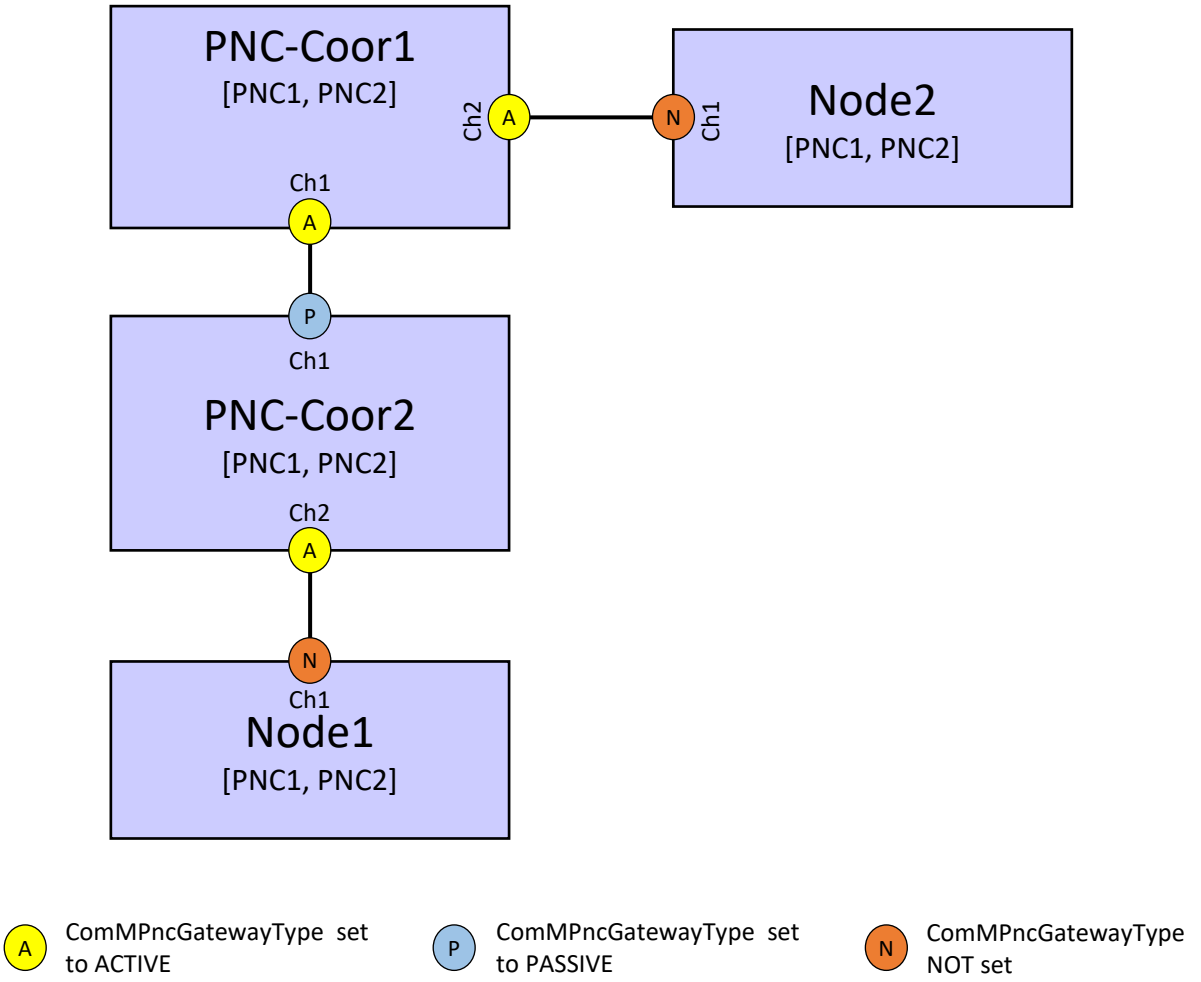


Figure 7.4: Example for a partial network (PN) topology that reflect the hierarchy

Figure 5 shows PNC-Coor1 as top-level PNC coordinator, PNC-Coor2 as intermediate PNC coordinator, Node1 and Node2 as PNC leaf node which resides on the lowest level of the PN topology. For example, if Node1 requests PNC1, then the PNC request is propagated across the PN to the top-level PNC coordinator. The top-level PNC coordinator "takes over" the PNC request and ensures that the PNC request is distributed across the PN. Therefore the top-level PNC coordinator mirrors back the PNC request on channel 1 (PNC-Coor1.Ch1) and forward the PNC request to channel 2 (PNC-Coor2.Ch2). If for example Node1 releases PNC1 and no other ECU in the network has PNC1 requested, then Node1 will still receive Nm frames from the top-level PNC coordinator where the PNC1 is requested. The release of the PNC leaf node is not forwarded immediately across the PN topology from the PNC leaf node to the top-level PNC coordinator. The release of a PNC is delayed by the PN reset time on each PN topology level. If the top-level PNC coordinator detects that a PN reset timer for a particular PNC expires, then no other ECU in the PN request this PNC. The top-level PNC coordinator resets the PN reset timer of the released PNC once more and transmits a so-called PN shutdown message to ensure a nearly synchronized shutdown of the PNC, across all PN levels from the top-level PNC coordinator down to the PNC leaf nodes. An intermediate PNC coordinator reacts immediately upon reception

on a PN shutdown message. Therefore the intermediate PNC coordinator releases the indicated PNC, resets the PN reset timer once more and forwards the PN shutdown message on all ComMChannels which are actively coordinated and assigned to the affected PNC. Thus, all PNC state machines of the released PNC across all PN level from the top-level PNC coordinator down to the PNC leaf nodes reside in COMM_PNC_READY_SLEEP and reset the corresponding PN reset timer nearly at the same point in time. This will lead to a synchronized PNC shutdown to avoid timeouts on application level.

Please refer also to the sequence diagrams Figure 9.5, Figure 9.6 and Figure 9.7 which depict the handling of a synchronized PNC shutdown in the role of a top-level PNC coordinator and an intermediate PNC coordinator.

Note:

- For ComMChannels which are configured for a uni-directional PNC handling (see 7.1.6.2), no synchronized PNC shutdown is performed.
- For PNCs which reference a ComMChannel via the parameter ComMChannel-PerTxOnlyPnc (see 7.1.6.3), no synchronized PNC shutdown is performed.

7.1.4.5 Support for multiple top-level PNC coordinators

According to chapter 7.1.4.4 a PN topology always have at least one top-level PNC coordinator. The top-level PNC coordinator for a particular PNC is designated if all ComMChannels have ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE where this particular PNC is assigned to (see [SWS_ComM_01083]). For different PNCs it is possible to have different top-level PNC coordinators. For the same PNC usually one top-level PNC coordinator is used (see Figure 7.5). But it is also possible to have multiple top-level PNC coordinators for the same PNC, if the PN topology fulfill the design constraint of SystemTemplate constr_pnc90. Figure 7.6 shows an example of valid PN with multiple top-level PNC coordinators of the same PNC.

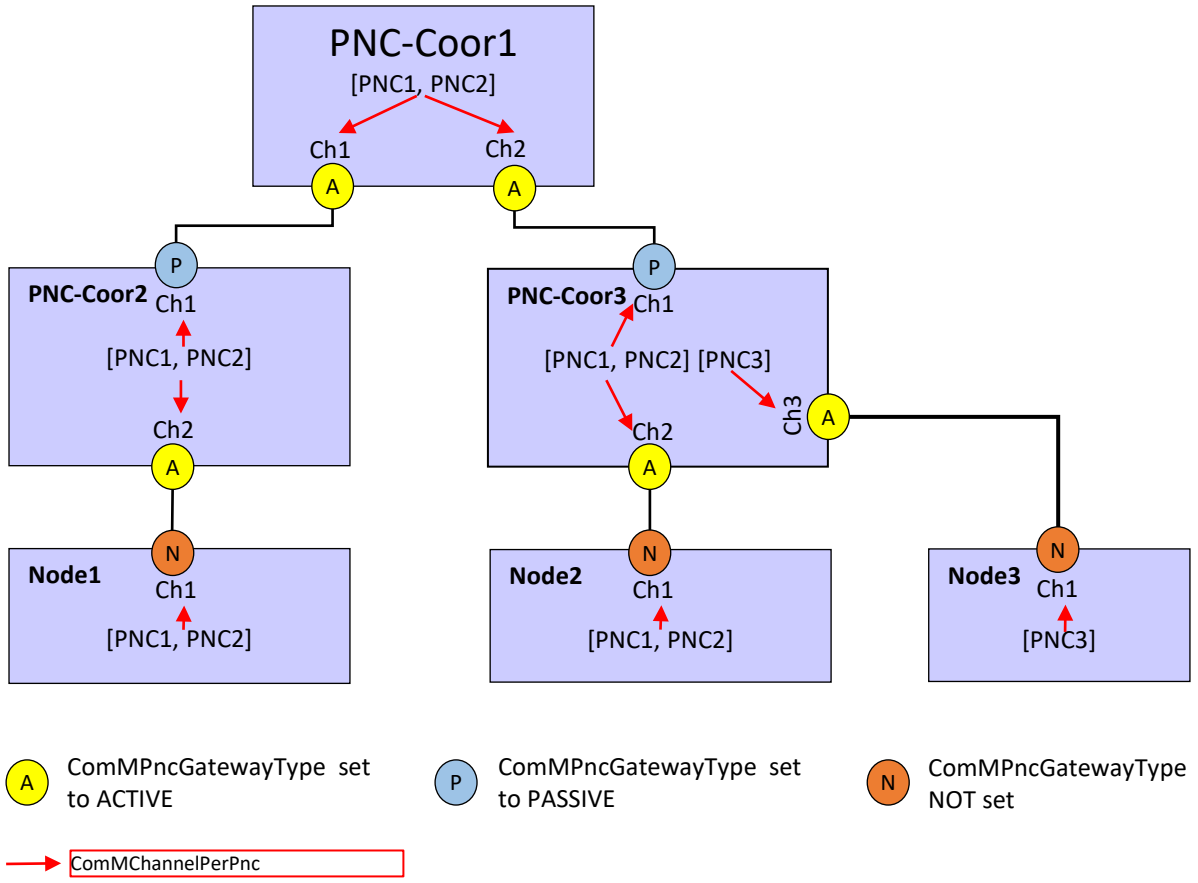


Figure 7.5: One top-level PNC coordinator per PNC

In Figure 7.5 PNC-Coor1 act as top-level PNC coordinator for PNC1 and PNC2. PNC-Coor3 act as top-level PNC coordinator for PNC3. Thus, if synchronized PNC shutdown is enabled, then PNC-Coor1 is responsible to initiate a synchronized PNC shutdown for PNC1 and PNC2. PNC-Coor3 is responsible to initiate a synchronized PNC shutdown for PNC3.

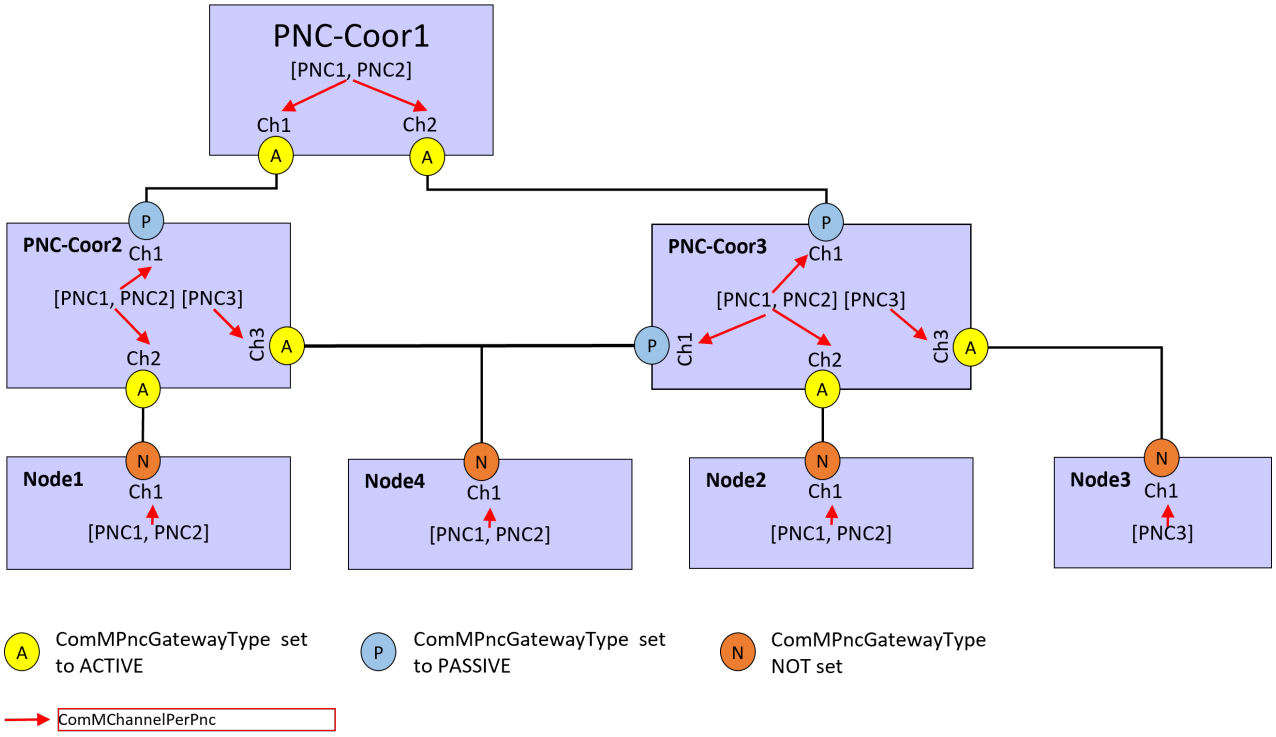


Figure 7.6: Multiple top-level PNC coordinators of the same PNC

In Figure 7.6 PNC-Coor2 and PNC-Coor3 act as top-level PNC coordinator for PNC3. For a proper behaviour, the NM filter mask at PNC-Coor3.Ch1 need to be configured such that PNC3 bit pass the filter. If synchronized PNC shutdown is enabled, then the expected behaviour should be as described in the following point:

- if PNC-Coor2 request the PNC3, then this PNC request is forwarded by PNC-Coor3 to Node3
- if only PNC-Coor2 has requested PNC3 and PNC-Coor2 release PNC3, then PNC-Coor2 transmit a PN shutdown message. PNC-Coor3 forward the synchronized PNC shutdown to PNC-Coor3.Ch3 (see [SWS_ComM_01082]). Therefore PNC3 synchronously shutdown on all affected nodes from PNC-Coor2, across PNC-Coor3 down to Node3. Thus, PNC-Coor2 act as the top-level coordinator for PNC3 in that scenario.
- If only Node3 request PNC3, then PNC-Coor3 consider only PNC-Coor3.Ch3 to mirror back the PNC request. The PNC request is not forwarded to any other channel for this PNC request, since PNC3 reference only PNC-Coor3.Ch3.

- If only Node3 has requested PNC3 and Node3 release PNC, then PNC-Coor3 transmit a PN shutdown message on PNC-Coor3.Ch3. Thus, PNC-Coor3 act as the top-level coordinator for PNC3 in that scenario.
- If Node3 and PNC-Coor2 has requested PNC3 and Node3 release PNC3, then the PNC3 request of PNC-Coor2 keep PNC3 requested.
- If Node3 and PNC-Coor2 has requested PNC3 and PNC-Coor2 release PNC3, then PNC-Coor2 transmit a PN shutdown message on PNC-Coor2.Ch3. PNC-Coor3 will not forward the PN shutdown request on PNC-Coor3-Ch3, since ERA of PNC-Coor3.Ch3 still contain PNC3 request from Node3 (see [SWS_ComM_01082]). Thus, PNC-Coor2 will asynchronously shutdown PNC3, since PNC-Coor3 will consider the PNC request from Node3 as long as Node3 request PNC3. After Node3 has released PNC3, PNC-Coor3 and Node3 shutdown PNC3 synchronously, sinc PNC-Coor3 act as top-level PNC coordinator

Note: The network topology and communication design has to ensure a valid and supported PN topology

7.1.5 Dynamic PNC-to-channel-mapping (optional)

This feature adds the possibility to update the PNC-to-channel-mapping of the PNC Gateway during runtime. This update works via a request-response-based learning process of all participating Nodes. When Partial Network learning is requested within the Nm PDUs, all participating Nodes will respond their current PNC membership on the corresponding channel and the PNC Gateway then updates the current PNC-to-channel-mapping accordingly.

[SWS_ComM_CONSTR_00004] ⌈If at least one channel is referenced by a PNC by using `ComMChannelPerTxOnlyPnc`, then `ComMDynamicPncToChannelMapping-Support` shall be set to FALSE. Otherwise the configuration is invalid. A configuration tool shall reject such a configuration as invalid (error).⌋

[SWS_ComM_01026]

Upstream requirements: SRS_ModeMgm_09260

⌈If the function ComM_Nm_PncLearningBitIndication has been called on a channel where ComMDynamicPncToChannelMappingEnabled is set to TRUE or when ComM calls Nm_PnLearningRequest on a channel ComM shall set the PNC Learning Phase to active for the according channel.⌋

[SWS_ComM_01029]

Upstream requirements: SRS_ModeMgm_09265

⌈If `ComMDynamicPncToChannelMappingEnabled` is set to TRUE and function ComM_Nm_RepeatMessageLeftIndication has been called ComM shall set the PNC Learning Phase to inactive for the according channel.⌋

[SWS_ComM_01028]

Upstream requirements: SRS_ModeMgm_09261

⌈If ComMPncGatewayEnabled is set to TRUE and the function ComM_Nm_PncLearningBitIndication has been called for a channel either of the following actions shall be performed:

- when ComM_Nm_PncLearningBitIndication is called for a channel where ComMPncGatewayType is set to COMM_GATEWAY_TYPE_ACTIVE, ComM shall forward the Learning Request by calling Nm_PnLearningRequest on all further coordinated ComM channels (active or passive) with ComMDynamicPncToChannelMappingEnabled is set to TRUE
- when ComM_Nm_PncLearningBitIndication is called for a channel where ComMPncGatewayType is set to COMM_GATEWAY_TYPE_PASSIVE, ComM shall forward the Learning Request by calling Nm_PnLearningRequest on ComM channels with ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE and ComMDynamicPncToChannelMappingEnabled is set to TRUE

⌋

Rational: Partial network learning bit needs to be forwarded to all nodes in the network but it needs not to be mirrored back even in the case when network topology contains circles.

[SWS_ComM_01090]

Upstream requirements: SRS_ModeMgm_09261

⌈If ComMPncGatewayEnabled and ComMPncDynamicMappingSupport are set to TRUE and when the PNC Learning Phase is active, then ComM shall forward received ERA Rx information on channels where ComMDynamicPncToChannelMappingEnabled is set to TRUE. ComM shall set the affected PNC bit(s) in all affected ERAn on all other channels where ComMDynamicPncToChannelMappingEnabled is set to TRUE considering the following rules:

- Rx ERA received on channels with ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE shall be forwarded on all other coordinated channels (active or passive)
- Rx ERA received on channel with ComMPncGatewayType set to COMM_GATEWAY_TYPE_PASSIVE shall be forwarded on all other channels where ComMPncGatewayType set to COMM_GATEWAY_TYPE_ACTIVE

⌋

7.1.5.1 Update PNC-to-channel-mapping

The PNC Gateway needs to be capable to update its PNC-to-channel Mapping on runtime.

[SWS_ComM_01091]

Status: DRAFT
Upstream requirements: SRS_ModeMgm_09258

⌈If ComMPncGatewayEnabled is set to TRUE and when the PNC Learning Phase is active and an PNC bit in the ERA is set to "1" on a channel where ComMDynamic-PncToChannelMappingEnabled is set to TRUE ComM shall set PNC-to-channel Mapping to 1 for every ComMPnc on the according channel where this PNC bit in the ERA has been set to "1" for the according PNC.⌋

7.1.5.2 PNC Membership Forwarding

Every participating Node has to transmit its current PNC membership during PNC Learning phase. The PNC Gateway needs additionally also forward PNC memberships received from other channels.

[SWS_ComM_01092]

Upstream requirements: SRS_ModeMgm_09262, SRS_ModeMgm_09250

⌈If ComMPncGatewayEnabled is set to FALSE and when the PNC Learning Phase is active, the ComM shall set the corresponding PNC bits in the IRA with the value of the current PNC membership and call Nm_UpdateIRA(<channel>, <IRA>) for all ComM channels where ComMDynamicPncToChannelMappingEnabled is set to TRUE.⌋

[SWS_ComM_01093]

Upstream requirements: SRS_ModeMgm_09261, SRS_ModeMgm_09250

⌈If ComMPncGatewayEnabled is set to TRUE and when the PNC Learning Phase is active, the ComM shall call Nm_UpdateIRA(<channel>, <IRA>) for all ComM channels where ComMDynamicPncToChannelMappingEnabled is set to TRUE with the IRA set with the value of the current PNC membership merged with the PNC information that needs to be forwarded according to [SWS_ComM_01090].⌋

7.1.6 Partial Networking Configuration Hints

The partial network configuration has to consider the configuration of the corresponding PN filter mask in NM of the corresponding NM-channels. If using a SystemDescriptionExtract to configure the BSW stack and the modelled partial network is available within the SystemDescriptionExtract, then the PN filter mask is derived automatically per each NM-channel. It is up to the integration process and the integration restriction to change the PN filter mask manually after the derivation. The integration process and particular restrictions is not dedined by AUTOSAR to support flexibility.

The following chapters describe the supported use cases to be considered for a proper PNC handling of PNC gateways and none PNC gateways

7.1.6.1 Bi-directional PNC handling

This means, that PNC requests are always transferred in both directions. The handling of PNC request is symmetrically for transmission and reception:

- PNC gateways forward incoming (external) PNC request and mirror them back
- None PNC gateways react on incoming PNC request and transmit PNC requests according to PNC assignment

Thus, ComM transmit and handle received PNC requests for a PNC on those ComMChannels, where a particular PNC refer to the ComMChannel by using the parameter ComMChannelPerPnc (see [ECUC_ComM_00880]). The correctness of received PNCs within the PNC Request Bit Vector according to the ComMChannel assignment has to be ensured by a proper configuration of the PN filter mask per NM-channel in the NmIf.

Note: ComM doesn't check the correctness of the received PNC according to the ComMChannel assignment:

- For EIRA updates, ComM has no possibility to check on which ComMChannel the PNC request was received, since the ComMChannel information is not forwarded by the NmIf.
- For ERA updates, a check could be done, but it was decided in AUTOSAR to release ComM from this responsibility.

In both cases (PNC gateway use case and none PNC gateway) the PN filter mask of a NM-channel have to pass all PNCs which are reference the corresponding ComMChannels via ComMChannelPerPnc

7.1.6.2 Uni-directional PNC handling

This means, that PNC requests are transferred in one direction. The handling of PNC request is asymmetrically for transmission and reception:

- PNC gateways forward incoming (external) PNC request but do not mirror it back on the ComMChannel the PNC request was received
- None PNC gateways transmit PNC requests for PNCs on ComMChannels, where this PNC is not assigned to

For PNC gateways the PN filter mask of a NM-channel has to pass all PNCs which are acceptable to be received on a ComMChannel and the PNCs do NOT refer the ComM-Channes via ComMChannelPerPnc (no PNC-channel relation exist). Addtionally, the PNC ERA handling has to be enabled for the according NM-channel. For received PNC requests on ComMChannel where no PNC-channel relation exit, only the forwarding of PNC requests and no mirroring back on the receiving ComMChannel will be performed. For received PNC requests on a ComMChannel where a PNC-channel relation exit, the bi-directional PNC handling will be performed.

The uni-directional PNC handling for PNC gateways could be used, e.g. when a network needs information from a certain PNC but there is no need to provide any information back.

For none PNC gateways the PN filter mask of a NM-channel has to reject all PNCs which are considered to be only transmitted on a ComMChannel. Received PNC request of those ComMChannel should not be handled and therefore should not reach ComM.

The uni-directional PNC handling for none PNC gateways could be used, e.g. when an ECU needs to wake-up or keep-alive some functionality without being part of it.

7.1.6.3 Transmission only PNC handling

This means, that internal PNC requests due to PNC coordination (i.e. triggered externally by a received PNC request (PNC bit set in the ERA)) are transferred for transmission. Thereby only the internal request array (IRA) is updated without requesting the according ComMChannel. A local ComMUser request which refer to this PNC, would result in ComMChannel request. This could be achieved via a proper configuration, such that a PNC refer to a ComMChannel via ComMChannelPerTxOnlyPnc.

Expected runtime behaviour:

- If a PNC refers to a ComMChannel (e.g. ComMChannel_A) by using the reference ComMChannelPerTxOnlyPnc (see [ECUC_ComM_00900]), this PNC refers to at least one further ComMChannel (e.g. ComMChannel_B) by using ComMChannelPerPnc (see [ECUC_ComM_00880]) and this PNC is requested externally by a received PNC request (PNC bit set in the ERA), then the corresponding PNC state machine transit to COMM_PNC_REQUESTED the IRA of all referenced ComMChannels are updated and the channel state machine of ComMChannel_B is requested, while the channel state machine of ComMChannel_A is NOT requested.
- If a PNC refers exclusively to ComMChannels by using the reference ComMChannelPerTxOnlyPnc (see [ECUC_ComM_00900]), this PNC is requested externally by a received PNC request (PNC bit set in the ERA) and no local user requests this PNC, then the corresponding PNC state machine and the according ComMChannels are not affected:
  - PNC statemachine stays in COMM_PNC_NO_COMMUNICATION and therefore IRAs for those ComMChannels are NOT updated
  - the referenced ComMChannel state machines are NOT requested
- If a PNC refer to a ComMChannel by using the reference ComMChannelPerTxOnlyPnc (see [ECUC_ComM_00900]) and this PNC is requested locally by ComMUser, then the corresponding PNC state machine transit to COMM_PNC_REQUESTED, IRA for this ComMChannel is updated and the referenced ComMChannel state machine is requested with COMM_FULL_COMMUNICATION.

- If a PNC refer to a ComMChannel by using the reference ComMChannelPerTxOnlyPnc (see [ECUC_ComM_00900]), this PNC is requested locally by ComMUser and additional externally by a received PNC request (PNC bit set in the ERA), then the corresponding PNC state machine transit to COMM_PNC_REQUESTED, IRA for this ComMChannel is updated and the referenced ComMChannel state machine is requested with COMM_FULL_COMMUNICATION. If the local ComMUser release the request for this PNC, then the ComMChannel will be released, but the IRA of this ComMChannel will still have the corresponding PNC bit set to '1' as long as the PNC is externally requested.
- If a PNC refer to a ComMChannel by using the reference ComMChannelPerTxOnlyPnc (see [ECUC_ComM_00900]), the ComMChannel is not referenced by another PNC via ComMChannelPerPnc and a wake up is detected, then the PNC statemachine will stay in COMM_PNC_NO_COMMUNICATION. (Please refer to [SWS_ComM_01063], [SWS_ComM_01064], [SWS_ComM_01065], [SWS_ComM_01066])

The transmission-only-PNC handling could be used e.g. for none PNC gateways to request only PNCs without additionally requesting the NM.

The transmission only PNC handling could be used e.g. for PNC gateways to receive uni-directional PNC request (PNC1) on one channel (channel A) and forward the PNC request without requesting the NM on another channel (channel B). On channel B PNC1 is configured for bi-directional PNC handling, therefore a received PNC request for PNC1 is forwarded to channel A by considering to request the affected ComMChannels and the according NM.

Note: The reference ComMChannelPerTxOnlyPnc cannot be derived from a SystemDescriptionExtract. The reference from a PNC to a ComM channel via ComMChannelPerTxOnlyPnc could only be added manually within the integration phase.

7.2 ComM channel state machine

[SWS_ComM_00979]

Upstream requirements: SRS_ModeMgm_09243

⌈If the optional PNC functionality is enabled (see [ECUC_ComM_00839]), all PNC actions shall be performed before the channel related actions are executed.⌋

[SWS_ComM_00980]

Upstream requirements: SRS_ModeMgm_09243

⌈If the parameter ComMPncNmRequest is set to TRUE (see [ECUC_ComM_00886]), if the "FULL Communication" is requested due to a change in the PNC state machine to COMM_PNC_REQUESTED (see [SWS_ComM_01068]) 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]

Upstream requirements: SRS_ModeMgm_09080

⌈ComM shall implement one channel state machine for every communication channel independently.⌋

Note to [SWS_ComM_00051]: The channel state maching is shown in Figure 7.7 and an overview of the requirements for the channel state machine transtions is listed in Table 7.1

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.

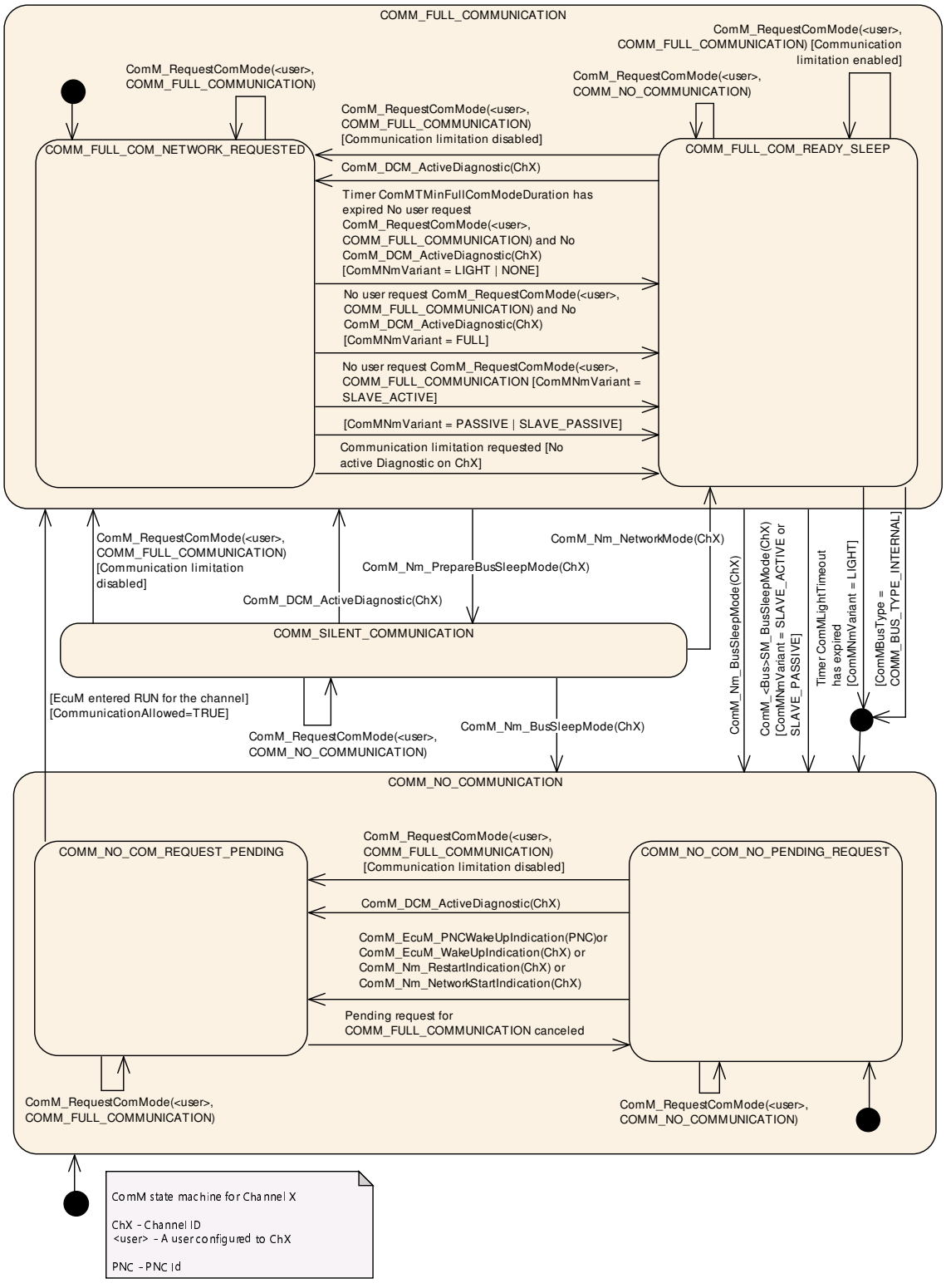


Figure 7.7: ComM channel state machine

State	Section / Requirement
COMM_NO_COMMUNICATION	Chapter 7.2.2 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], [SWS_ComM_01014], [SWS_ComM_01015] In sub-state COMM_NO_COM_REQUEST_PENDING: [SWS_ComM_00895] ,[SWS_ComM_00897]
COMM_SILENT_COMMUNICATION	Chapter 7.2.3 Entering state: [SWS_ComM_00071] In state: [SWS_ComM_00877], [SWS_ComM_00878] [SWS_ComM_00295], [SWS_ComM_00296]
COMM_FULL_COMMUNICATION	Chapter 7.2.4 Entering state: [SWS_ComM_00069] In state: [SWS_ComM_00637], [SWS_ComM_00826] Chapter 7.2.4.1 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] Chapter 7.2.4.2 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], [SWS_ComM_01014], [SWS_ComM_01015]
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 7.1: Link to detailed explanation of the channel state machine resp. transition

[SWS_ComM_00879]

Upstream requirements: SRS_ModeMgm_09083

⌈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]

Upstream requirements: SRS_ModeMgm_09083

⌈The COMM_FULL_COMMUNICATION state shall have two sub-states COMM_FULL_COM_NETWORK_REQUESTED and COMM_FULL_COM_READY_SLEEP.⌋

[SWS_ComM_00881]

Upstream requirements: SRS_ModeMgm_09083

⌈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]

Upstream requirements: SRS_ModeMgm_09083

⌈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

Note for [SWS_ComM_00896]: It is assumed, that CommunicationAllowed is set to TRUE via ComM_CommunicationAllowed() after a ECU was woken up and all potential validation checks are finalized to ensure a proper communication via the corresponding channel. Therefore no transition from COMM_NO_COM_REQUEST_PENDING to COMM_NO_COM_NO_PENDING_REQUEST is available, since CommunicationAllowed

is already set to TRUE or will be set to TRUE via ComM_CommunicationAllowed() within the wake-up processing of an ECU.

[SWS_ComM_00472]

Upstream requirements: SRS_ModeMgm_09085

⌈Main state changes (see [SWS_ComM_00879]) shall be indicated to the users with the corresponding notifications. Exception: Default state after initialization, see [SWS_ComM_00313].⌋

Comment on [SWS_ComM_00472]: 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.

Note for [SWS_ComM_00472]: For more details regarding the notification refer to Chapter 8.5.1.5 and Chapter 8.5.1.6.

[SWS_ComM_00191]

Upstream requirements: SRS_ModeMgm_09172, SRS_ModeMgm_09149

⌈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 7.2.

Communication Mode	Message Transmission	Message Reception	NM (ComMNm-Variant=FULL)	Wake-up/Restart capability
COMM_FULL_COMMUNICATION	On	On	Bus communication requested	N/A
COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST	On	On	Bus communication requested	Request the lower layer to trigger a wake-up on the network
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 7.2: Granted communication capabilities in the corresponding modes

[SWS_ComM_01056]

Upstream requirements: SRS_ModeMgm_09268

⌈Requests for communication mode COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST shall be handled as request for COMM_FULL_COMMUNICATION within the ComM channel state machine. Deviations of ComM channel machine state transitions and behavior within the states are specified explicitly.⌋

Notes 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).
- 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 ComM managed and managing channels

A ComM channel could reference other ComM channels. The reference is configurable by setting `ComMManageReference` (see [ECUC_ComM_00893]). The source ComM channel of a `ComMManageReference` is called "managing channel" and the target ComM channel is called "managed channel". A managing channel could reference 0..n managed channels. A managed channel could be referenced by exclusively 1 managing channel.

This is used to support use cases, were a managing channel handle the interaction with the NM module and the managed channel has no NM.

Note: The following limitation have to be considered for a managing channel:

- ComMNmVariant of a managing channel is set to `FULL` (see [ECUC_ComM_00568])

Note: The following limitations have to be considered for a managed channel:

- ComMNmVariant of a managed channel is set to `LIGHT`, since the managing channel is responsible for the interaction with the NmChannel (see [ECUC_ComM_00568])
- `ComMPncGatewayType` of a managed channel is neither set to `COMM_GATEWAY_TYPE_ACTIVE` nor `COMM_GATEWAY_TYPE_PASSIVE` (see [ECUC_ComM_00842])

7.2.2 Behavior in state `COMM_NO_COMMUNICATION`

[SWS_ComM_00898]

Upstream requirements: SRS_ModeMgm_09247

⌈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]

Upstream requirements: SRS_ModeMgm_09085

⌈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]

Upstream requirements: SRS_ModeMgm_09083

⌈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 (<Bus>SM_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.

[SWS_ComM_00288]

Upstream requirements: SRS_ModeMgm_09132

⌈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().⌋

Note: Nm_NetworkRelease is needed if ComM has requested the NM (Nm_NetworkRequest or Nm_PassiveStartup) for that channel before and has not yet released it.

Rationale for [SWS_ComM_00073], [SWS_ComM_00288] , [SWS_ComM_00875] and [SWS_ComM_00876]: FlexRay shutdown cannot be interrupted to avoid partial networks.

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.

7.2.2.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state

[SWS_ComM_00875]

Upstream requirements: SRS_ModeMgm_09083

⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST and user requests COMM_FULL_COMMUNICATION and communication limitation is disabled, the ComM channel state machine shall immediately switch to sub-state COMM_NO_COM_REQUEST_PENDING.⌋

Note to [SWS_ComM_00875]: For more details on communication limitation refer to Chapter 7.4.1

[SWS_ComM_00876]

Upstream requirements: SRS_ModeMgm_09083, SRS_ModeMgm_09085

⌈In sub-state COMM_NO_COM_NO_PENDING_REQUEST ,configuration parameter ComMNmVariant=FULL|LIGHT|NONE (see [ECUC_ComM_00568]) and DCM indicate ComM_DCM_ActiveDiagnostic (see [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 Chapter 7.4.1) shall temporarily be inactive during an active diagnostic session (see [SWS_ComM_00182])

Note for [SWS_ComM_00876]: For diagnostic activation it is assumed that diagnostic tester keeps the bus awake, therefore no special handling needed for managed channels.

[SWS_ComM_00893]

Upstream requirements: SRS_ModeMgm_09087

⌈If ComM_EcuM_WakeUpIndication is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp is set to FALSE (see [ECUC_ComM_00695]), the ComM module shall switch the requested ComM channel state machine (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING. If the indicated ComM channel is a managed channel, then the ComM channel state machine of the referencing managing channel (see [ECUC_ComM_00893]) shall also be switched to sub-state COMM_NO_COM_REQUEST_PENDING.⌋

[SWS_ComM_00894]

Upstream requirements: SRS_ModeMgm_09087

⌈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.⌋

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]

Upstream requirements: SRS_ModeMgm_09248

⌈If ComM_EcuM_WakeUpIndication is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp is set to TRUE (see [ECUC_ComM_00695]), the ComM module shall switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.⌋

[SWS_ComM_01014]

Upstream requirements: SRS_ModeMgm_09248

⌈If ComM_EcuM_PNCWakeUpIndication(<PNC>) (see [SWS_ComM_91001]) is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST, the configuration parameters ComMSynchronousWakeUp is set to FALSE (see [ECUC_ComM_00695]) and ComMPncSupport is set to TRUE (see [ECUC_ComM_00839]), the ComM module shall switch those ComM channel state machines (resp. channels) which are referenced via ComMChannelPerPnc (see [ECUC_ComM_00880]) by the given PNC to sub-state COMM_NO_COM_REQUEST_PENDING.⌋

Note for [SWS_ComM_01014]:

- This includes ComM channel state machines of managing channels, which are referenced by the indicated managed channels, as ComMPncs reference always both types (see [11] [constr_3484])
- A channel which is referenced via ComMChannelPerTxOnlyPnc, ComMSynchronousWakeUp is set to FALSE and a PNC related wake-up is configured (call of EcuM_ComM_PNCWakeupIndication()), need always a EcuMWakeupSource which refer to this channel to ensure a proper ComMChannel handling. A wake up on this channel would bring the channel to a defined state, but the PNC will stay in COMM_PNC_NO_COMMUNICATION, since this PNC reference the channel via ComMChannelPerTxOnlyPnc

[SWS_ComM_01015]

Upstream requirements: SRS_ModeMgm_09248

⌈If ComM_EcuM_PNCWakeUpIndication(<PNC>) (see [SWS_ComM_91001]) is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameters ComMSynchronousWakeUp is set to TRUE (see [ECUC_ComM_00695]) and

ComMPncSupportis set to TRUE (see [ECUC_ComM_00839]), the ComM module shall switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.⌋

7.2.2.2 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 is set to TRUE, the ComM channel state machine shall immediately switch to state COMM_FULL_COMMUNICATION.⌋

[SWS_ComM_00897]

Upstream requirements: SRS_ModeMgm_09083

⌈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]: This enable 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)(see [SWS_ComM_00871]) or DCM indicated ComM_DCM_ActiveDiagnostic(<channel>) (see [SWS_ComM_00873]), but now canceled with ComM_RequestComMode(<user>,COMM_NO_COMMUNICATION) (see [SWS_ComM_00871]) or DCM ComM_DCM_InactiveDiagnostic(<channel>) (see [SWS_ComM_00874]).

7.2.3 Behavior 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 (<Bus>SM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_SILENT_COMMUNICATION), see [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]

Upstream requirements: SRS_ModeMgm_09246

⌈In state COMM_SILENT_COMMUNICATION and user requests COMM_FULL_COMMUNICATION and communication limitation is disabled, the ComM channel state machine shall switch to state COMM_FULL_COMMUNICATION.⌋

Note to [SWS_ComM_00877]: For more details on communication limitation refer to Chapter 7.4.1

[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 Chapter 7.4.1) 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 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.⌋

7.2.4 Behavior in state COMM_FULL_COMMUNICATION

[SWS_ComM_00899]

Upstream requirements: SRS_ModeMgm_09083

⌈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]

Upstream requirements: SRS_ModeMgm_09268

⌈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:

- If Communication Mode COMM_FULL_COMMUNICATION was requested , then <Bus>SM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_FULL_COMMUNICATION) shall be called
- If Communication Mode COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST was requested and ComMWakeupSleepRequestEnabled of the ComM channel is set to TRUE, then <Bus>SM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST) shall be called
- If Communication Mode COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST was requested and ComMWakeupSleepRequestEnabled of the ComM channel is set to FALSE or not available, then <Bus>SM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_FULL_COMMUNICATION) shall be called

⌋

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

[SWS_ComM_01057] Handling if COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST is called for a ComMChannel that has ComMWakeupSleepRequestEnabled set to TRUE

Upstream requirements: SRS_ModeMgm_09268

⌈Every time a ComM channel is requested with COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST and ComMWakeupSleepRequestEnabled of the ComM channel is set to TRUE, ComM shall request the corresponding network of the ComM channel by calling

<Bus>SM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST), even if the ComM channel is already in state COMM_FULL_COMMUNICATION.⌋

Note: The re-trigger of the <Bus>SM state machine is used to trigger a wake-up on the network, if the used hardware is supporting such a functionality (e.g. Ethernet hardware compliant to OA TC10 (see [2]))

[SWS_ComM_00637]

Upstream requirements: SRS_ModeMgm_09083

⌈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_01018] Handling upon indication of ComM_<Bus>SM_BusSleepMode for ComMChannels with ComMNmVariant=SLAVE_ACTIVE | SLAVE_PASSIVE

Upstream requirements: SRS_ModeMgm_09266, SRS_ModeMgm_09267

⌈In state COMM_FULL_COMMUNICATION and configuration parameter ComMNmVariant=SLAVE_ACTIVE | SLAVE_PASSIVE and the Bus State Manager module indicates ComM_<Bus>SM_BusSleepMode, the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION.⌋

[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.

7.2.4.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state

[SWS_ComM_00886]

Upstream requirements: SRS_ModeMgm_09087

⌈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]

Upstream requirements: SRS_ModeMgm_09132

⌈On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED from COMM_NO_COM_REQUEST_PENDING and EcuM module has indicated a wake-up by ComM_EcuM_WakeUpIndication(<channel>) (see [SWS_ComM_00275]) or by ComM_EcuM_PNCWakeUpIndication(<PNC>) (see [SWS_ComM_91001]), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management. If the indicated ComM channel is a managed channel, the ComM

module shall request Nm_PassiveStartup(<referencing managing channel>) (see [ECUC_ComM_00893]) from the Network Management.⌋

[SWS_ComM_01016] ⌈If the indicated ComM channel is a managed channel, the ComM module shall request Nm_PassiveStartup(<referencing managing channel>) (see [ECUC_ComM_00893]) from the Network Management.⌋

[SWS_ComM_00902]

Upstream requirements: SRS_ModeMgm_09132

⌈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]

Upstream requirements: SRS_ModeMgm_09132

⌈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 Chapter 7.7.1)

[SWS_ComM_00869]

Upstream requirements: SRS_ModeMgm_00049

⌈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.⌋

Note: Additionally Nm_NetworkRequest may be invoked due to [SWS_ComM_00980].

[SWS_ComM_00870]

Upstream requirements: SRS_ModeMgm_00049

⌈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.⌋

[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 (see [ECUC_ComM_00568]) and no user request ComM_RequestComMode(<user>,COMM_FULL_COMMUNICATION) and the DCM does not indicate ComM_DCM_ActiveDiagnostic(<channel>)(see [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_01017]

Upstream requirements: SRS_ModeMgm_09266

⌈In sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=SLAVE_ACTIVE ([ECUC_ComM_00568]) and no user request ComM_RequestComMode(<user>,COMM_FULL_COMMUNICATION), the ComM channel state machine shall switch to sub-state COMM_FULL_COM_READY_SLEEP.⌋

[SWS_ComM_00915]

Upstream requirements: SRS_ModeMgm_09267

⌈In sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=PASSIVE | SLAVE_PASSIVE ([ECUC_ComM_00568]), the ComM channel state machine shall switch to sub-state COMM_FULL_COM_READY_SLEEP.⌋

[SWS_ComM_00890]

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09087

⌈In sub-state COMM_FULL_COM_NETWORK_REQUESTED and the DCM does not indicate ComM_DCM_ActiveDiagnostic(<channel>)(see [SWS_ComM_00873]) and communication limitation is requested (see section [REF]), ComM channel state machine shall immediately switch to sub-state COMM_FULL_COM_READY_SLEEP and cancel the timer for ComMTMinFullComModeDuration.⌋

7.2.4.2 COMM_FULL_COM_READY_SLEEP sub-state

[SWS_ComM_00133]

Upstream requirements: SRS_ModeMgm_09132

⌈On entering sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMNmVariant=FULL (see [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 (see [ECUC_ComM_00568]), the timer for ComMNmLightTimeout (see [ECUC_ComM_00606]) shall be started.⌋

[SWS_ComM_00610] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMNmVariant=LIGHT (see [ECUC_ComM_00568]), this ComMChannel has no PNC relation (either ComMPncSupport is set to FALSE or this ComMChannel is not referenced by a PNC) and the timer for ComMNmLightTimeout (see [ECUC_ComM_00606]) has expired, the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION.⌋

[SWS_ComM_01095] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMNmVariant=LIGHT (see [ECUC_ComM_00568]), this ComMChannel is referenced by a PNC and the timer for ComMNmLightTimeout (see [ECUC_ComM_00606]) has expired, the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION as soon as all referencing PNCs reside in COMM_PNC_NO_COMMUNICATION.⌋

Note: [SWS_ComM_01095] prevents a ComMChannel to transit to COMM_NO_COMMUNICATION, if this ComMChannel acts in the role of a managed channel, this ComMChannel is referenced by at least one PNC and the PNC is requested passively (PNC reside in COMM_PNC_READY_SLEEP).

[SWS_ComM_01096] ⌈In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMNmVariant=LIGHT (see [ECUC_ComM_00568]), this ComMChannel act in role of an managed channel and is referenced by a ComMChannel in the role of a managing channel but not referenced by any PNC and the timer for ComMNmLightTimeout (see [ECUC_ComM_00606]) has expired, the ComM channel state machine shall switch to state COMM_NO_COMMUNICATION as soon as the referencing ComMChannel (managing channel) transit to COMM_PNC_NO_COMMUNICATION.⌋

Note: [SWS_ComM_01096] prevents a ComMChannel to transit to COMM_NO_COMMUNICATION, if this ComMChannel acts in the role of a managed channel, this ComMChannel is referenced by a ComMChannel in the role of a managing channel without any referencing PNC and this ComMChannel is requested passively (ComM channel statemachine reside in COMM_FULL_COM_READY_SLEEP).

[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 [REF]), 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 [REF]) 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 ComM User to PNC Relations

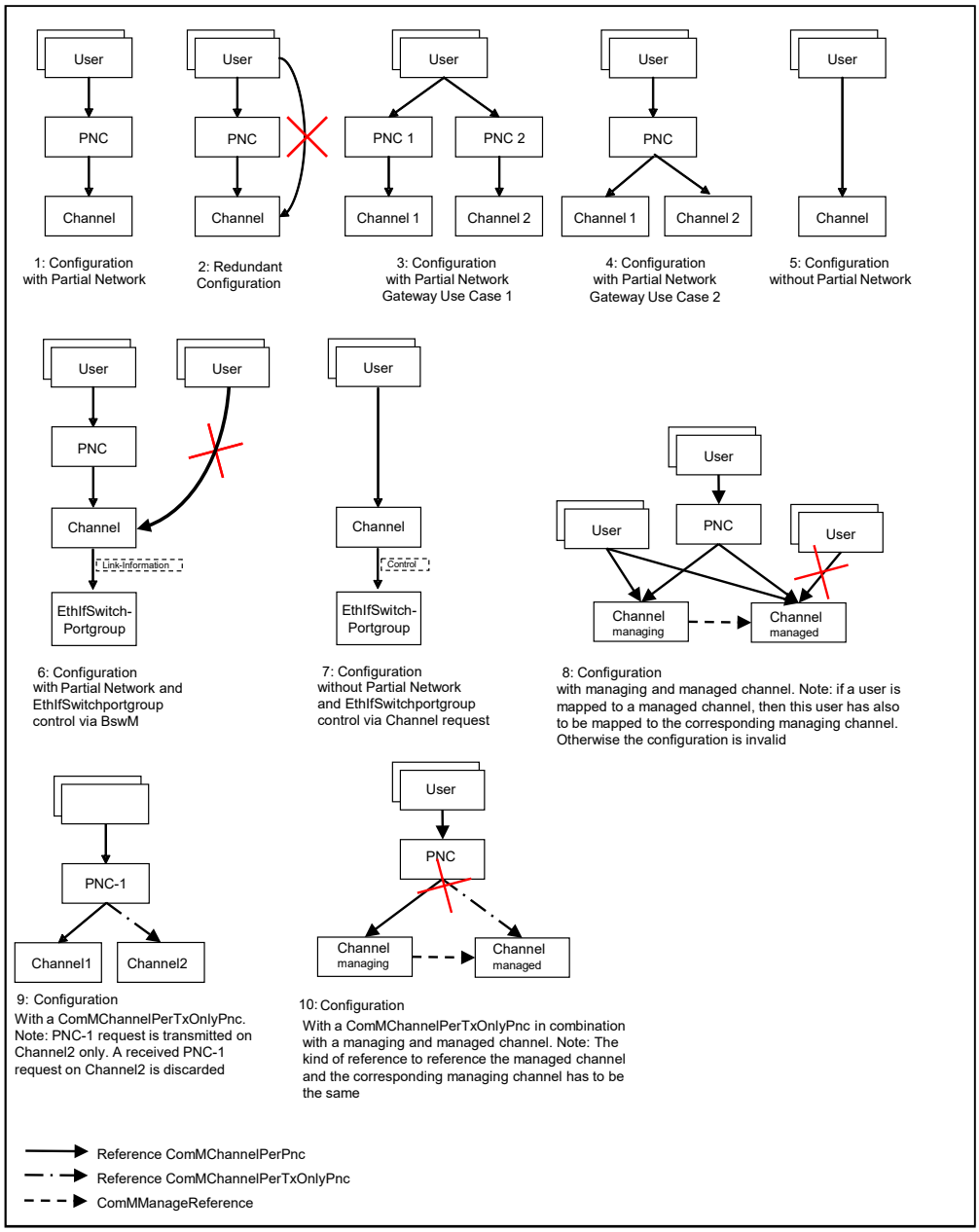


Figure 7.8: User to Partial network and channel Mapping Use Cases

[SWS_ComM_00994] ⌈No restrictions from the configuration of the <Bus>Nm Filter for partial networking shall apply to ComM user assignment to PNCs.⌋

Comment: The <Bus>NM Filter configuration shall be independent 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]

Upstream requirements: SRS_ModeMgm_09133, SRS_ModeMgm_09090

⌈It shall be possible to map a configurable amount of ComMUsers to one or more ComM channels using the parameter ComMUserPerChannel.⌋

Comment:

1. The existing mapping of ComM users to system channels shall still be possible for backward compatibility. (i.e. the configuration containers will stay untouched)
2. 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.

1

[SWS_ComM_00912]

Upstream requirements: SRS_ModeMgm_09246

⌈It shall be possible to map a configurable amount of ComMUsers to one or more PNCs using the parameter ComMUserPerPnc (see [ECUC_ComM_00876]).⌋

[SWS_ComM_01094]

Upstream requirements: SRS_ModeMgm_09246

⌈It shall be possible to map a configurable amount of PNC(s) to a configurable amount of ComM channels by using the parameter ComMChannelPerPnc (see [ECUC_ComM_00880]) or ComMChannelPerTxOnlyPnc (see [ECUC_ComM_00900]). The mapping shall be possible for all ComMChannels in combination with the following ComMNmVariants:

- ComMVariant=FULL
- ComMVariant=LIGHT, if the ComMChannel is in the role of a managed ComMChannel and the corresponding managing ComMChannel is also mapped to this PNC

⌋

Note to [SWS_ComM_01094]: For more details regarding managed and managing ComM channels refer to Chapter 7.2.1

[SWS_ComM_00996]

Upstream requirements: SRS_ModeMgm_09246

⌈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.⌋

Rational for [SWS_ComM_00996]: Avoid redundant configuration since the channel is implicitly already referenced by the PNC.

Note on [SWS_ComM_00996]: For more details refer to Figure 7.8 "use case 2"

[SWS_ComM_CONSTR_00001] ⌈ComM channel's that are referenced by a PNC are not allowed to be referenced by any ComMUsers, if the PNC references at least one EthIfSwitchPortGroup . A configuration tool shall reject such a configuration as invalid (error). This constraint is only valid for a host ecu that control an Ethernet switch. In all other UseCases ComMChannels can be referenced by a PNC's and ComMUsers.⌋

Rational on [SWS_ComM_CONSTR_00001]: If using PN in combination with EthIfSwitchPortGroups (derivation from a SystemDescriptionExtract if EcuInstance.ethSwitchPortGroupDerivation is set to TRUE), then the EthIfSwitchPortGroups are switched by the EthIf_SwitchPortGroupRequestMode API and not by a channel request.

Note for [SWS_ComM_CONSTR_00001]: For more details refer to Figure 7.8 "use Case 6".

[SWS_ComM_CONSTR_00002] ⌈If a ComM user reference a managed channel, then this ComM user shall also reference the corresponding managing channel. Otherwise the configuration is invalid. A configuration tool shall reject a configuration as invalid (error), if a user references a managed channel without referencing the corresponding managing channel.⌋

[SWS_ComM_CONSTR_00003]

Status: DRAFT

⌈ComM channels with ComMNmVariant = SLAVE_PASSIVE are not allowed to be referenced by any ComMUser or PNC. A configuration tool shall reject such a configuration as invalid (error).⌋

Rational: ComM channels with ComMNmVariant = SLAVE_PASSIVE shall always follow the communication request of their communication master and are not allowed to request the corresponding master to wake-up the communication channel.

[SWS_ComM_CONSTR_00005] ⌈If a PNC references a ComM channel, then this PNC shall reference that ComM channel either using ComMChannelPerPnc or ComMChannelPerTxOnlyPnc, but not both. Otherwise the configuration is invalid. A configuration tool shall reject such a configuration as invalid (error).⌋

[SWS_ComM_CONSTR_00006] ⌈The kind of reference (either ComMChannelPerPnc or ComMChannelPerTxOnlyPnc) from a PNC to a managed channel and the corresponding managing channel shall be the same. Otherwise the configuration is invalid. A configuration tool shall reject such a configuration as invalid (error).⌋

7.4 Extended functionality

[SWS_ComM_00470]

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157, SRS_ModeMgm_09089, SRS_ModeMgm_09155

⌈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]).

7.4.1 Communication inhibition

The purpose of communication inhibition is to limit the communication capabilities. For details see section 7.4.1.2 “Bus wake up inhibition” and section 7.4.1.3 “Limit to `COMM_NO_COMMUNICATION` mode”. The utilization of communication inhibition is manageable. For details see section 7.4.1.1 “Manage communication inhibition”. Some examples are available to denote the communication inhibition (see section 7.4.1.4 “Examples for communication inhibition function”).

Note: some of the communication inhibition function states could require handling with non-volatile memory to survive an ECU reset:

- ECU start-up behaviour is specified in [SWS_ComM_00864] to re-store the values from non-volatile memory if available. Otherwise use the values of the corresponding ECUC parameter.
- A call of `ComM_DeInit` would trigger the storage to NvM (see [SWS_ComM_00865]), if a NvM block descriptor is configured

[SWS_ComM_00301]

Upstream requirements: SRS_ModeMgm_09071

⌈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]

Upstream requirements: SRS_ModeMgm_09071

⌈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]

Upstream requirements: SRS_ModeMgm_09155

⌈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]

Upstream requirements: SRS_ModeMgm_09155, SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈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]

Upstream requirements: SRS_ModeMgm_09157

⌈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>)` (see [SWS_ComM_00873]) and the end of a diagnostic session with `ComM_DCM_InactiveDiagnostic(<channel>)` (see [SWS_ComM_00874]).

7.4.1.1 Manage communication inhibition

In principle the utilization of communication inhibition is manageable. The execution permission of communication inhibition functionality is controlled via an inhibition status field (bit field). Each bit represents one communication inhibition functionality. The value of the control bit classifies, if the corresponding communication inhibition has the permission to be executed. The inhibition status field is stored non-volatile to survive an

ECU reset. ComM restores the inhibition status field at start-up, if the inhibition status field is available in a non-volatile memory. Otherwise ComM uses the configuration of ComMEcuGroupClassification as default.

Note: The layout of the inhibition status field is defined in [SWS_ComM_00669].

The following parameters are relevant for communication inhibition and have a relationship to APIs:

- ComMNoCom: "request bit" of mode inhibition (limit to NoCom), can be controlled by ComM_LimitChannelToNoComMode and ComM_LimitECUToNoComMode, only if ComMEcuGroupClassification enables this functionality (see [ECUC_ComM_00563], [SWS_ComM_00163], [SWS_ComM_00124]).
- ComMNoWakeup: "request bit" of mode inhibition (wakeup inhibition), can be controlled by ComM_PreventWakeUp, only if ComMEcuGroupClassification enables this functionality (see [ECUC_ComM_00563], [SWS_ComM_00156]).
- ComMEcuGroupClassification: "mask bits" of mode inhibition behavior, can be controlled by ComM_SetECUGroupClassification, regardless of ComMNoCom and ComMNoWakeup values

[SWS_ComM_01102] Call of ComM_SetECUGroupClassification

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈If ComM_SetECUGroupClassification is called, ComM shall update the internal status field with the values of the given inhibition status (Status).⌋

Note: A change of the internal status field has no impact on a activated inhibition function (see 7.4.1.4.2 “Wake up inhibition - example 2” for an example)

[SWS_ComM_01103] Call of communication inhibition function

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈If a communication inhibition function is called and the corresponding bit in the inhibition status field is set (0b1), then the communication inhibition function shall be executed. Otherwise the function shall ignore the inhibition function request and return with E_NOT_OK.⌋

[SWS_ComM_01104] Non volatile storage of ComMEcuGroupClassification

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈ComMEcuGroupClassification status shall be stored non-volatile, if it was changed during runtime via call of ComM_SetECUGroupClassification and ComMEcuGroupClassificationNvMStorage is set to TRUE.⌋

[SWS_ComM_01105] Default values of `ComMEcuGroupClassification`

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈The values of `ComMEcuGroupClassification` configured in the ECUC shall be considered as default values and used, if the `ComMEcuGroupClassification` status is not available in non-volatile memory.⌋

7.4.1.2 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]

Upstream requirements: SRS_ModeMgm_09089, SRS_ModeMgm_09141

⌈Bus wake up Inhibition shall be performed by ignoring user requests.⌋

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]

Upstream requirements: SRS_ModeMgm_09141

⌈A communication request (`COMM_FULL_COMMUNICATION`) by a user shall be inhibited if the ComM Inhibition status is equal to `ComMNoWakeup` is set to TRUE (see [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]

Upstream requirements: SRS_ModeMgm_09141

⌈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]

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09141

⌈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] Non volatile storage of ComMNoWakeup status

Upstream requirements: SRS_ModeMgm_09089, SRS_ModeMgm_09141

⌈ComMNoWakeup status shall be stored non volatile, if it was changed during runtime via call of ComM_PreventWakeUp and ComMNoWakeUpInhibitionNvmStorage is set to TRUE.⌋

[SWS_ComM_01106] Default values of ComMNoWakeup

Upstream requirements: SRS_ModeMgm_09089, SRS_ModeMgm_09141

⌈The value of ComMNoWakeup configured in the ECUC shall be considered as default value and used, if ComMNoWakeup status is not available in a non-volatile memory.⌋

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]

Upstream requirements: SRS_ModeMgm_09155, SRS_ModeMgm_09141

⌈The status of the user requests shall also be updated if a user releases a request.⌋

7.4.1.3 Limit to COMM_NO_COMMUNICATION mode

[SWS_ComM_00303]

Upstream requirements: SRS_ModeMgm_09071

⌈If the current state is COMM_FULL_COM_NETWORK_REQUESTED and when mode limitation to COMM_NO_COMMUNICATION has been requested for the corresponding channel, ComM module shall switch to COMM_FULL_COM_READY_SLEEP state to initiate a shutdown despite any user requests for COMM_FULL_COMMUNICATION.⌋

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

Comment for [SWS_ComM_00303]: Limit to COMM_NO_COMMUNICATION will only be performed if a channel was request actively. In that case all current user requests for full communication or even new requests will be ignored (see also [SWS_ComM_00215]). The limit to no communication will not be performed, if a ComM channel is remotely kept awake due to a passive wakeup.

[SWS_ComM_00842]

Upstream requirements: SRS_ModeMgm_09071

⌈When ComM_LimitChannelToNoComMode() is called, ComM module shall update the inhibition status (limitation to COMM_NO_COMMUNICATION) for the corresponding channel.⌋

Note: An update of the inhibition status due to a request for limit to COMM_NO_COMMUNICATION has to be performed always, independent of the current state.

[SWS_ComM_00355]

Upstream requirements: SRS_ModeMgm_09071

⌈If ComMResetAfterForcingNoComm is set to TRUE (see [ECUC_ComM_00558]) and when ComM enters COMM_NO_COMMUNICATION after state transition from COMM_FULL_COM_NETWORK_REQUESTED to COMM_FULL_COM_READY_SLEEP has been forced due to mode limitation to COMM_NO_COMMUNICATION request (see [SWS_ComM_00303]), then ComM shall call BswM_ComM_InitiateReset().⌋

Note: A call of BswM_ComM_InitiateReset() is the trigger for an ECU reset which has to be executed as soon as possible, depending on further needed actions (e.g. storing all NvM blocks).

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_00215]

Upstream requirements: SRS_ModeMgm_09071

⌈When mode limitation to COMM_NO_COMMUNICATION has been requested, ComM module shall ignore all user requests with COMM_FULL_COMMUNICATION for the corresponding channel.⌋

[SWS_ComM_00582]

Upstream requirements: SRS_ModeMgm_09078

⌈The ComM module shall clear the user requests after all the channels that belong to the corresponding user enter COMM_NO_COMMUNICATION mode.⌋

Note: [SWS_ComM_00582] describes the desired behaviour after an executed mode limitation.

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

[SWS_ComM_01107] Initialization value of ComM inhibition status "limit to COMM_NO_COMMUNICATION mode"

Upstream requirements: SRS_ModeMgm_09078

⌈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 is not stored persistently

7.4.1.4 Examples for communication inhibition function

7.4.1.4.1 Wake up inhibition - example 1

Precondition:

- ECU_A has a ComM with ComM.Ch1 and ComM.Ch2 configured
- ComMUser1 is assigned to ComM.Ch1 and ComM.Ch2
- ComMUser2 is assigned to ComM.Ch2
- All ComM.User request COMM_NO_COMMUNICATION and all ComM channels reside in COMM_NO_COMMUNICATION
- No passive request is available. Thus, no ComM channel is remotely kept awake
- Internal inhibition field is set to 0x01 (Note: This allows to execute "wakeup inhibition" and disallows to execute "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00669]))

Runtime:

- ComMUser1 requests COMM_FULL_COMMUNICATION (Note: ComM.Ch1 and ComM.Ch2 are requested with COMM_FULL_COMMUNICATION
- A "wakup inhibition" for ComM.Ch2 is activated via a call of ComM_PreventWakeUp. (Note: current channel state for ComM.Ch2 is not impacted. Thus, ComM.Ch2 stays in COMM_FULL_COMMUNICATION)
- ComMUser1 requests COMM_NO_COMMUNICATION (Note: ComM.Ch1 and ComM.Ch2 are requested with COMM_NO_COMMUNICATION)
- ComMUser2 requests COMM_FULL_COMMUNICATION (Note: user request is stored but not executed, due to active "wakeup inhibition" for ComM.Ch2, thus ComM.Ch2 stays in COMM_NO_COMMUNICATION)
- ComMUser1 requests COMM_FULL_COMMUNICATION (Note: ComM.Ch1 is requested with COMM_FULL_COMMUNICATION, ComM.Ch2 stays in COMM_NO_COMMUNICATION, due to active "wakeup inhibition")
- ComMUser1 requests COMM_NO_COMMUNICATION (ComM.Ch1 is requested with COMM_NO_COMMUNICATION, ComM.Ch2 stays in COMM_NO_COMMUNICATION, due to active "wakeup inhibition")
- The "wake up inhibition" for ComM.Ch2 is deactivated via a call of ComM_PreventWakeUp. (Note: ComM.Ch2 is requested with COMM_FULL_COMMUNICATION, since ComMUser2 still requests COMM_FULL_COMMUNICATION)

7.4.1.4.2 Wake up inhibition - example 2

Precondition:

- ECU_A has a ComM with ComM.Ch1 and ComM.Ch2 configured
- ComMUser1 is assigned to ComM.Ch1 and ComM.Ch2
- ComMUser2 is assigned to ComM.Ch2
- All ComM.User request COMM_NO_COMMUNICATION and all ComM channels reside in COMM_NO_COMMUNICATION
- No passive request is available. Thus, no ComM channel is remotely kept awake
- Internal inhibition field is set to 0x01 (Note: This allows to execute "wakeup inhibition" and disallows to execute "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00669]))

Runtime:

- ComMUser1 requests COMM_FULL_COMMUNICATION (Note: ComM.Ch1 and ComM.Ch2 are requested with COMM_FULL_COMMUNICATION

- A "wakeup inhibition" for ComM.Ch2 is activated via a call of ComM_PreventWakeUp. (Note: current channel state for ComM.Ch2 is not impacted. Thus, ComM.Ch2 stays in COMM_FULL_COMMUNICATION)
- ComMUser1 requests COMM_NO_COMMUNICATION (Note: ComM.Ch1 and ComM.Ch2 are requested with COMM_NO_COMMUNICATION)
- ComMUser2 requests COMM_FULL_COMMUNICATION (Note: user request is stored but not executed, due to active "wakeup inhibition" for ComM.Ch2, thus ComM.Ch2 stays in COMM_NO_COMMUNICATION)
- ComMUser1 requests COMM_FULL_COMMUNICATION (Note: ComM.Ch1 is requested with COMM_FULL_COMMUNICATION, ComM.Ch2 stays in COMM_NO_COMMUNICATION, due to active "wakeup inhibition")
- ComMUser1 requests COMM_NO_COMMUNICATION (ComM.Ch1 is requested with COMM_NO_COMMUNICATION, ComM.Ch2 stays in COMM_NO_COMMUNICATION, due to active "wakeup inhibition")
- Internal inhibition field is set to 0x00 via call of ComM_SetECUGroupClassification (Note: This disallows to set "wakeup inhibition" and disallows to set "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00669]). ComM.Ch2 stays in COMM_NO_COMMUNICATION, due to active "wakeup inhibition". Changing the internal inhibition field has no impact on activated communication inhibition.)
- The "wake up inhibition" for ComM.Ch2 is requested to be deactivated via a call of ComM_PreventWakeUp. (Note: ComM.Ch2 stays in COMM_NO_COMMUNICATION, since the execution of ComM_PreventWakeUp ("wakeup inhibition") is disallowed, due to the state of internal inhibition field)

7.4.1.4.3 Limit to COMM_NO_COMMUNICATION - example 1

Precondition:

- ECU_A has a ComM with ComM.Ch1 and ComM.Ch2 configured
- ComMUser1 is assigned to ComM.Ch1 and ComM.Ch2
- ComMUser2 is assigned to ComM.Ch2
- All ComM.User request COMM_NO_COMMUNICATION and all ComM channels reside in COMM_NO_COMMUNICATION
- No passive request is available. Thus, no ComM channel is remotely kept awake
- Internal inhibition field is set to 0x02 (Note: This disallows to execute "wakeup inhibition" and allows to execute "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00669])

Runtime:

- ComMUser1 requests COMM_FULL_COMMUNICATION (Note: ComM.Ch1 and ComM.Ch2 are requested with COMM_FULL_COMMUNICATION
- A "limit to COMM_NO_COMMUNICATION" for ComM.Ch2 is activated via a call of ComM_LimitChannelToNoComMode. (Note: ComM.Ch2 transitions to COMM_FULL_COM_READY_SLEEP and then to COMM_NO_COMMUNICATION. ComM.Ch1 is requested to stay in COMM_FULL_COMMUNICATION, due to the ComMUser1 request)
- ComMUser2 requests COMM_FULL_COMMUNICATION (Note: user request is ignored and cleared, due to active "limit to COMM_NO_COMMUNICATION" for ComM.Ch2. Thus, ComM.Ch2 stays in COMM_NO_COMMUNICATION)
- ComMUser1 requests COMM_NO_COMMUNICATION (Note: ComM.Ch1 is requested with COMM_NO_COMMUNICATION, ComM.Ch2 already reside in COMM_NO_COMMUNICATION, due to active "limit to COMM_NO_COMMUNICATION")
- The "limit to COMM_NO_COMMUNICATION" for ComM.Ch2 is deactivated via a call of ComM_LimitChannelToNoComMode. (Note: ComM.Ch2 stays in COMM_NO_COMMUNICATION, since request of ComMUser2 was cleared, due to active "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00215]). Thus, ComMUser2 needs to request ComM.Ch2 again for COMM_FULL_COMMUNICATION)

7.4.1.4.4 Limit to COMM_NO_COMMUNICATION - example 2

Precondition:

- ECU_A has a ComM with ComM.Ch1 and ComM.Ch2 configured
- ComMUser1 is assigned to ComM.Ch1 and ComM.Ch2
- ComMUser2 is assigned to ComM.Ch2
- All ComM.User request COMM_NO_COMMUNICATION and all ComM channels reside in COMM_NO_COMMUNICATION
- No passive request is available. Thus, no ComM channel is remotely kept awake
- Internal inhibition field is set to 0x02 (Note: This disallows to execute "wakeup inhibition" and allows to execute "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00669]))

Runtime:

- ComMUser1 requests COMM_FULL_COMMUNICATION (Note: ComM.Ch1 and ComM.Ch2 are requested with COMM_FULL_COMMUNICATION
- A "limit to COMM_NO_COMMUNICATION" for all ComM channels is activated via a call of ComM_LimitECUToNoComMode. (Note: all ComM channels transition to COMM_FULL_COM_READY_SLEEP and then to COMM_NO_COMMUNICATION)

- ComMUser2 requests COMM_FULL_COMMUNICATION (Note: user request is ignored and cleared, due to active "limit to COMM_NO_COMMUNICATION" for all ComM channels)
- The "limit to COMM_NO_COMMUNICATION" for all ComM channels is deactivated via a call of ComM_LimitECUToNoComMode. (Note: all ComM channels stay in COMM_NO_COMMUNICATION, since requests of ComMUser1 and ComMUser2 were cleared, due to active "limit to COMM_NO_COMMUNICATION" (see [SWS_ComM_00215]). Thus, ComMUser1 and ComUser2 needs to be requested again with COMM_FULL_COMMUNICATION).

7.5 Bus communication management

[SWS_ComM_00402]

Upstream requirements: SRS_ModeMgm_00049, SRS_ModeMgm_09083

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

[SWS_ComM_00664]

Upstream requirements: SRS_ModeMgm_09168

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

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

7.6 Network management dependencies

[SWS_ComM_00599]

Upstream requirements: SRS_ModeMgm_09132

⌈The ComM module shall support the shutdown synchronization variants (configured with ComMNmVariant, see [ECUC_ComM_00568]) LIGHT, SLAVE_ACTIVE, SLAVE_PASSIVE, PASSIVE and FULL.⌋

Comment on [SWS_ComM_00599]: 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 E_OK, but network will not be released until all coordinated networks are ready to go to sleep.

Note on [SWS_ComM_00599]: For more details refer to Table 7.3.

NM variant	Keep bus awake capability	Shutdown synchronization
NONE		No shutdown synchronization by ComM. Shutdown by switching off the power of the ECU.
SLAVE_ACTIVE	No (but the corresponding master could trigger a wake-up based on a slave request for a wake-up. E.g. the LIN State Manager of a LIN master restarts wake-up repetition)	Synchronized by its master (e.g. LIN master)
SLAVE_PASSIVE	No (the slave will always follow the communication request of the corresponding master. The slave has no possibility to request a wake-up on the corresponding communication channel.	Synchronized by its master (e.g. ComM channel with ComMBusType set to COMM_BUS_TYPE_ETH and used Ethernet hardware is compliant to OA TC10 (see [2]))
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 7.3: 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_00602]

Upstream requirements: SRS_ModeMgm_09132

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

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

[SWS_ComM_00667]

Upstream requirements: SRS_ModeMgm_09132

⌈The ComM module shall omit to call Nm_NetworkRequest() from NM if configuration parameter ComMNmVariant= LIGHT|SLAVE_ACTIVE|SLAVE_PASSIVE|NONE (see [ECUC_ComM_00568]).⌋

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

7.7 Bus error management

7.7.1 Network Start Indication

[SWS_ComM_00583]

Upstream requirements: SRS_ModeMgm_09132

⌈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".

7.8 Test support requirements

7.8.1 Inhibited Full Communication Request Counter

[SWS_ComM_00138]

Upstream requirements: SRS_ModeMgm_09155

⌈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.⌋

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

[SWS_ComM_00140] Storage of Inhibit counter for rejected mode requests

Upstream requirements: SRS_ModeMgm_09155

⌈If ComMGlobalNvMBlockDescriptor is configured, then the Inhibit counter ([SWS_ComM_00138]) for all rejected COMM_FULL_COMMUNICATION mode requests shall be stored in non-volatile memory.⌋

[SWS_ComM_00141]

Upstream requirements: SRS_ModeMgm_09155

⌈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]

Upstream requirements: SRS_ModeMgm_09155

⌈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]

Upstream requirements: SRS_ModeMgm_09156

⌈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.

7.9 Error Classification

Chapter [5, General Specification of Basic Software Modules] 7.2 *"Error Handling"* describes the error handling of the Basic Software in detail. Above all, it constitutes a classification scheme consisting of five error types which may occur in BSW modules.

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

7.9.1 Development Errors

[SWS_ComM_00234] Definition of development errors in module ComM

Upstream requirements: SRS_BSW_00323, SRS_BSW_00327, SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

⌈

Type of error	*Related error code*	*Error value*
API service used without module initialization	COMM_E_UNINIT	0x1
API service used with wrong parameters	COMM_E_WRONG_PARAMETERS	0x2
API Service used with a null pointer	COMM_E_PARAM_POINTER	0x3
Initialization failed	COMM_E_INIT_FAILED	0x4

⌋

[SWS_ComM_00612]

Upstream requirements: SRS_BSW_00406

⌈If ComM is not initialized, all ComM module and all API service other than ComM_Init() (see [SWS_ComM_00146]) 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.

⌋

[SWS_ComM_00858]

Upstream requirements: SRS_BSW_00406

⌈If development error detection is enabled by ComMDevErrorDetect (see [ECUC_ComM_00555]): the function shall check that the service ComM_Init was previously called. If the check fails, the function shall raise the development error COMM_E_UNINIT⌋

7.9.2 Runtime Errors

[SWS_ComM_91110] Definition of runtime errors in module ComM ⌈

Type of error	*Related error code*	*Error value*
Reading of data from NVRAM failed	COMM_E_READ_NV_FAILED	0x01

⌋

7.9.3 Production Errors

There are no production errors.

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 7.9:

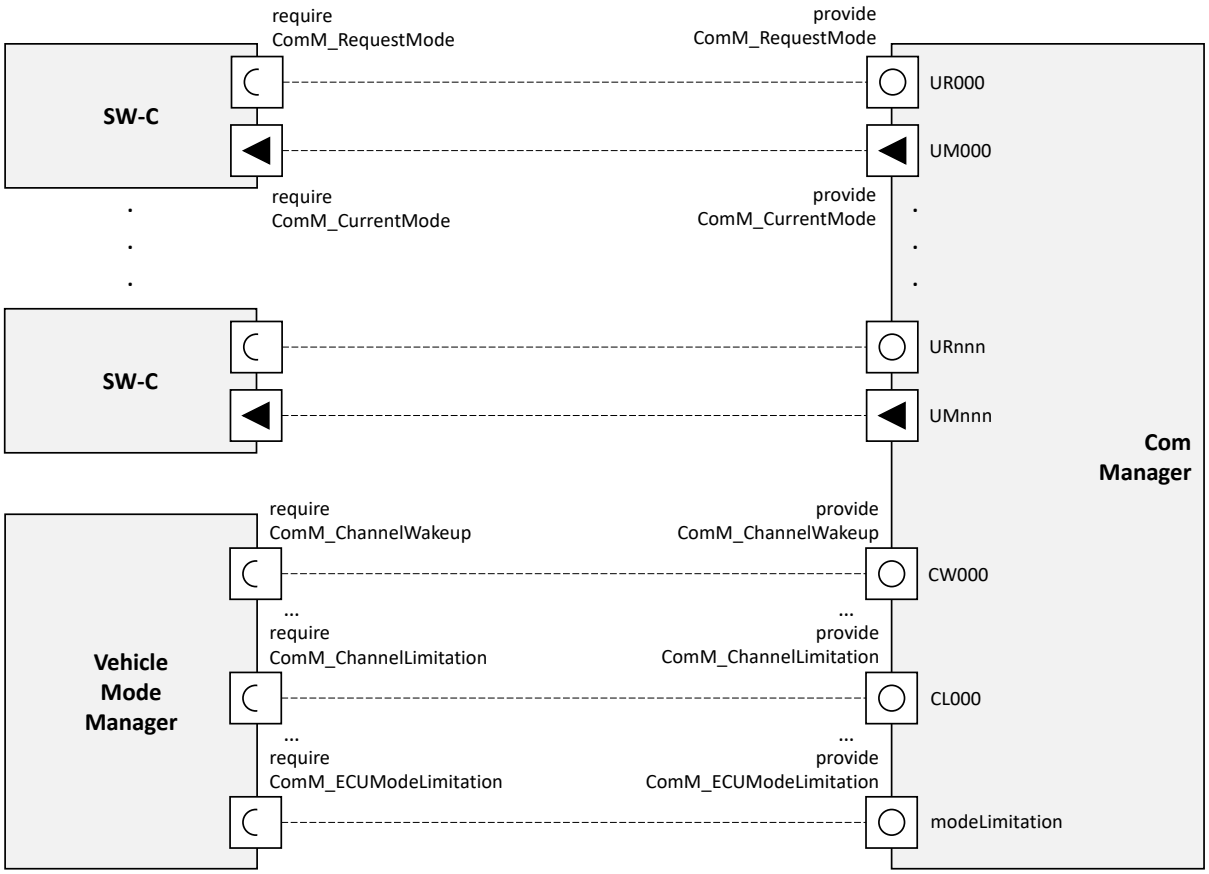


Figure 7.9: ARPackage of the Communication Manager Module

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.

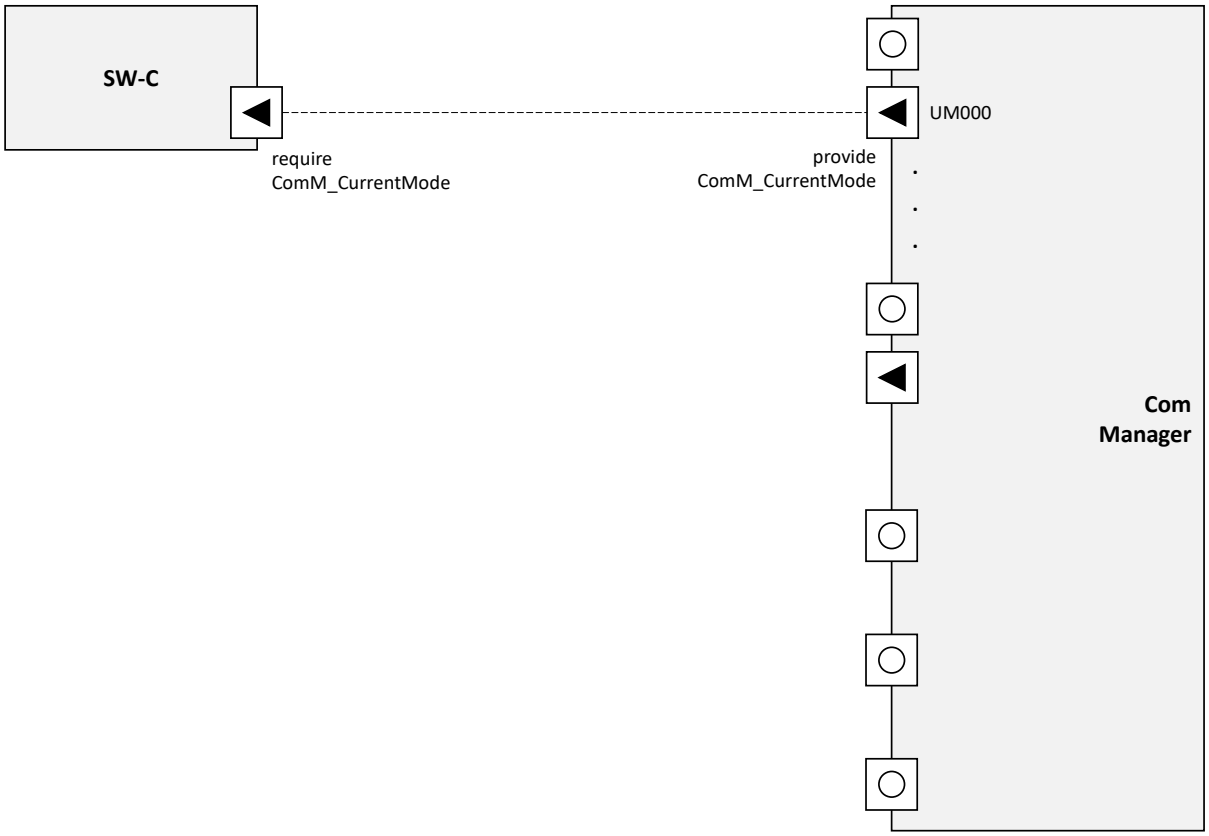


Figure 7.10: SW-C requests state changes to the Communication Manager Module

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

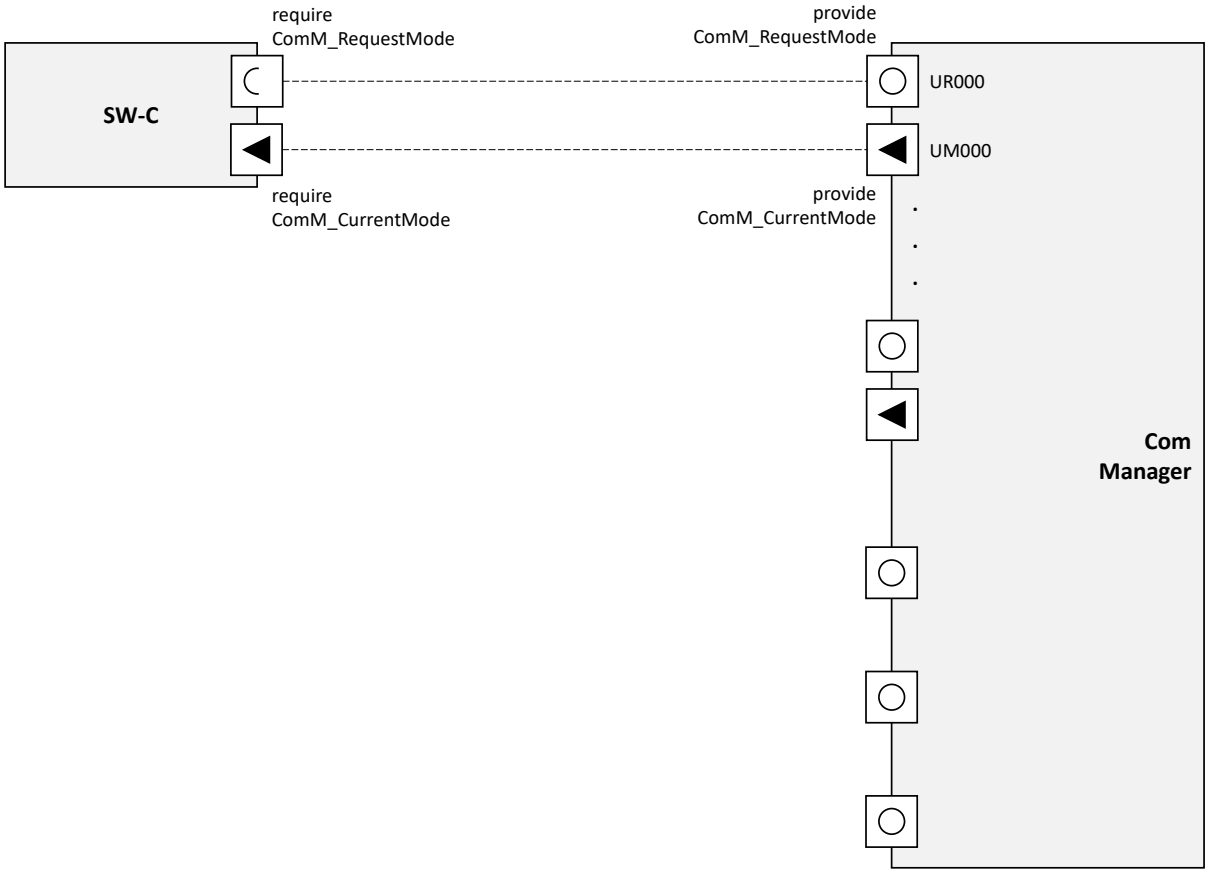


Figure 7.11: SW-C requires state changes within the Communication Manager Module and reads out current communication state

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]

Upstream requirements: SRS_ModeMgm_09078

⌈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:

```
1 MySW-C_Runnable_Init(self)
2 {
3       // SW-C wants to send and receive data
4       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.

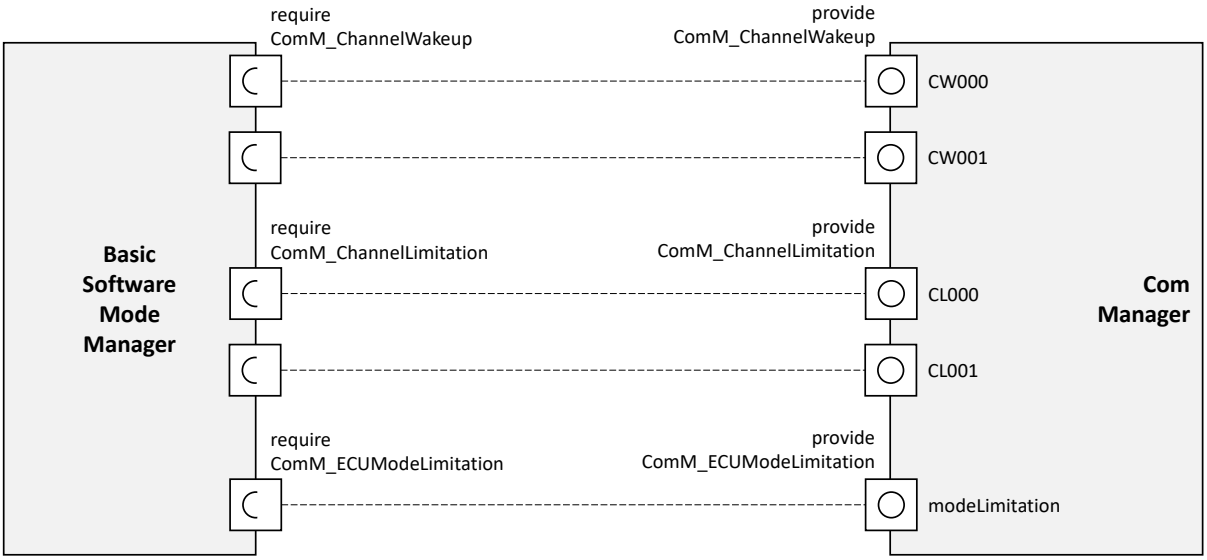


Figure 7.12: Interaction between BswM and the ComM module

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 8.6.4 “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 8.6.2.4 “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]

Upstream requirements: SRS_ModeMgm_09085

⌈The Communication Manager Module shall have an AUTOSAR port providing the ModeSwitchInterface interface ’ComM_CurrentMode’.⌋

[SWS_ComM_00733]

Upstream requirements: SRS_ModeMgm_09085

⌈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 8.6.3.1 “ComM_CurrentMode”.

7.10.3.4 Ports and Port Interfaces for the ComM users currently requesting `COMM_FULL_COMMUNICATION`

7.10.3.4.1 General approach

[SWS_ComM_00734]

Upstream requirements: SRS_ModeMgm_09084

⌈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 COMM_FULL_COMMUNICATION requires the sender-receiver interface ComM_CurrentChannelRequest'.

[SWS_ComM_00736]

Upstream requirements: SRS_ModeMgm_09078

⌈Whenever the set of ComM users currently requesting COMM_FULL_COMMUNICATION 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.⌋

Note: Requests which are accepted but not processed because of active ModeLimitations will lead to an update of the data element.

7.10.3.4.2 Data Types

See 8.6.4.4 "ComM_UserHandleArrayType".

7.10.3.4.3 Port Interface ComM_CurrentChannelRequest

See 8.6.1.1 "ComM_CurrentChannelRequest".

7.10.3.5 Ports and Port Interface for ECU Mode Limitation

7.10.3.5.1 General approach

[SWS_ComM_00740]

Upstream requirements: SRS_ModeMgm_09071

⌈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 8.6.2.3 "ComM_ECUModeLimitation".

7.10.3.6 Ports and Port Interface for Channel Wake up

7.10.3.6.1 General approach

[SWS_ComM_00747]

Upstream requirements: SRS_ModeMgm_09089

⌈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 8.6.2.2 "ComM_ChannelWakeup".

7.10.3.7 Ports and Port Interface for interface Channel Limitation

7.10.3.7.1 General approach

[SWS_ComM_00752]

Upstream requirements: SRS_ModeMgm_09071

⌈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 8.6.2.1 “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]

Upstream requirements: SRS_ModeMgm_09078, SRS_ModeMgm_09080, SRS_ModeMgm_09084, SRS_ModeMgm_09172, SRS_ModeMgm_09149, SRS_ModeMgm_09168, SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈

```
/* 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.
    // 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]

Upstream requirements: SRS_ModeMgm_09078, SRS_ModeMgm_09080, SRS_ModeMgm_09084, SRS_ModeMgm_09172, SRS_ModeMgm_09149, SRS_ModeMgm_09168, SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈

```
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
```

```
25      RunnableEntity RequestComMode
26          symbol "ComM_RequestComMode" /* see SWS_ComM_00110 */
27          canbeInvokedConcurrently = TRUE
28      RunnableEntity GetMaxComMode
29          symbol "ComM_GetMaxComMode" /* see SWS_ComM_00085 */
30          canbeInvokedConcurrently = TRUE
31      RunnableEntity GetRequestedComMode
32          symbol "ComM_GetRequestedComMode"
33          canbeInvokedConcurrently = TRUE
34      RunnableEntity GetCurrentComMode
35          symbol "ComM_GetCurrentComMode" /*see SWS_ComM_00083 */
36          canbeInvokedConcurrently = TRUE
37      // the following applies if ComMModeLimitationEnabled
38      // (see ECUC_ComM_00560)
39      modeLimitation.LimitECUToNoComMode -> LimitECUToNoComMode
40      modeLimitation.ReadInhibitCounter -> ReadInhibitCounter
41      modeLimitation.ResetInhibitCounter -> ResetInhibitCounter
42      modeLimitation.SetECUGroupClassification ->
            SetECUGroupClassification
43      // per-channel behaviour only present
44      // if ComMModeLimitationEnabled (see ECUC_ComM_00560)
45      // there are NC channels
46      // To facilitate configuration, the names of the channels
            correspond
47      // to the index of the channel in the "Channel" container used to
48      // configure the Communication Manager Module
49      CL000.LimitChannelToNoComMode -> LimitChannelToNoComMode
50      CL000.GetInhibitionStatus -> GetInhibitionStatus
51      PortArgument {port=CL000,
52                    value.type=NetworkHandleType,
53                    value.value=Channel[0].COMM_CHANNEL_ID}
54      ...
55  CLnnn.LimitChannelToNoComMode -> LimitChannelToNoComMode
56  CLnnn.GetInhibitionStatus -> GetInhibitionStatus
57      PortArgument {port=CLnnn,
58                    value.type=NetworkHandleType,
59                    value.value=Channel[nnn].COMM_CHANNEL_ID}
60      // per-channel behaviour only present
61      // if COMM_WAKEUP_INHIBITION_ENABLED (see ECUC_ComM_00559)
62      CW000.preventWakeUp -> PreventWakeUp
63      PortArgument {port=CW000,
64                    value.type=NetworkHandleType,
65                    value.value=Channel[0].COMM_CHANNEL_ID}
66      ...
67      CWnnn.preventWakeUp -> PreventWakeUp
68      PortArgument {port=CWnnn,
69                    value.type=NetworkHandleType,
70                    value.value=Channel[nnn].COMM_CHANNEL_ID}
71      // per-user behaviour
72      // Note that the port-argument value must be consistent with the
73      // value in the configuration COMM_USER_LIST
74      // Note that the exact data-type of the UserHandleType must of
            course
75      // be defined BEFORE RTE_configuration, but does NOT affect the
76      // API seen by the SW-Cs that use the service
77      UR000.RequestComMode -> RequestComMode
```

```
78      UR000.GetMaxComMode -> GetMaxComMode
79      UR000.GetRequestedComMode -> GetRequestedComMode
80      UR000.GetCurrentComMode -> GetCurrentComMode
81      PortArgument {port=UR000,
82                    value.type= ComM_UserhandleType,
83                    value.value=COMM_USER_LIST[0]}
84      ...
85      URnnn.RequestComMode -> RequestComMode
86      URnnn.GetMaxComMode -> GetMaxComMode
87      URnnn.GetRequestedComMode -> GetRequestedComMode
88      URnnn.GetCurrentComMode -> GetCurrentComMode
89      PortArgument {port=URnnn,
90                    value.type= ComM_UserhandleType,
91                    value.value=COMM_USER_LIST[n]}
92  };
```

⌋

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.

7.11 Multicore Distribution

In its role as central module dealing with different network types the ComM interaction spans across partitions in case the Com-Stack is distributed and so shall provide required multi-core features to ensure a clean architecture and keep the network dependent clusters free of multi-partition (multi-core) add-ons.

Note: Even though the basic software is distributed across several partitions, ComM and Nm Masters should reside in the same partition in order to keep mode interfaces between the two modules simple (for further information see chapter Master/Satellite-approach in [12] (Guide to BSW Distribution)).

7.12 Non functional requirements

[SWS_ComM_00459]

Upstream requirements: SRS_BSW_00342

⌈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.

⌋

7.13 Security Events

The module does not report security events.

8 API specification

8.1 Imported types

8.1.1 Standard types

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

[SWS_ComM_00820] Definition of imported datatypes of module ComM

Upstream requirements: SRS_BSW_00348, SRS_BSW_00357

⌈

Module	*Header File*	*Imported Type*
Comtype	ComStack_Types.h	NetworkHandleType
	ComStack_Types.h	PNCHandleType
NvM	Rte_NvM_Type.h	NvM_BlockIdType
	Rte_NvM_Type.h	NvM_RequestResultType
Std	Std_Types.h	Std_ReturnType

⌋

The ComM API uses the following extension to Std_ReturnType:

[SWS_ComM_91027] Definition of Std_ReturnType-extension for module ComM

Upstream requirements: SRS_BSW_00331, SRS_BSW_00369, SRS_BSW_00377, SRS_BSW_00441

⌈

Range	COMM_E_MODE_LIMITATION	2	Function call has been successful but mode can not be granted because of mode inhibition.
	COMM_E_MULTIPLE_PNC_ASSIGNED	3	Function could not provide the current mode of the PNC, since multiple PNCs are assigned to the affected user
	COMM_E_NO_PNC_ASSIGNED	4	Function could not provide the current mode of the PNC, since no PNC is assigned to the affected user
	COMM_E_LEARNING_ACTIVE	5	Function call has been successfully, but functionality cannot be executed because PNC learning phase is active.
Description	–		
Available via	ComM.h		

⌋

8.2 Type definitions

[SWS_ComM_00863]

Upstream requirements: SRS_BSW_00441

⌈The following Data Types shall be used for the functions defined in this Specification.⌋

8.2.1 ComM_InitStatusType

[SWS_ComM_00668] Definition of datatype ComM_InitStatusType

Upstream requirements: SRS_BSW_00101, SRS_BSW_00406

⌈

Name	ComM_InitStatusType		
Kind	Enumeration		
Range	COMM_UNINIT	0x00	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	0x01	The COM Manager is initialized and usable.
Description	Initialization status of ComM.		
Available via	ComM.h		

⌋

8.2.2 ComM_PncModeType

[SWS_ComM_00673] Definition of datatype ComM_PncModeType

Upstream requirements: SRS_ModeMgm_09247

⌈

Name	ComM_PncModeType		
Kind	Enumeration		
Range	COMM_PNC_REQUESTED	0x00	PNC is requested by a local ComM user
	COMM_PNC_READY_SLEEP	0x01	PNC is requested by a remote ComM user
	COMM_PNC_PREPARE_SLEEP	0x02	PNC is active with no deadline monitoring
	COMM_PNC_NO_COMMUNICATION	0x03	PNC does not communicate
	COMM_PNC_REQUESTED_WITH_WAKEUP_REQUEST	0x04	PNC is requested by a local ComM user. The mode is used to indicate the BswM, that an active PNC request should trigger also a wake-up of the used communication hardware, if this is supported and configured (e.g. used for Ethernet switch port switching in combination with OA TC10 compliant Ethernet hardware).

▽

△

Description	Current mode of a PNC
Available via	ComM.h

⌋

8.2.3 ComM_StateType

[SWS_ComM_00674] Definition of datatype ComM_StateType ⌈

Name	ComM_StateType		
Kind	Type		
Derived from	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'		
Available via	ComM.h		

⌋

8.2.4 ComM_ConfigType

[SWS_ComM_00162] Definition of datatype ComM_ConfigType

Upstream requirements: SRS_BSW_00404, SRS_BSW_00405

⌈

Name	ComM_ConfigType	
Kind	Structure	
Elements	implementation specific	
	Type	–
	Comment	The contents of the initialization data structure are implementation specific
Description	This type contains the implementation-specific post build configuration structure.	
Available via	ComM.h	

⌋

8.3 Function definitions

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

Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:

- ComM_Init

[SWS_ComM_00620]

Upstream requirements: SRS_BSW_00345

⌈All the provided indication functions shall be implemented pre-compile time.⌋

8.3.1 ComM_Init

[SWS_ComM_00146] Definition of API function ComM_Init

Upstream requirements: SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414

⌈

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.	
Available via	ComM.h	

⌋

[SWS_ComM_00793]

Upstream requirements: SRS_BSW_00101

⌈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] Read non-volatile parameters in `ComM_Init`

Upstream requirements: SRS_BSW_00101, SRS_BSW_00497

⌈In `ComM_Init` ComM shall read non-volatile parameters specified in [SWS_ComM_00103] with the values read from non-volatile memory (NVRAM). If no parameters available (e.g. reading of NvM Block fails), ComM shall use the default values for the initialization.⌋

Note for [SWS_ComM_00864]: Default values for ComMEcuGroupClassification and inhibition status are used according to configuration parameters referenced in [SWS_ComM_00103].

[SWS_ComM_01098]

Upstream requirements: SRS_BSW_00101

⌈Default value used for initialization of the inhibition counter shall be 0.⌋

[SWS_ComM_01099]

Upstream requirements: SRS_BSW_00101

⌈Default value used for initialization of the PNC-to-channel Mapping shall be a two-dimensional array with the statically configured mapping of PNC to channels of the PNC Gateway.⌋

[SWS_ComM_01100]

Upstream requirements: SRS_BSW_00101

⌈Default value used for initialization of the PNC Membership shall be a PNC Request Bit Vector that holds the statically configured PNCs for the node.⌋

[SWS_ComM_01101]

Upstream requirements: SRS_BSW_00101, SRS_BSW_00497

⌈If reading the data from non-volatile memory (NVRAM) fails, ComM shall report the runtime error COMM_E_READ_NV_FAILED to DET.⌋

Note to [SWS_ComM_01101]: If ComM reports COMM_E_READ_NV_FAILED, an application could react on this error scenario, e.g. update PNC Membership by calling ComM_UpdatePncMembership or update PNC-to-channel Mapping by calling ComM_UpdatePncToChannelMapping.

8.3.2 ComM_DeInit

[SWS_ComM_00147] Definition of API function ComM_DeInit

Upstream requirements: SRS_BSW_00336

⌈

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.
Available via	ComM.h

⌋

[SWS_ComM_00794]

Upstream requirements: SRS_ModeMgm_09083

⌈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.

[SWS_ComM_00865]

Upstream requirements: SRS_BSW_00406

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

8.3.3 ComM_GetStatus

[SWS_ComM_00242] Definition of API function ComM_GetStatus

Upstream requirements: SRS_BSW_00406

⌈

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 Com M_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.
Available via	ComM.h

⌋

8.3.4 ComM_GetInhibitionStatus

[SWS_ComM_00619] Definition of API function ComM_GetInhibitionStatus ⌈

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.	
Available via	ComM.h	

⌋

8.3.5 ComM_RequestComMode

[SWS_ComM_00110] Definition of API function ComM_RequestComMode

Upstream requirements: SRS_ModeMgm_09081, SRS_ModeMgm_09078

⌈

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: The following modes are invalid user requests, since they are used as internal modes: • COMM_SILENT_COMMUNICATION (this mode is used for synchronization at shutdown) • COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST (this mode is used internally within the ComM channel statemachine to trigger the lower layers to request a wakeup on the network if the used hardware support such a feature. (e.g. Ethernet hardware which is compatible with OA TC10). The following modes are valid user requests: • COMM_NO_COMMUNICATION • COMM_FULL_COMMUNICATION. The communication request could also be released due to a ComM communication inhibition	
Available via	ComM.h	

⌋

[SWS_ComM_00795]

Upstream requirements: SRS_ModeMgm_09090

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

8.3.6 ComM_GetMaxComMode

[SWS_ComM_00085] Definition of API function ComM_GetMaxComMode

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157, SRS_ModeMgm_09081, SRS_ModeMgm_09078

⌈

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.	
Available via	ComM.h	

⌋

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]

Upstream requirements: SRS_ModeMgm_09149

⌈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]

Upstream requirements: SRS_ModeMgm_09090

⌈Configuration of ComM_GetMaxComMode: Relationship between users and channels. A user is statically mapped to one or more channels.⌋

8.3.7 ComM_GetRequestedComMode

[SWS_ComM_00079] Definition of API function ComM_GetRequestedComMode

Upstream requirements: SRS_ModeMgm_09149, SRS_ModeMgm_09081, SRS_ModeMgm_09078

⌈

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.	

▽

△

Available via	ComM.h

⌋

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.

Note: A user is statically mapped to one or more channels. The relationship between users and channels is reflected by the configuration (see [ECUC_ComM_00658]).

8.3.8 ComM_GetCurrentComMode

[SWS_ComM_00083] Definition of API function ComM_GetCurrentComMode

Upstream requirements: SRS_ModeMgm_09084, SRS_ModeMgm_09081, SRS_ModeMgm_09078

⌈

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: <Bus>SM _GetCurrentComMode(...))	
Available via	ComM.h	

⌋

[SWS_ComM_00176]

Upstream requirements: SRS_ModeMgm_09172

⌈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]).⌋

[SWS_ComM_00798]

Upstream requirements: SRS_ModeMgm_09090

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

8.3.9 ComM_GetCurrentPNCComMode

[SWS_ComM_91002] Definition of API function ComM_GetCurrentPNCComMode

Upstream requirements: SRS_ModeMgm_09084, SRS_ModeMgm_09081, SRS_ModeMgm_09078

⌈

Service Name	ComM_GetCurrentPNCComMode	
Syntax	`Std_ReturnType ComM_GetCurrentPNCComMode ( ComM_UserHandleType User, ComM_ModeType* ComMode )`	
Service ID [hex]	0x6a	
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 the state of the PNC referenced by the given ComMUser E_NOT_OK: Return of the PNC state referenced by the given Com MUser failed COMM_E_MULTIPLE_PNC_ASSIGNED: Function could not provide the current mode of the PNC, since multiple PNCs are assigned to the affected user COMM_E_NO_PNC_ASSIGNED: Function could not provide the current mode of the PNC, since no PNC is assigned to the affected user
Description	The function returns the current Communication Mode of the corresponding PNC the affected user is assigned to.	
Available via	ComM.h	

⌋

[SWS_ComM_01022]

Upstream requirements: SRS_ModeMgm_09149

⌈If more than one PNC is assigned to the affected user, the function ComM_GetCurrentPNCComMode shall return COMM_E_MULTIPLE_PNC_ASSIGNED as ComMode .⌋

Comment to [SWS_ComM_01022]: For multiple PNCs it is not possible to return a consistent communication mode since the PNCs could have different communication modes.

[SWS_ComM_01023]

Upstream requirements: SRS_ModeMgm_09149

⌈If no PNC is assigned to the affected user, the function ComM_GetCurrentPNCComMode shall return COMM_E_NO_PNC_ASSIGNED as ComMode.⌋

[SWS_ComM_01024]

Upstream requirements: SRS_ModeMgm_09149

⌈If [SWS_ComM_01022] and [SWS_ComM_01023] do not apply, the function shall query for the current communication mode of the corresponding PNC statemachine the user is assigned to. If the corresponding PNC statemachine is in main state COMM_PNC_FULL_COMMUNICATION, then the function shall return COMM_FULL_COMMUNICATION as ComMode. If the corresponding PNC statemachine is main state COMM_PNC_NO_COMMUNCATION, then the function shall return COMM_NO_COMMUNICATION as ComMode.⌋

Note: The service interface ComM_UserRequest provides the possibility among others to query for the current mode of a channel and to query for the current mode of a PNC. Since the service interface has ComM_ModeType as a return value type, the main state of the ComM PNC statemachine has to be mapped to the main state of the ComM channel statemachine

[SWS_ComM_01025]

Upstream requirements: SRS_ModeMgm_09090, SRS_ModeMgm_09246

⌈Configuration of ComM_GetCurrentPNCComMode: Relationship between users and PNCs. A user is statically mapped to one or more PNCs.⌋

8.3.10 ComM_GetPncToChannelMapping

[SWS_ComM_91013] Definition of API function ComM_GetPncToChannelMapping

Upstream requirements: SRS_ModeMgm_09259

⌈

Service Name	ComM_GetPncToChannelMapping

Syntax	`Std_ReturnType ComM_GetPncToChannelMapping (`
	`  uint8* MappingTable,`
	`  uint8* ChannelCnt,`
	`  uint8* PNCRequestBitVectorLength`
	`)`

Service ID [hex]	0x68
Sync/Async	Synchronous
Reentrancy	Non Reentrant

▽

△

Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	MappingTable	Pointer to an array of uint8 which stores for each channel a PNC Request Bit Vector representing the mapping of PNCs to the channel
	ChannelCnt	Pointer to the number of physical channels passed in the MappingTable
	PNCRequestBitVector Length	Pointer to the length in bytes of the PNC Request Bit Vector where each bit represents one PNC
Return value	Std_ReturnType	E_OK: Successfully get PNC-to-channel-mapping entry E_NOT_OK: Getting of PNC-to-channel-mapping entry failed COMM_E_LEARNING_ACTIVE: Functionality cannot be executed because PNC learning phase is active.
Description	This function returns the current configuration of the ECUs PNC-to-channel-mapping.	
Available via	ComM.h	

⌋

[SWS_ComM_01034]

Upstream requirements: SRS_ModeMgm_09258

⌈Function ComM_GetPncToChannelMapping shall be only available if ComMPncGatewayEnabled and ComMDynamicPncToChannelMappingSupport are set to TRUE.⌋

[SWS_ComM_01035]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when PNC learning phase is active, then the function ComM_GetPncToChannelMapping shall return with COMM_E_LEARNING_ACTIVE.⌋

[SWS_ComM_01036]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when PNC learning phase is not active, then the function ComM_GetPncToChannelMapping shall provide within MappingTable the current PNC-to-channel mapping as a two-dimensional array where on first dimension all ComM channels where ComMPncGatewayType is set are handled according to their derived order in ComM and on second dimension all configured ComMPnc according to their order given by their ComMPncId. ComM shall also set the parameter ChannelCnt and PNCRequestBitVectorLength accordingly and return with E_OK.⌋

Note: The content of this MappingTable can only be interpreted correctly by application or tester correctly if the number of Channels and PNCs and their order is known.

8.3.11 ComM_UpdatePncToChannelMapping

[SWS_ComM_91015] Definition of API function ComM_UpdatePncToChannelMapping

Upstream requirements: SRS_ModeMgm_09259

⌈

Service Name	ComM_UpdatePncToChannelMapping	
Syntax	`Std_ReturnType ComM_UpdatePncToChannelMapping ( const uint8* MappingTable, uint8 channelCnt, uint8 PNCRequestBitVectorLength )`	
Service ID [hex]	0x62	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	MappingTable	Pointer to an array of uint8 which stores for each channel a PNC Request Bit Vector representing the mapping of PNCs to the channel
	channelCnt	Number of physical channels passed in the MappingTable
	PNCRequestBitVector Length	Length in bytes of the PNC Request Bit Vector where each bit represents one PNC
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Successfully set PNC-to-channel-mapping entry E_NOT_OK: Set of PNC-to-channel-mapping entry failed COMM_E_LEARNING_ACTIVE: Functionality cannot be executed because PNC learning phase is active.
Description	This function can be used to set entries within the ECUs PNC-to-channel-mapping	
Available via	ComM.h	

⌋

[SWS_ComM_01037]

Upstream requirements: SRS_ModeMgm_09258

⌈Function ComM_UpdatePncToChannelMapping shall be only available if ComMPnc-GatewayEnabled and ComMDynamicPncToChannelMappingSupport are set to TRUE.⌋

[SWS_ComM_01038]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and the function ComM_UpdatePncToChannelMapping is called, ComM shall check if ChannelCnt matches the number of ComM channels where ComMPnc-GatewayType is set and PNCRequestBitVectorLength matches the length PNC Request Bit Vector. If one parameter does not match and ComMDevErrorDetect is set to TRUE, then ComM shall call Det_ReportError with COMM_E_WRONG_PARAMETERS. If one parameter does not match, then ComM shall return with E_NOT_OK.⌋

[SWS_ComM_01039]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel, when passed parameters match (see [SWS_ComM_01038]) and when PNC learning phase is active, then the function ComM_UpdatePncToChannelMapping shall return with COMM_E_LEARNING_ACTIVE.⌋

[SWS_ComM_01040]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel, when passed parameters match (see [SWS_ComM_01038]) and PNC learning phase is not active, then the function ComM_UpdatePncToChannelMapping shall merge for all PNCs the provided information with their current PNC-to-channel mappings whereby MappingTable shall be interpreted as a two-dimensional array with on first dimension all ComM channels where ComMPncGatewayType is set are handled according to their derived order in ComM and on second dimension all configured ComMPnc according to their order given by their ComMPncId. Additionally it shall return with E_OK.⌋

8.3.12 ComM_ResetPncToChannelMapping

[SWS_ComM_91017] Definition of API function ComM_ResetPncToChannelMapping

Upstream requirements: SRS_ModeMgm_09259

⌈

Service Name	ComM_ResetPncToChannelMapping	
Syntax	`Std_ReturnType ComM_ResetPncToChannelMapping ( void )`	
Service ID [hex]	0x63	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Successfully reset PNC-to-channel-mapping to default E_NOT_OK: Reset of PNC-to-channel-mapping to default failed COMM_E_LEARNING_ACTIVE: Functionality cannot be executed because PNC learning phase is active.
Description	This function resets dynamic entries within the ECUs PNC-to-channel-mapping to default values	
Available via	ComM.h	

⌋

[SWS_ComM_01041]

Upstream requirements: SRS_ModeMgm_09258

⌈Function ComM_ResetPncToChannelMapping shall be only available if ComMPncGatewayEnabled and ComMDynamicPncToChannelMappingSupport are set to TRUE.⌋

[SWS_ComM_01042]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when PNC learning phase is active, then the function ComM_ResetPncToChannelMapping shall return with COMM_E_LEARNING_ACTIVE.⌋

[SWS_ComM_01043]

Upstream requirements: SRS_ModeMgm_09259

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when PNC learning phase is not active, then the function ComM_ResetPncToChannelMapping shall set the PNC-to-channel mappings to the default values from the original configuration (i.e. static entries) and return with E_OK.⌋

8.3.13 ComM_PnLearningRequest

[SWS_ComM_91019] Definition of API function ComM_PnLearningRequest

Upstream requirements: SRS_ModeMgm_09260

⌈

Service Name	ComM_PnLearningRequest	
Syntax	`Std_ReturnType ComM_PnLearningRequest ( void )`	
Service ID [hex]	0x64	
Sync/Async	Asynchronous	
Reentrancy	Non Reentrant	
Parameters (in)	None	
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: Successfully started PNC Learning algorithm E_NOT_OK: PNC Learning algorithm could not be started COMM_E_LEARNING_ACTIVE: Functionality cannot be executed because PNC learning phase is active.
Description	Triggers the NM to return into NM Repeat Message State and send the Partial Network Learning Bit (in order for receiving nodes to respond) together with the Repeat Message Request Bit (in order for receiving nodes to return into NM Repeat Message State). This function is used for the optional Dynamic PNC-to-channel-mapping feature.	
Available via	ComM.h	

⌋

[SWS_ComM_01044]

Upstream requirements: SRS_ModeMgm_09258

⌈Function ComM_PnLearningRequest shall be only available if ComMDynamicPncToChannelMappingSupport is set to TRUE.⌋

[SWS_ComM_01045] Handling for a call of ComM_PnLearningRequest and PNC learning phase is active

Upstream requirements: SRS_ModeMgm_09260

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when PNC learning phase is active, then the function ComM_PnLearningRequest shall return with COMM_E_LEARNING_ACTIVE.⌋

[SWS_ComM_01058]

Upstream requirements: SRS_ModeMgm_09260

⌈If ComM_PnLearningRequest is called, PNC learning phase is inactive and at least one ComMChannel resides in another state than COMM_FULL_COMMUNICATION, then the function ComM_PnLearningRequest shall return with E_NOT_OK.⌋

Note: When ComM_PnLearningRequest is called, all relevant communication channels need to be already in COMM_FULL_COMMUNICATION state. This could be achieved by requesting an active diagnostic session via call of ComM_DCM_ActiveDiagnostic(). The learning phase may be triggered by a diagnostic tester.

[SWS_ComM_01046] Handling for a call of ComM_PnLearningRequest and PNC learning phase is note active

Upstream requirements: SRS_ModeMgm_09260

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when the PNC learning phase is not active, then the function ComM_PnLearningRequest shall call the API Nm_PnLearningRequest on all channels where ComMDynamicPncToChannelMappingEnabled is set to TRUE and return with E_OK.⌋

8.3.14 ComM_UpdatePncMembership

[SWS_ComM_91021] Definition of API function ComM_UpdatePncMembership

Upstream requirements: SRS_ModeMgm_09263

⌈

Service Name	ComM_UpdatePncMembership	
Syntax	`Std_ReturnType ComM_UpdatePncMembership ( boolean Control, const uint8* PncMembership )`	
Service ID [hex]	0x65	
Sync/Async	Synchronous	
Reentrancy	Non Reentrant	
Parameters (in)	Control	Boolean Parameter: 0 = Unset the corresponding Bits in PncBit Mask 1 = Set the corresponding Bits in PncBitMask
	PncMembership	Array of uint8 with <PNC Request Bit Vector length> Elements that holds the current PNC Membership of the node
Parameters (inout)	None	
Parameters (out)	None	
Return value	Std_ReturnType	E_OK: ComM_PncMembership successfully updated E_NOT_OK: Error occurred while updating the PNC membership. COMM_E_LEARNING_ACTIVE: Functionality cannot be executed because PNC learning phase is active.
Description	This function is used by SWCs to update the PNC membership which is transmitted during PNC Learning. This function is used for the optional Dynamic PNC-to-channel-mapping feature.	
Available via	ComM.h	

⌋

[SWS_ComM_01047]

Upstream requirements: SRS_ModeMgm_09258

⌈Function ComM_UpdatePncMembership shall be only available if ComMDynamicPncToChannelMappingSupport is set to TRUE.⌋

[SWS_ComM_01048] Handling for a call of ComM_UpdatePncMembership and PNC learning phase is active

Upstream requirements: SRS_ModeMgm_09260

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and when PNC learning phase is active, then the function ComM_UpdatePncMembership shall return with COMM_E_LEARNING_ACTIVE.⌋

[SWS_ComM_01049] Handling for a call of ComM_UpdatePncMembership and PNC learning phase is not active

Upstream requirements: SRS_ModeMgm_09260

⌈If ComMDynamicPncToChannelMappingEnabled is set to TRUE on at least one channel and PNC Learning phase is not active, then the function ComM_UpdatePncMembership shall perform the following actions:

- When Control = 0, then the current PNC membership shall be applied with logical AND (conjunction) operation on the parameter `PncMembership` (This clears all PNC bits of the current membership besides the one provided by the parameter)
- When Control = 1, then the current PNC membership shall be applied with logical OR (disjunction) operation on the parameter `PncMembership` (This sets additional all PNC bits provided by the parameter to the current membership)
- Return with E_OK.

⌋

8.3.15 ComM_PreventWakeUp

[SWS_ComM_00156] Definition of API function ComM_PreventWakeUp

Upstream requirements: SRS_ModeMgm_09157, SRS_ModeMgm_09089

⌈

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`: Change of wake up status for the channel failed, e.g. ComMEcuGroupClassification disables the functionality (see [ECUC_ComM_00563])
Description	Changes the inhibition status COMM_NO_WAKEUP for the corresponding channel.	
Available via	ComM.h	

⌋

[SWS_ComM_00799]

Upstream requirements: SRS_ModeMgm_09089

⌈Configuration of ComM_PreventWakeUp: Configurable with `ComMWakeupInhibitionEnabled` (see [ECUC_ComM_00559]).⌋

8.3.16 ComM_LimitChannelToNoComMode

[SWS_ComM_00163] Definition of API function ComM_LimitChannelToNoComMode

Upstream requirements: SRS_ModeMgm_09157, SRS_ModeMgm_09071

⌈

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`: Change of inhibition status for the channel failed, e.g. ComMEcuGroupClassification disables the functionality (see [ECUC_ComM_00563])
Description	Changes the inhibition status for the channel for changing from COMM_NO_COMMUNICATION to a higher Communication Mode. (See also ComM_LimitECUToNoCom Mode, same functionality but for all channels)	
Available via	ComM.h	

⌋

[SWS_ComM_00800]

Upstream requirements: SRS_ModeMgm_09071

⌈Configuration of ComM_LimitChannelToNoComMode: Configurable with `ComMModeLimitationEnabled` (see [ECUC_ComM_00560]) and `ComMResetAfterForcingNoComm` (see [ECUC_ComM_00558]).⌋

8.3.17 ComM_LimitECUToNoComMode

[SWS_ComM_00124] Definition of API function ComM_LimitECUToNoComMode

Upstream requirements: SRS_ModeMgm_09157, SRS_ModeMgm_09071

⌈

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: Change of inhibition status for the ECU failed, e.g. ComMEcuGroupClassification disables the functionality (see [ECUC_ComM_00563])
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)	
Available via	ComM.h	

⌋

[SWS_ComM_00801]

Upstream requirements: SRS_ModeMgm_09071

⌈Configuration of ComM_LimitECUToNoComMode: Configurable with ComMModeLimitationEnabled (see [ECUC_ComM_00560]) and ComMResetAfterForcingNoComm (see [ECUC_ComM_00558]).⌋

8.3.18 ComM_ReadInhibitCounter

[SWS_ComM_00224] Definition of API function ComM_ReadInhibitCounter

Upstream requirements: SRS_ModeMgm_09156

⌈

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.	
Available via	ComM.h	

⌋

[SWS_ComM_00802]

Upstream requirements: SRS_ModeMgm_09156

⌈Configuration of ComM_ReadInhibitCounter: Configurable with `ComMModeLimitationEnabled` (see [ECUC_ComM_00560]). Function will only be available if `ComMModeLimitationEnabled` (see [ECUC_ComM_00560]) is enabled and `ComMGlobalNvMBlockDescriptor` is configured.⌋

8.3.19 ComM_ResetInhibitCounter

[SWS_ComM_00108] Definition of API function ComM_ResetInhibitCounter

Upstream requirements: SRS_ModeMgm_09156

⌈

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.	
Available via	ComM.h	

⌋

[SWS_ComM_00803]

Upstream requirements: SRS_ModeMgm_09155

⌈Configuration of ComM_ResetInhibitCounter: Configurable with `ComMModeLimitationEnabled` (see [ECUC_ComM_00560]). Function will only be available if `ComMModeLimitationEnabled` (see [ECUC_ComM_00560]) is enabled and `ComMGlobalNvMBlockDescriptor` is configured.⌋

8.3.20 ComM_SetECUGroupClassification

[SWS_ComM_00552] Definition of API function ComM_SetECUGroupClassification

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈

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)	
Available via	ComM.h	

⌋

8.3.21 AUTOSAR Network Management Interface

8.3.21.1 ComM_Nm_NetworkStartIndication

[SWS_ComM_00383] Definition of callback function ComM_Nm_NetworkStartIndication

Upstream requirements: SRS_ModeMgm_00049, SRS_ModeMgm_09132

⌈

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.	
Available via	ComM_Nm.h	

⌋

8.3.21.2 ComM_Nm_NetworkMode

[SWS_ComM_00390] Definition of callback function ComM_Nm_NetworkMode

Upstream requirements: SRS_ModeMgm_00049, SRS_ModeMgm_09132

⌈

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.	
Available via	ComM_Nm.h	

⌋

8.3.21.3 ComM_Nm_PrepareBusSleepMode

[SWS_ComM_00391] Definition of callback function ComM_Nm_PrepareBus SleepMode

Upstream requirements: SRS_ModeMgm_00049, SRS_ModeMgm_09132

⌈

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)	
Available via	ComM_Nm.h	

⌋

8.3.21.4 ComM_Nm_BusSleepMode

[SWS_ComM_00392] Definition of callback function ComM_Nm_BusSleepMode

Upstream requirements: SRS_ModeMgm_00049, SRS_ModeMgm_09132

⌈

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.	
Available via	ComM_Nm.h	

⌋

8.3.21.5 ComM_Nm_RestartIndication

[SWS_ComM_00792] Definition of callback function ComM_Nm_RestartIndication

Upstream requirements: SRS_ModeMgm_00049, SRS_ModeMgm_09132

⌈

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.	
Available via	ComM_Nm.h	

⌋

8.3.21.6 ComM_Nm_RepeatMessageLeftIndication

[SWS_ComM_91024] Definition of API function ComM_Nm_RepeatMessageLeftIndication

Upstream requirements: SRS_ModeMgm_09265

⌈

Service Name	ComM_Nm_RepeatMessageLeftIndication	
Syntax	`void ComM_Nm_RepeatMessageLeftIndication ( NetworkHandleType Channel )`	
Service ID [hex]	0x66	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	See NetworkHandleType
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notification that the state of all <Bus>Nm has left RepeatMessage. This interface is used to indicate by the optional Dynamic PNC-to-channel-mapping feature to indicate that learning phase ends.	
Available via	ComM_Nm.h	

⌋

8.3.21.7 ComM_Nm_PncLearningBitIndication

[SWS_ComM_91026] Definition of API function ComM_Nm_PncLearningBitIndication

Upstream requirements: SRS_ModeMgm_09261

⌈

Service Name	ComM_Nm_PncLearningBitIndication	
Syntax	`void ComM_Nm_PncLearningBitIndication ( NetworkHandleType Channel )`	
Service ID [hex]	0x69	
Sync/Async	Asynchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	See NetworkHandleType
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Service to indicate that an NM message with set PNC Learning Bit has been received.	
Available via	ComM_Nm.h	

⌋

8.3.21.8 ComM_Nm_ForwardSynchronizedPncShutdown

[SWS_ComM_91030] Definition of callback function ComM_Nm_ForwardSynchronizedPncShutdown

Upstream requirements: SRS_ModeMgm_09269

⌈

Service Name	ComM_Nm_ForwardSynchronizedPncShutdown	
Syntax	`void ComM_Nm_ForwardSynchronizedPncShutdown ( NetworkHandleType Channel, const uint8* PncRequestBitVectorPtr )`	
Service ID [hex]	0x6b	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	Channel
	PncRequestBitVectorPtr	Pointer to PNC Request Bit Vector with all PNC bits set to “1” which are indicated for a synchronized PNC shutdown
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	If an ECU in role of an intermediate PNC coordinator receives a PN shutdown message via a <Bus>Nm, then ComM is immediately indicated via ComM_Nm_ForwardSynchronizedPnc Shutdown to forward the request for a synchronized PNC shutdown of the affected PNCs given by PncRequestBitVectorPtr. Therefore, ComM will immediately release the affected PNC state machines and forward the PNC Request Bit Vector to the affected ComM Channels and the corresponding NM channels, respectively. Note: This supports a nearly synchronized PNC shutdown across the PN topology from the top-level PNC coordinator down to the subordinated PNC node.	
Available via	ComM_Nm.h	

⌋

8.3.21.9 ComM_Nm_UpdateEIRA

[SWS_ComM_91028] Definition of callback function ComM_Nm_UpdateEIRA

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09250

⌈

Service Name	ComM_Nm_UpdateEIRA	
Syntax	`void ComM_Nm_UpdateEIRA ( const uint8* PncRequestBitVectorPtr )`	
Service ID [hex]	0x6c	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PncRequestBitVectorPtr	Pointer to the PNC Request Bit Vector which contain the current aggregated internal and external PNC requests (EIRA)
Parameters (inout)	None	

▽

△

Parameters (out)	None
Return value	None
Description	Function to indicate the current aggregated external / internal PNC request called by Nm.
Available via	ComM_Nm.h

⌋

8.3.21.10 ComM_Nm_UpdateERA

[SWS_ComM_91029] Definition of callback function ComM_Nm_UpdateERA

Upstream requirements: SRS_ModeMgm_09248, SRS_ModeMgm_09250

⌈

Service Name	ComM_Nm_UpdateERA	
Syntax	`void ComM_Nm_UpdateERA ( NetworkHandleType Channel, const uint8* PncRequestBitVectorPtr )`	
Service ID [hex]	0x6d	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	Channel
	PncRequestBitVectorPtr	PNC Request Bit Vector which contain the current external PNC requests (ERA) received on the given channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Function to indicate the current external PNC request per channel called by Nm.	
Available via	ComM_Nm.h	

⌋

8.3.22 AUTOSAR Diagnostic Communication Manager Interface

8.3.22.1 ComM_DCM_ActiveDiagnostic

[SWS_ComM_00873] Definition of callback function ComM_DCM_ActiveDiagnostic ⌈

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.	
Available via	ComM_Dcm.h	

⌋

8.3.22.2 ComM_DCM_InactiveDiagnostic

[SWS_ComM_00874] Definition of callback function ComM_DCM_InactiveDiagnostic ⌈

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.	
Available via	ComM_Dcm.h	

⌋

8.3.23 AUTOSAR ECU State Manager Interface

8.3.23.1 ComM_EcuM_WakeUpIndication

[SWS_ComM_00275] Definition of callback function ComM_EcuM_WakeUpIndication ⌈

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.	
Available via	ComM_EcuM.h	

⌋

8.3.23.2 ComM_EcuM_PNCWakeUpIndication

[SWS_ComM_91001] Definition of callback function ComM_EcuM_PNCWakeUp Indication ⌈

Service Name	ComM_EcuM_PNCWakeUpIndication	
Syntax	`void ComM_EcuM_PNCWakeUpIndication ( PNCHandleType PNCid )`	
Service ID [hex]	0x37	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	PNCid	Identifier of the partial network cluster
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notification of a wake up on the corresponding partial network cluster.	
Available via	ComM_EcuM.h	

⌋

8.3.24 AUTOSAR ECU State Manager and Basic Software Mode Manager Interface

8.3.24.1 ComM_CommunicationAllowed

[SWS_ComM_00871] Definition of callback function ComM_CommunicationAllowed ⌈

Service Name	ComM_CommunicationAllowed
Syntax	`void ComM_CommunicationAllowed ( NetworkHandleType Channel, boolean Allowed )`
Service ID [hex]	0x35
Sync/Async	Asynchronous

▽

△

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/Flex is used: BswM	
Available via	ComM_BswM.h	

⌋

8.3.25 Bus State Manager Interface

8.3.25.1 ComM_<Bus>SM_ModeIndication

[SWS_ComM_00675] Definition of callback function ComM_<Bus>SM_ModeIndication

Upstream requirements: SRS_ModeMgm_09081

⌈

Service Name	ComM_<Bus>SM_ModeIndication	
Syntax	`void ComM_<Bus>SM_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.	
Available via	ComM.h	

⌋

8.3.25.2 ComM_<Bus>SM_BusSleepMode

[SWS_ComM_91000] Definition of callback function ComM_<Bus>SM_BusSleep Mode ⌈

Service Name	ComM_<Bus>SM_BusSleepMode	

Syntax	`void ComM_<Bus>SM_BusSleepMode (`
	`  NetworkHandleType Channel`
	`)`	

Service ID [hex]	0x34	
Sync/Async	Synchronous	
Reentrancy	Reentrant	
Parameters (in)	Channel	Identification of the channel
Parameters (inout)	None	
Parameters (out)	None	
Return value	None	
Description	Notification of the corresponding Bus State Manager that the actual bus mode is Bus-Sleep. Only applicable for ComM channels with ComMNmVariant set to SLAVE_ACTIVE or SLAVE_PASSIVE. E.g. LIN slaves (ComMNMVariant = SLAVE_ACTIVE) or Ethernet channels with OA TC10 compliant Ethernet hardware which act as passive communication slave (ComMNMVariant = SLAVE_PASSIVE and EthTrcvActAsSlavePassiveEnabled set to TRUE)	
Available via	ComM.h	

⌋

8.4 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.

8.4.1 ComM_MainFunction

[SWS_ComM_00429] Definition of scheduled function ComM_MainFunction_<ComMChannel.ShortName>

Upstream requirements: SRS_BSW_00373

⌈

Service Name	ComM_MainFunction_<ComMChannel.ShortName>

Syntax	`void ComM_MainFunction_<ComMChannel.ShortName> (`
	`  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
Available via	SchM_ComM.h

⌋

[SWS_ComM_00818] ⌈Channel.ShortName shall be used to configure ComM_MainFunction_<ComMChannel.ShortName> (see section 10.2.2) .⌋

Note: ComMChannel.ShortName is the short name of the ComMChannel container that will be managed by the ComM_MainFunction_<ComMChannel.ShortName> function

8.5 Expected interfaces

In this chapter all interfaces required from other modules are shown. An overview of the required interfaces is shown in Figure 5.1.

8.5.1 Mandatory interfaces

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

[SWS_ComM_00828] Definition of mandatory interfaces required by module Com M ⌈

API Function	*Header File*	*Description*
<Bus>SM_GetCurrentComMode	<Bus>SM.h	Query the current communication mode.
<Bus>SM_RequestComMode	<Bus>SM.h	Request of a new communication mode.
BswM_ComM_CurrentMode	BswM_ComM.h	Function called by ComM to indicate the current communication mode of a ComM channel.
Dcm_ComM_FullComModeEntered	Dcm_ComM.h	This call informs the Dcm module about a ComM mode change to COMM_FULL_COMMUNICATION.
Dcm_ComM_NoComModeEntered	Dcm_ComM.h	This call informs the Dcm module about a ComM mode change to COMM_NO_COMMUNICATION.
Dcm_ComM_SilentComModeEntered	Dcm_ComM.h	This call informs the Dcm module about a ComM mode change to COMM_SILENT_ COMMUNICATION.
Nm_NetworkRelease	Nm.h	This function calls the <Bus>Nm_NetworkRelease bus specific function in case NmBusType is not set to NM_BUSNM_LOCALNM (e.g. CanNm_Network Release function is called if channel is configured as CAN).

▽

△

API Function	Header File	Description
Nm_NetworkRequest	Nm.h	This function calls the <Bus>Nm_NetworkRequest (e.g. CanNm_NetworkRequest function is called if channel is configured as CAN) function in case Nm BusType is not set to NM_BUSNM_LOCALNM.
Nm_PassiveStartUp	Nm.h	This function calls the <Bus>Nm_PassiveStartUp function in case NmBusType is not set to NM_ BUSNM_LOCALNM (e.g. CanNm_PassiveStartUp function is called for NM_BUSNM_CANNM).
NvM_GetErrorStatus	NvM.h	Service to read the block dependent error/status information.
NvM_ReadBlock	NvM.h	Service to copy the data of the NV block to its corresponding RAM block.
NvM_RestorePRAMBlockDefaults	NvM.h	Service to restore the default data to its corresponding permanent RAM block.
NvM_WriteBlock	NvM.h	Service to copy the data of the RAM block to its corresponding NV block.

⌋

8.5.1.1 AUTOSAR NVRAM Manager module

[SWS_ComM_00103] Use standardized services of the NVRAM Manager module for storing and reading non-volatile configuration data ⌈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] and [SWS_ComM_00157])
- `ComMEcuGroupClassification` (see [SWS_ComM_01104])
- the Inhibit counter (see [SWS_ComM_00140])
- the PNC-to-channel Mapping (see [SWS_ComM_01040]) and the PNC membership (see [SWS_ComM_01049])

⌋

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 [13].

8.5.1.2 AUTOSAR Bus State Manager

[SWS_ComM_00962]

Upstream requirements: SRS_ModeMgm_09155

⌈The prefix for the StateManager APIs ("<Bus>SM") 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]

Upstream requirements: SRS_ModeMgm_09207

⌈If ComMBusType = "COMM_BUS_TYPE_CDD" the API prefix ("<Bus>SM") shall be configured in the Parameter "ComMCDDBusPrefix".⌋

[SWS_ComM_00963] ⌈The Communication Manager module shall use <Bus>SM_GetCurrentComMode() from the State Manager to query the current communication mode if necessary.⌋

[SWS_ComM_00958] Use <Bus>SM_RequestComMode to request a dedicated communication mode ⌈The Communication Manager module shall use <Bus>SM_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 [14], [15], [16], [17].

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

8.5.1.3 AUTOSAR Network Management Interface

[SWS_ComM_00261]

Upstream requirements: SRS_ModeMgm_00049

⌈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 [18].⌋

8.5.1.4 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 [19].

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

[SWS_ComM_00091]

Upstream requirements: SRS_ModeMgm_09085

⌈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.⌋

[SWS_ComM_00663]

Upstream requirements: SRS_ModeMgm_09085

⌈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]

Upstream requirements: SRS_ModeMgm_09090

⌈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]

Upstream requirements: SRS_ModeMgm_09085

⌈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 [6] and AUTOSAR Services Mode Management specification [10].

8.5.1.6 Basic Software Mode Manager (BswM)

[SWS_ComM_00861]

Upstream requirements: SRS_ModeMgm_09251

⌈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 [20].

8.5.2 Optional interfaces

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

[SWS_ComM_00829] Definition of optional interfaces requested by module Com M ⌈

API Function	*Header File*	*Description*
BswM_ComM_CurrentPNCMode	BswM_ComM.h	Function is called by ComM to indicate the current mode of the PNC.
BswM_ComM_InitiateReset	BswM_ComM.h	Function is called by ComM to signal a shutdown.
Det_ReportRuntimeError	Det.h	Service to report runtime errors. If a callout has been configured then this callout shall be called.
Nm_PnLearningRequest	Nm.h	Set Repeat Message Request Bit and Partial Network Learning Bit for NM messages transmitted next on the bus. For that purpose <Bus>Nm_Pn LearningRequest shall be called (e.g. CanNm_Pn LearningRequest function if channel is configured as CAN). This will force all nodes to enter the PNC Learning Phase and re-enter Repeat Message Stat. This is needed for the optional Dynamic PNC-to-channel-mapping feature.
Nm_RequestSynchronizedPnc Shutdown	Nm.h	This function store the request for a synchronized PNC shutdown of a particular PNC given by PncId per given NM-Channel. The handling of the synchronized PNC shutdown process is mainly done in the context of the Nm_Mainfunction. The function call is only valid if NmStandardBusType is not set to NM_BUSNM_LOCALNM as a <Bus>Nm like CanNm is needed to transmit the PNC shutdown requests.
Nm_UpdateIRA	Nm.h	Indication by ComM of internal PNC requests. This is used to aggregate the internal PNC requests.

⌋

8.5.3 Callout interfaces

None.

8.5.4 Configurable interfaces

None.

8.6 Service Interfaces

8.6.1 Sender-Receiver-Interfaces

8.6.1.1 ComM_CurrentChannelRequest

[SWS_ComM_00904] Definition of SenderReceiverInterface ComM_CurrentChannelRequest_{channel_name}

Upstream requirements: SRS_ModeMgm_09149

⌈

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)}

⌋

8.6.2 Client-Server-Interfaces

8.6.2.1 ComM_ChannelLimitation

[SWS_ComM_00743] Definition of ClientServerInterface ComM_ChannelLimitation

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈

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	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetInhibitionStatus
Comment	returns the inhibition status of a channel
Relates to	ComM_GetInhibitionStatus
Variation	–

▽

△

Parameters	Status	
	Type	ComM_InhibitionStatusType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	LimitChannelToNoComMode	
Comment	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)	
Relates to	ComM_LimitChannelToNoComMode	
Variation	–	
Parameters	Status	
	Type	boolean
	Direction	IN
	Comment	FALSE: Limit channel to COMM_NO_COMMUNICATION disabled TRUE: Limit channel to COMM_NO_COMMUNICATION enabled
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.6.2.2 ComM_ChannelWakeup

[SWS_ComM_00742] Definition of ClientServerInterface ComM_ChannelWakeup

Upstream requirements: SRS_ModeMgm_09089

⌈

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	Operation successful
	1	E_NOT_OK	Operation failed

Operation	GetInhibitionStatus	
Comment	returns the inhibition status of a channel	
Relates to	ComM_GetInhibitionStatus	
Variation	–	
Parameters	Status	
	Type	ComM_InhibitionStatusType

▽

△

	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	PreventWakeUp	
Comment	Changes the inhibition status COMM_NO_WAKEUP for the corresponding channel.	
Relates to	ComM_PreventWakeUp	
Variation	–	
Parameters	Status	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.6.2.3 ComM_ECUModeLimitation

[SWS_ComM_00741] Definition of ClientServerInterface ComM_ECUModeLimitation

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09156, SRS_ModeMgm_09157

⌈

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	Operation successful
	1	E_NOT_OK	Operation failed

Operation	LimitECUToNoComMode	
Comment	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)	
Relates to	ComM_LimitECUToNoComMode	
Variation	–	
Parameters	Status	
	Type	boolean
	Direction	IN

▽

△

	Comment	FALSE: Limit ECU to COMM_NO_COMMUNICATION disabled TRUE: Limit ECU to COMM_NO_COMMUNICATION enabled
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	ReadInhibitCounter	
Comment	returns the value of the 'inhibited full communication request counter'	
Relates to	ComM_ReadInhibitCounter	
Variation	{ecuc(ComM/ComMGeneral.ComMGlobalNvMBlockDescriptor)} != NULL	
Parameters	CounterValue	
	Type	uint16
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	ResetInhibitCounter	
Comment	reset the "inhibited full communication request counter"	
Relates to	ComM_ResetInhibitCounter	
Variation	{ecuc(ComM/ComMGeneral.ComMGlobalNvMBlockDescriptor)} != NULL	
Possible Errors	E_OK E_NOT_OK	

Operation	SetECUGroupClassification	
Comment	changes the ECU group classification status	
Relates to	ComM_SetECUGroupClassification	
Variation	–	
Parameters	Status	
	Type	ComM_InhibitionStatusType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

⌋

8.6.2.4 ComM_UserRequest

[SWS_ComM_01000] Definition of ClientServerInterface ComM_UserRequest

Upstream requirements: SRS_ModeMgm_09081

⌈

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 could either set the desired state of all communication channels (if the user is mapped to one or more channels) or of all PNCs (if the user is mapped to one or more PNCs) that are relevant for that component to "No Communication" or "Full Communication".		
IsService	true		
Variation	–		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	2	COMM_E_MODE_ LIMITATION	ComMMode cannot be granted because of Com MMode inhibition
	3	COMM_E_MULTIPLE_ PNC_ASSIGNED	Operation is not possible since multiple PNCs are assigned to the affected ComMUser
	4	COMM_E_NO_PNC_ ASSIGNED	Operation is not possible since no PNC is assigned to the affected ComMUser

Operation	GetCurrentComMode	
Comment	Returns the current Communication Manager Module mode for the SW-C-Return the current Communication Manager Module channel mode to the SW-C. Please note: the channel mode is returned. Even though the affected user is assigned to a PNC. (see ComM_GetCurrentCom Mode)	
Relates to	ComM_GetCurrentComMode	
Variation	–	
Parameters	ComMode	
	Type	ComM_ModeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK	

Operation	GetCurrentPNCComMode	
Comment	Return the current Communication Manager Module PNC mode to the SW-C. Please note: the PNC mode is returned as ComM_ModeType (COMM_NO_COMMUNICATION == COMM_ PNC_NO_COMMUNICATION, COMM_FULL_COMMUNICATION == COMM_PNC_FULL_ COMMUNICATION). If the affected ComM user is mapped to multiple PNCs than the operation shall return COMM_E_MULTIPLE_PNC_ASSIGNED. If the affected ComM user is mapped to no PNC than the operation shall return COMM_E_NO_PNC_ASSIGNED.	
Relates to	ComM_GetCurrentPNCComMode	
Variation	–	
Parameters	ComMode	
	Type	ComM_ModeType
	Direction	OUT
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK COMM_E_MULTIPLE_PNC_ASSIGNED COMM_E_NO_PNC_ASSIGNED	

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

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

Operation	RequestComMode	
Comment	The SW-C requests that all communication channels it needs are in the provided Communication Manager Module mode	
Relates to	ComM_GetRequestedComMode	
Variation	–	
Parameters	ComMode	
	Type	ComM_ModeType
	Direction	IN
	Comment	–
	Variation	–
Possible Errors	E_OK E_NOT_OK COMM_E_MODE_LIMITATION	

⌋

8.6.2.5 ComM_PncToChannelMapping

[SWS_ComM_91102] Definition of ClientServerInterface ComM_PncToChannel Mapping

Upstream requirements: SRS_ModeMgm_09259

⌈

Name	ComM_PncToChannelMapping		
Comment	Client-server interface to get, update or clear the PNC-to-channel-mapping		
IsService	true		
Variation	{ecuc(ComM/ComMGeneral/ComMDynamicPncToChannelMappingSupport)} == true		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	5	COMM_E_LEARNING_ ACTIVE	Operation not possible as PNC Learning Phase is active

Operation	GetPncToChannelMapping	
Comment	Returns the current PNC-to-channel-mapping **Tags:** atp.Status=draft	
Relates to	ComM_GetPncToChannelMapping	
Variation	–	
Parameters	MappingTable	
	Type	uint8*
	Direction	OUT
	Comment	Pointer to an array of uint8 which stores for each channel a PNC Request Bit Vector representing the mapping of PNCs to the channel
	Variation	–
	ChannelCnt	
	Type	uint8
	Direction	OUT
	Comment	Number of physical channels passed in the MappingTable
	Variation	–
	PNCRequestBitVectorLength	
	Type	uint8
	Direction	OUT
	Comment	Length in bytes of the PNC Request Bit Vector where each bit represents one PNC
	Variation	–
Possible Errors	E_OK E_NOT_OK COMM_E_LEARNING_ACTIVE	

Operation	ResetPncToChannelMapping
Comment	Resets the current PNC-to-channel mapping to its static configured default **Tags:** atp.Status=draft
Relates to	ComM_ResetPncToChannelMapping
Variation	–
Possible Errors	E_OK E_NOT_OK COMM_E_LEARNING_ACTIVE

Operation	UpdatePncToChannelMapping
Comment	Updates the current PNC-to-channel-mapping **Tags:** atp.Status=draft
Relates to	ComM_UpdatePncToChannelMapping
Variation	–

▽

△

Parameters	MappingTable	
	Type	const uint8*
	Direction	IN
	Comment	Pointer to an array of uint8 which stores for each channel a PNC Request Bit Vector representing the mapping of PNCs to the channel
	Variation	–
	channelCnt	
	Type	uint8
	Direction	IN
	Comment	Number of physical channels passed in the MappingTable
	Variation	–
	PNCRequestBitVectorLength	
	Type	uint8
	Direction	IN
	Comment	Length in bytes of the PNC Request Bit Vector where each bit represents one PNC
	Variation	–
Possible Errors	E_OK E_NOT_OK COMM_E_LEARNING_ACTIVE	

⌋

8.6.2.6 ComM_DynamicPncToChannelMapping

[SWS_ComM_91108] Definition of ClientServerInterface ComM_DynamicPncToChannelMapping

Upstream requirements: SRS_ModeMgm_09260, SRS_ModeMgm_09263

⌈

Name	ComM_DynamicPncToChannelMapping		
Comment	A SW-C can use this interface in order to update during runtime the PNC membership and trigger a learning request by sending NM messages with Partial Network Learning and Repeat Message Request bits set.		
IsService	true		
Variation	{ecuc(ComM/ComMGeneral/ComMDynamicPncToChannelMappingSupport)} == true		
Possible Errors	0	E_OK	Operation successful
	1	E_NOT_OK	Operation failed
	5	COMM_E_LEARNING_ACTIVE	Operation not possible as PNC Learning Phase is active

Operation	ComM_PnLearningRequest
Comment	Triggers a learning request
Relates to	ComM_PnLearningRequest
Variation	–

▽

△

Possible Errors	E_OK E_NOT_OK COMM_E_LEARNING_ACTIVE

Operation	ComM_UpdatePncMembership	
Comment	Used by SWCs to update the PNC membership which is transmitted during PNC Learning	
Relates to	ComM_UpdatePncMembership	
Variation	–	
Parameters	Control	
	Type	boolean
	Direction	IN
	Comment	–
	Variation	–
	PncMembership	
	Type	const uint8*
	Direction	IN
	Comment	Array of uint8 with <PNC Request Bit Vector length> Elements that holds the current PNC Membership of the node.
	Variation	–
Possible Errors	E_OK E_NOT_OK COMM_E_LEARNING_ACTIVE	

⌋

8.6.3 Mode-Switch-Interfaces

8.6.3.1 ComM_CurrentMode

[SWS_ComM_01001] Definition of ModeSwitchInterface ComM_CurrentMode

Upstream requirements: SRS_ModeMgm_09172, SRS_ModeMgm_09085

⌈

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

⌋

8.6.4 Implementation Data Types

8.6.4.1 ComM_InhibitionStatusType

[SWS_ComM_00669] Definition of ImplementationDataType ComM_InhibitionStatusType

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈

Name	ComM_InhibitionStatusType			
Kind	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			
Variation	–			
Available via	Rte_ComM_Type.h			

⌋

8.6.4.2 ComM_ModeType

[SWS_ComM_00672] Definition of ImplementationDataType ComM_ModeType

Upstream requirements: SRS_ModeMgm_09081

⌈

Name	ComM_ModeType		
Kind	Type		
Derived from	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.
	COMM_FULL_COMMUNICATION_WITH_WAKEUP_REQUEST	3	ComM state machine is in "Full Communication" mode. Configured channel shall have both transmission and reception towards the lower layer (e.g. Ethernet hardware compliant to OA TC10). This is only for internal use within the ComM channel statemachine.
Description	Current mode of the Communication Manager (main state of the state machine).		
Variation	–		
Available via	Rte_ComM_Type.h		

⌋

8.6.4.3 ComM_UserHandleType

[SWS_ComM_00670] Definition of ImplementationDataType ComM_UserHandleType

Upstream requirements: SRS_ModeMgm_09078

⌈

Name	ComM_UserHandleType
Kind	Type
Derived from	uint16
Description	Handle to identify a user. For each user, a unique value must be defined at system generation time. Maximum number of users is 65535. Legal user IDs are in the range 0 .. 65534; user ID 65535 is reserved and shall have the symbolic representation COMM_NOT_USED_USER_ID.
Variation	–
Available via	Rte_ComM_Type.h

⌋

8.6.4.4 ComM_UserHandleArrayType

[SWS_ComM_00906] Definition of ImplementationDataType ComM_UserHandleArrayType_{channel_name}

Upstream requirements: SRS_ModeMgm_09149

⌈

Name	ComM_UserHandleArrayType_{channel_name}	
Kind	Structure	
Elements	numberOfRequesters	
	Type	uint8
	Comment	–
	handleArray	
	Type	ComM_UserHandleSubArrayType_{channel_name}
	Comment	–
	Variation	channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-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)}	
Available via	Rte_ComM_Type.h	

⌋

8.6.4.5 ComM_UserHandleSubArrayType

[SWS_ComM_01005] Definition of ImplementationDataType ComM_UserHandleSubArrayType_{channel_name}

Upstream requirements: SRS_ModeMgm_09149

⌈

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)}		
Available via	Rte_ComM_Type.h		

⌋

8.6.5 Ports

8.6.5.1 ComM_CL

[SWS_ComM_01006] Definition of Port CL_{channel_name} provided by module ComM

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09157

⌈

Name	CL_{channel_name}		
Kind	ProvidedPort	***Interface***	ComM_ChannelLimitation
Description	–		
Port Defined Argument Value(s)	***Type***	NetworkHandleType	
	Value	{ecuc(ComM/ComMConfigSet/ComMChannel/ComMChannelId.value)}	
Variation	{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel)}		

⌋

8.6.5.2 ComM_CR

[SWS_ComM_01007] Definition of Port CR_{channel_name} provided by module ComM

Upstream requirements: SRS_ModeMgm_09149

⌈

Name	CR_{channel_name}		
Kind	ProvidedPort	***Interface***	ComM_CurrentChannelRequest_{channel_name}
Description	–		
Variation	{ecuc(ComM/ComMConfigSet/ComMChannel/ComMFullCommRequestNotificationEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}		

⌋

8.6.5.3 ComM_CW

[SWS_ComM_01008] Definition of Port CW_{channel_name} provided by module ComM

Upstream requirements: SRS_ModeMgm_09089, SRS_ModeMgm_09157

⌈

Name	CW_{channel_name}		
Kind	ProvidedPort	***Interface***	ComM_ChannelWakeup
Description	–		
Port Defined Argument Value(s)	***Type***	NetworkHandleType	
	Value	{ecuc(ComM/ComMConfigSet/ComMChannel/ComMChannelId.value)}	
Variation	{ecuc(ComM/ComMGeneral.ComMWakeupInhibitionEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel)}		

⌋

8.6.5.4 ComM_modeLimitation

[SWS_ComM_01009] Definition of Port modeLimitation provided by module ComM

Upstream requirements: SRS_ModeMgm_09071, SRS_ModeMgm_09156, SRS_ModeMgm_09157

⌈

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

⌋

8.6.5.5 ComM_UM

[SWS_ComM_01010] Definition of Port UM_{user_name} provided by module ComM

Upstream requirements: SRS_ModeMgm_09172, SRS_ModeMgm_09085

⌈

Name	UM_{user_name}		
Kind	ProvidedPort	***Interface***	ComM_CurrentMode
Description	–		
Variation	user_name = {ecuc(ComM/ComMConfigSet/ComMUser.SHORT-NAME)}		

⌋

8.6.5.6 ComM_UR

[SWS_ComM_01011] Definition of Port UR_{user_name} provided by module ComM

Upstream requirements: SRS_ModeMgm_09081, SRS_ModeMgm_09071, SRS_ModeMgm_09157, SRS_ModeMgm_09078

⌈

Name	UR_{user_name}		
Kind	ProvidedPort	***Interface***	ComM_UserRequest
Description	–		
Port Defined Argument Value(s)	***Type***	ComM_UserHandleType	
	Value	ecuc(ComM/ComMConfigSet/ComMUser/ComMUserIdentifier.value)}	
Variation	user_name = {ecuc(ComM/ComMConfigSet/ComMUser.SHORT-NAME)}		

⌋

8.6.5.7 ComM_PncToChannelMapping

[SWS_ComM_91107] Definition of Port PncToChannelMapping provided by module ComM

Upstream requirements: SRS_ModeMgm_09259

⌈

Name	PncToChannelMapping		
Kind	ProvidedPort	***Interface***	ComM_PncToChannelMapping
Description	–		
Variation	{ecuc(ComM/ComMGeneral/ComMDynamicPncToChannelMappingSupport)} == true		

⌋

8.6.5.8 ComM_DynamicPncToChannelMapping

[SWS_ComM_91109] Definition of Port ComM_DynamicPncToChannelMapping provided by module ComM

Upstream requirements: SRS_ModeMgm_09260, SRS_ModeMgm_09263

⌈

Name	ComM_DynamicPncToChannelMapping		
Kind	ProvidedPort	***Interface***	ComM_DynamicPncToChannelMapping
Description	–		
Variation	{ecuc(ComM/ComMGeneral/ComMDynamicPncToChannelMappingSupport)} == true		

⌋

8.6.6 ModeDeclarationGroups

8.6.6.1 ComMMode

[SWS_ComM_01012] Definition of ModeDeclarationGroup ComMMode

Upstream requirements: SRS_ModeMgm_09172, SRS_ModeMgm_09085

⌈

Name	ComMMode	
Kind	ModeDeclarationGroup	
Category	ALPHABETIC_ORDER	
Initial mode	COMM_NO_COMMUNICATION	
On transition value	–	
Modes	COMM_FULL_COMMUNICATION	–
	COMM_NO_COMMUNICATION	–
	COMM_SILENT_COMMUNICATION	–
Description	–	

⌋

9 Sequence diagrams

9.1 Transmission and Reception start (CAN)

Figure 9.1 shows the sequence for starting transmission and reception on CAN. The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.

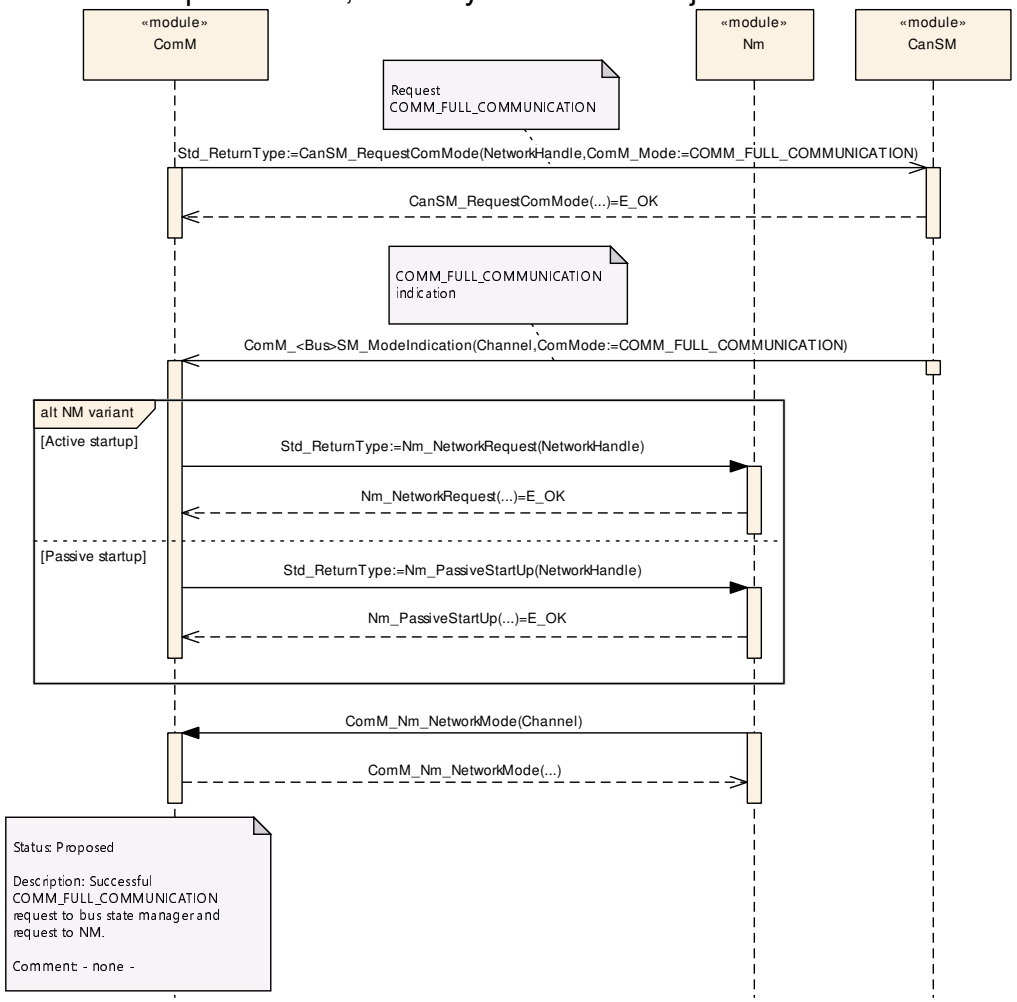


Figure 9.1: Starting transmission and reception on CAN

9.2 Passive Wake-up (CAN)

Figure 9.2 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.

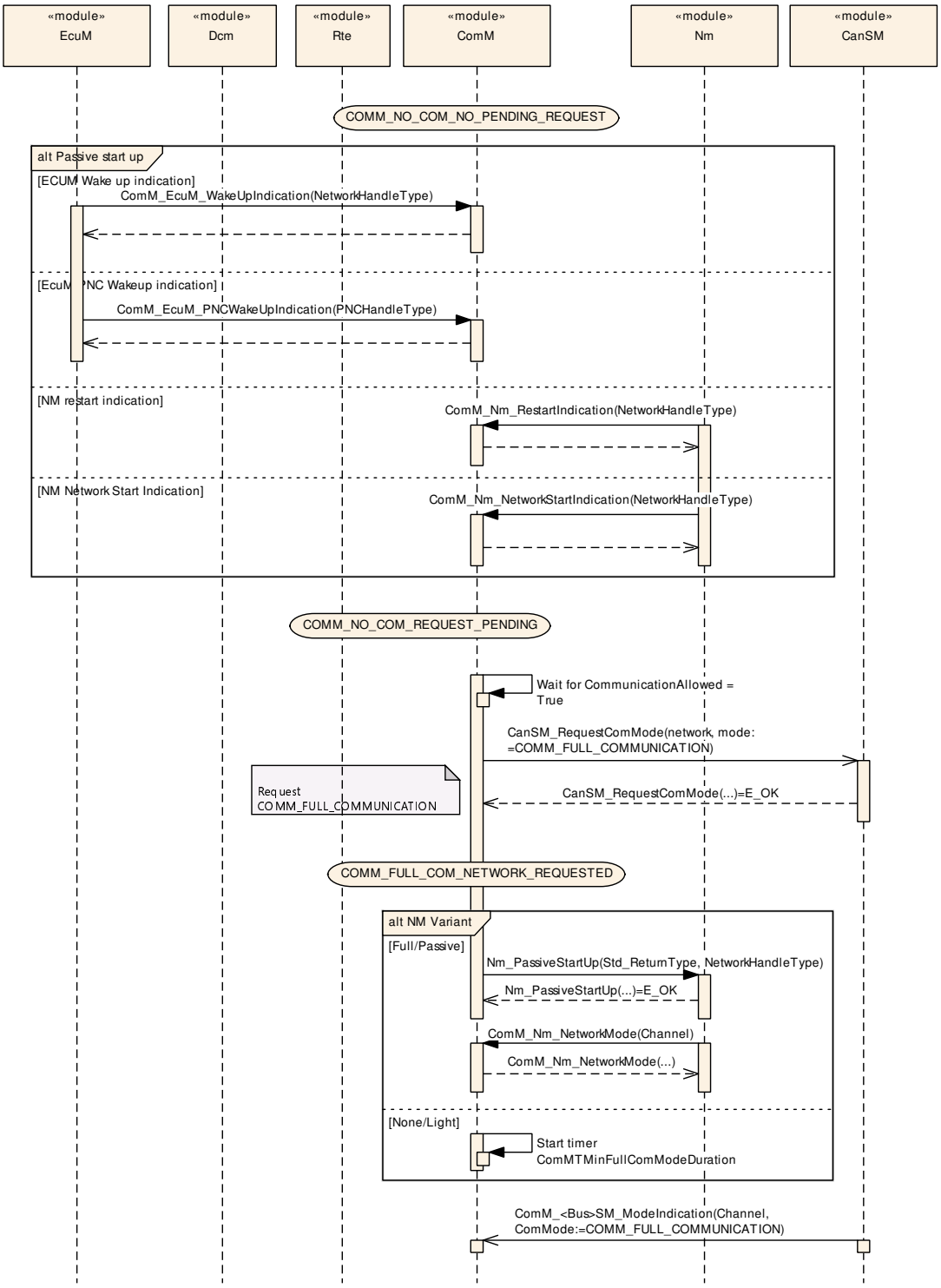


Figure 9.2: Reaction on a wake-up indicated by the ECU State Manager module

9.3 Network shutdown (CAN)

Figure 9.3 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.

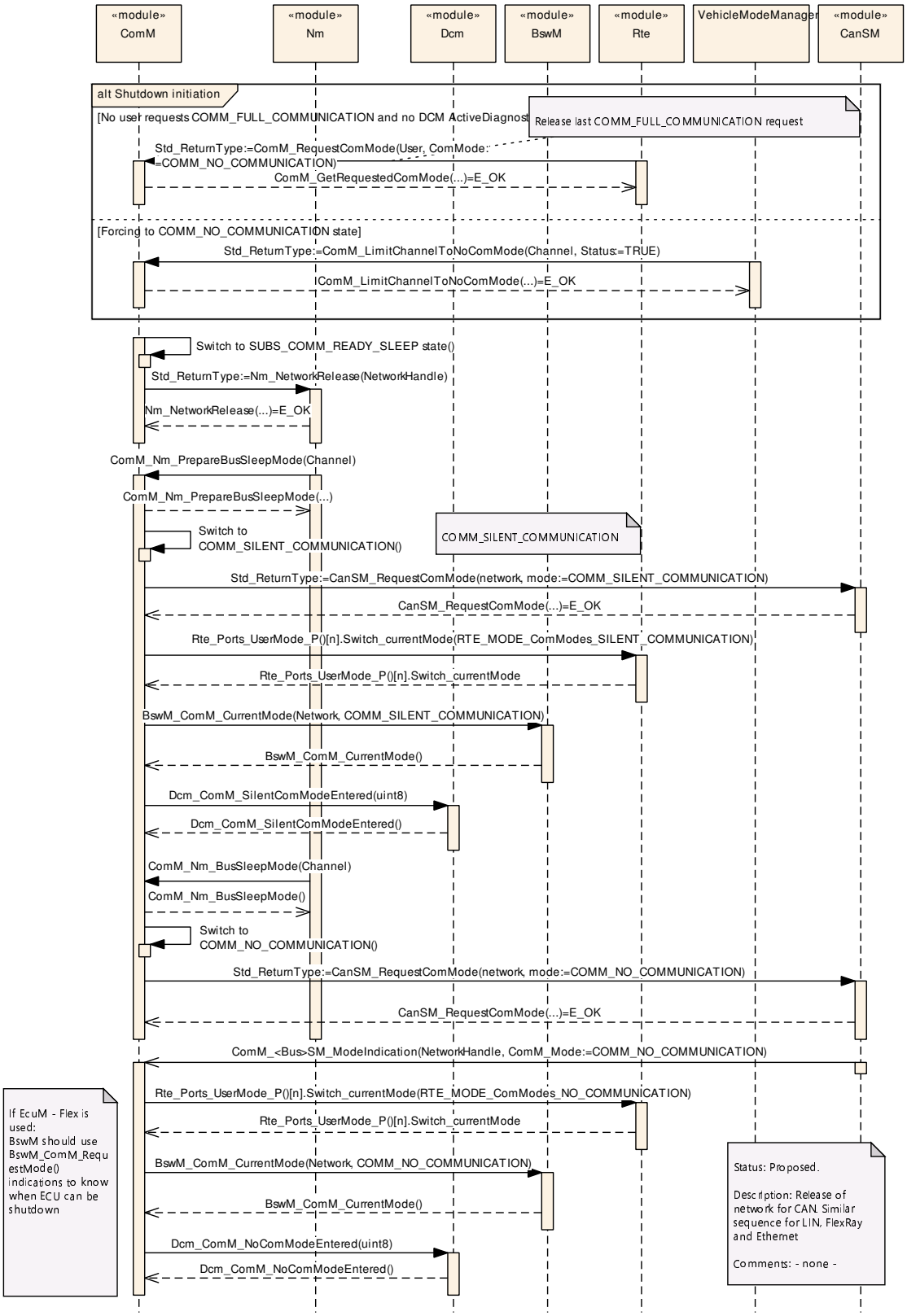


Figure 9.3: Network shutdown (CAN)

9.4 Communication request

Figure 9.4 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.

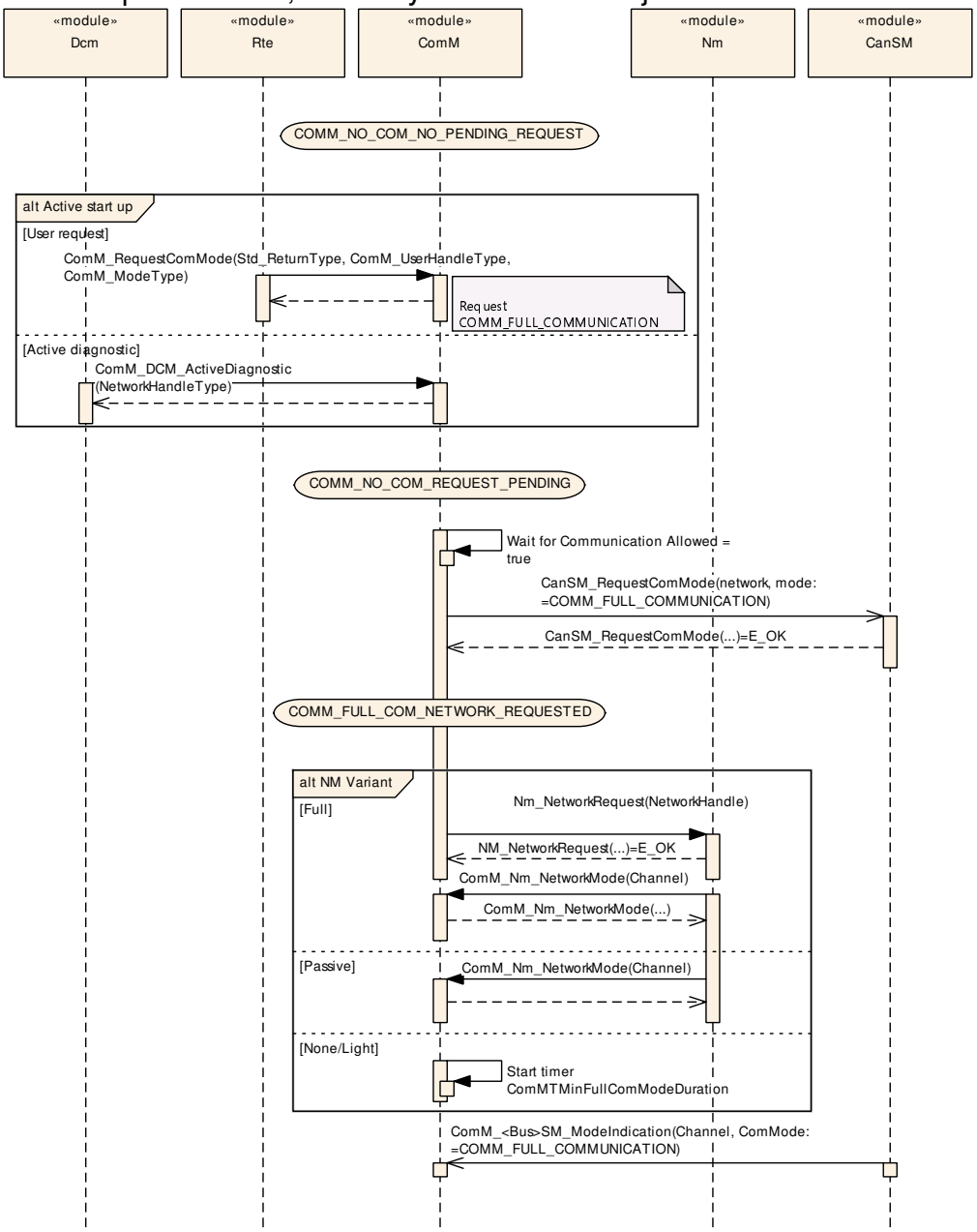


Figure 9.4: Request Communication

9.5 Synchronized PNC shutdown

Note: The sequence diagrams shows the expected behaviour, but not the implementation

Figure 9.5 shows the request for a synchronized PNC shutdown if an ECU in the role of a top-level PNC coordinator detects a release of a PNC.

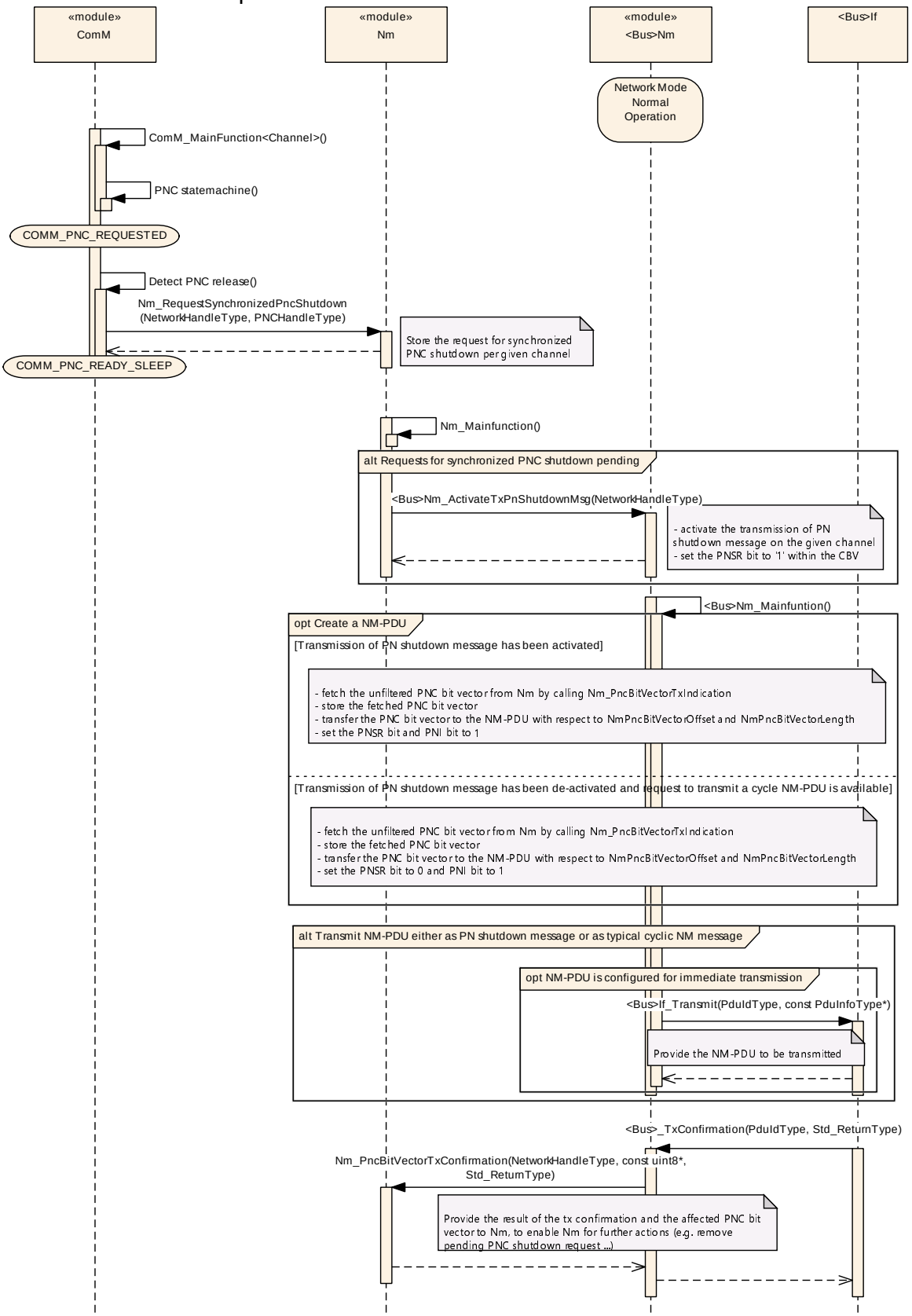


Figure 9.5: Request for a synchronized PNC shutdown in the role of a top-level PNC coordinator (TLPC)

Figure 9.6 and Figure 9.7 shows the request to forward a received synchronized PNC shutdown if an ECU in role of an intermediate PNC coordinator receives a PN shutdown message.

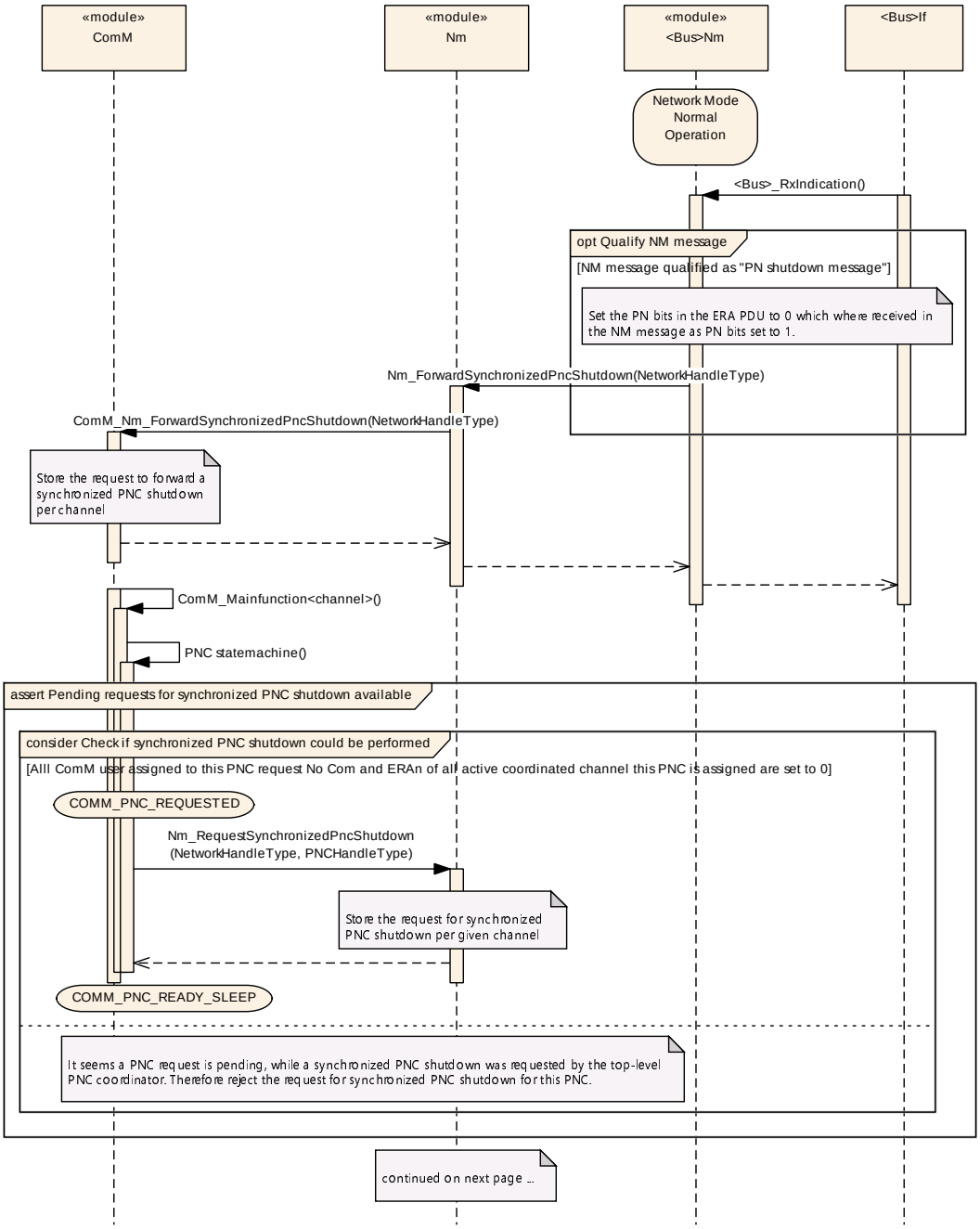


Figure 9.6: Request to forward a synchronized PNC shutdown in the role of an intermediate PNC coordinator (part 1)

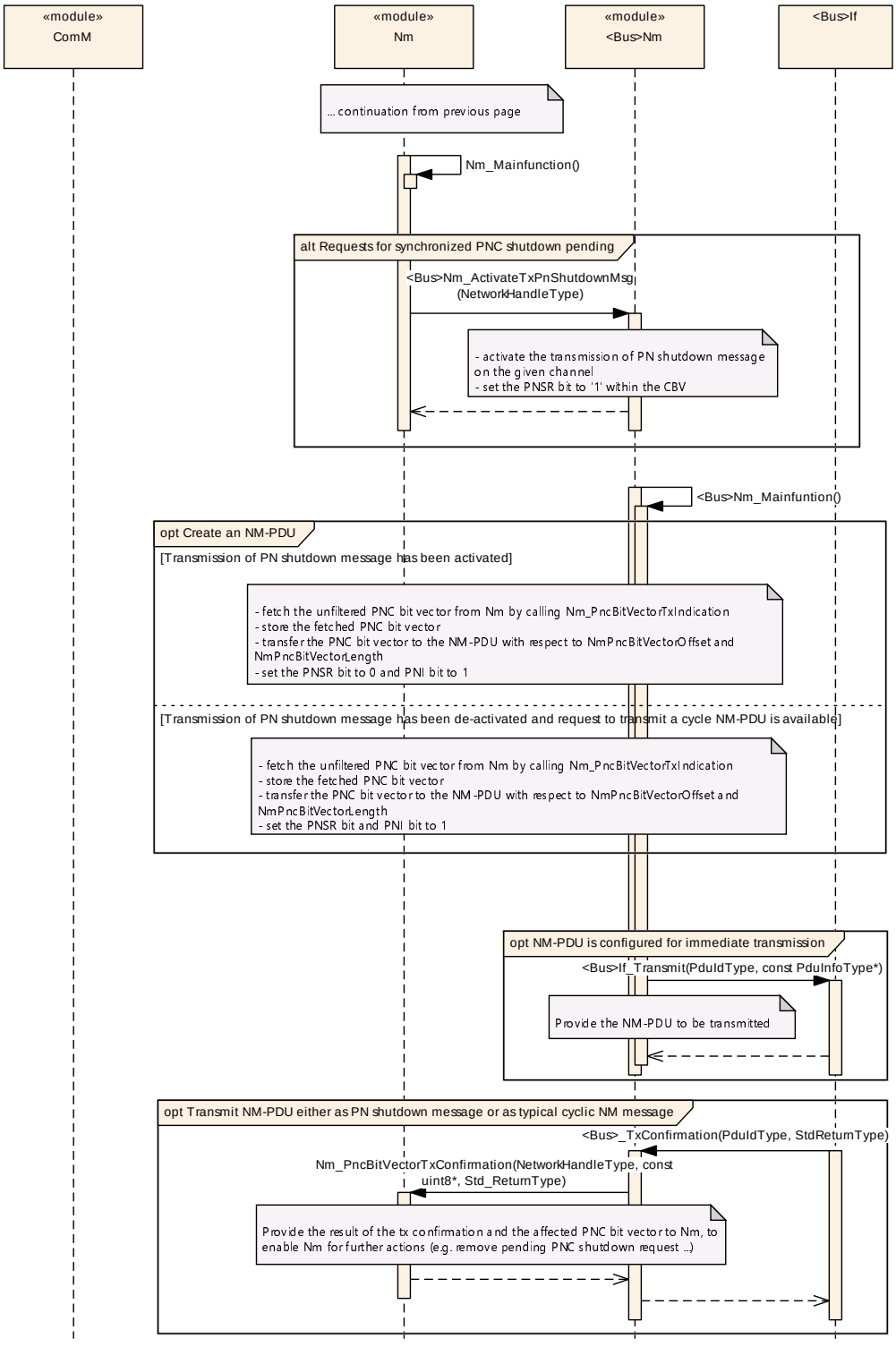


Figure 9.7: Request to forward a synchronized PNC shutdown in the role of an intermediate PNC coordinator (part 2)

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.

Chapter 10.3 specifies published information of the module Communication Manager.

10.1 How to read this chapter

For details refer to [5] Chapter 10.1 *"Introduction to configuration specification"*.

10.2 Containers and configuration parameters

The following chapters summarize all configuration parameters. The detailed meanings of the parameters describe Chapter 7 and Chapter 8.

[SWS_ComM_00419]

Upstream requirements: SRS_BSW_00167

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

[SWS_ComM_00322]

Upstream requirements: SRS_ModeMgm_09133

⌈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]

Upstream requirements: SRS_BSW_00345, SRS_BSW_00159, SRS_BSW_00167

⌈The ComM module shall strictly separate configuration from implementation.⌋

Rationale for [SWS_ComM_00464]: Easy and clear configuration.

10.2.1 ComM

[ECUC_ComM_00890] Definition of EcucModuleDef ComM ⌈

Module Name	ComM
Description	Configuration of the ComM (Communications Manager) module.
Post-Build Selectable Support	true
Post-Build Loadable Support	true

Included Containers		
Container Name	Multiplicity	Description
ComMConfigSet	1	This container contains the configuration parameters and sub containers of the AUTOSAR ComM module.
ComMGeneral	1	General configuration parameters of the Communication Manager.

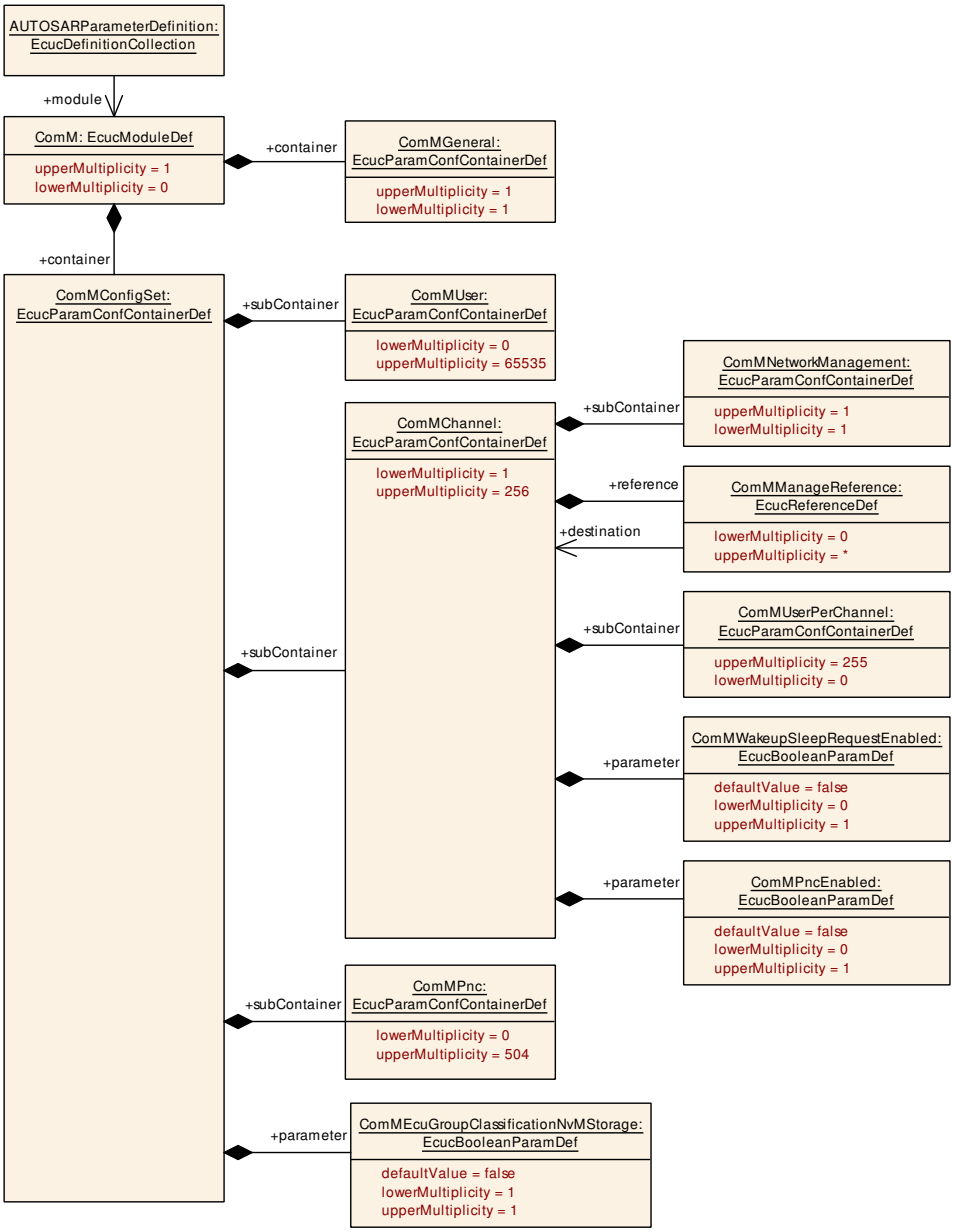


Figure 10.1: Configuration ComM

10.2.2 ComMGeneral

[ECUC_ComM_00554] Definition of EcucParamConfContainerDef ComMGeneral ⌈

Container Name	ComMGeneral
Parent Container	ComM
Upstream mapping defined	–
Description	General configuration parameters of the Communication Manager.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComM0PncVectorAvoidance	0..1	[ECUC_ComM_00892]
ComMDevErrorDetect	1	[ECUC_ComM_00555]
ComMDirectUserMapping	0..1	[ECUC_ComM_00840]
ComMDynamicPncToChannelMappingSupport	1	[ECUC_ComM_00895]
ComMEcuGroupClassification	1	[ECUC_ComM_00563]
ComMModeLimitationEnabled	1	[ECUC_ComM_00560]
ComMPncGatewayEnabled	0..1	[ECUC_ComM_00887]
ComMPncPrepareSleepTimer	0..1	[ECUC_ComM_00841]
ComMPncSupport	1	[ECUC_ComM_00839]
ComMResetAfterForcingNoComm	1	[ECUC_ComM_00558]
ComMSynchronizedPncShutdownEnabled	0..1	[ECUC_ComM_00897]
ComMSynchronousWakeUp	1	[ECUC_ComM_00695]
ComMTMinFullComModeDuration	1	[ECUC_ComM_00557]
ComMWakeupInhibitionEnabled	1	[ECUC_ComM_00559]
ComMGlobalNvMBlockDescriptor	0..1	[ECUC_ComM_00783]

No Included Containers

⌋

[ECUC_ComM_00892] Definition of EcucBooleanParamDef ComM0PncVectorAvoidance ⌈

Parameter Name	ComM0PncVectorAvoidance
Parent Container	ComMGeneral
Upstream mapping defined	–
Description	This parameter avoids sending of 0-PNC-Vectors in case ComMPncGatewayEnabled is enabled. Dependency: ComMPncGatewayEnabled is enabled
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00555] Definition of EcucBooleanParamDef ComMDevErrorDetect ⌈

Parameter Name	ComMDevErrorDetect
Parent Container	ComMGeneral
Upstream mapping defined	–
Description	Switches the development error detection and notification on or off. • true: detection and notification is enabled. • false: detection and notification is disabled.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00840] Definition of EcucBooleanParamDef ComMDirectUserMapping ⌈

Parameter Name	ComMDirectUserMapping
Parent Container	ComMGeneral
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_ComM_00895] Definition of EcucBooleanParamDef ComMDynamicPncToChannelMappingSupport ⌈

Parameter Name	ComMDynamicPncToChannelMappingSupport
Parent Container	ComMGeneral
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	Precompile time switch to enable the dynamic PNC-to-channel-mapping handling. False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled Dependency: shall only be TRUE if ComMPncSupport = TRUE
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00563] Definition of EcucIntegerParamDef ComMEcuGroupClassification ⌈

Parameter Name	ComMEcuGroupClassification
Parent Container	ComMGeneral
Upstream mapping defined	–
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
Multiplicity	1
Type	EcucIntegerParamDef
Range	0 .. 255
Default value	3
Pre-Compile only	true

⌋

[ECUC_ComM_00560] Definition of EcucBooleanParamDef ComMModeLimitationEnabled ⌈

Parameter Name	ComMModeLimitationEnabled
Parent Container	ComMGeneral
Upstream mapping defined	–
Description	true if mode limitation functionality shall be enabled. true: Enabled false: Disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00887] Definition of EcucBooleanParamDef ComMPncGatewayEnabled ⌈

Parameter Name	ComMPncGatewayEnabled
Parent Container	ComMGeneral
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
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
Pre-Compile only	true

⌋

[ECUC_ComM_00841] Definition of EcucFloatParamDef ComMPncPrepareSleepTimer ⌈

Parameter Name	ComMPncPrepareSleepTimer	
Parent Container	ComMGeneral	
Upstream mapping defined	[11, CP_TPS_SystemTemplate]	
Description	Time in seconds the PNC state machine shall wait in COMM_PNC_PREPARE_SLEEP. Dependency: #CanNm: (NmPnResetTime + ComMPncPrepareSleepTimer) < CanNm TimeoutTime # FrNm: (NmPnResetTime + ComMPncPrepareSleepTimer) < ((FrNm ReadySleepCnt +1) * FrNmRepetitionCycle * "Duration of one FlexRay Cycle") # Udp Nm: (NmPnResetTime + ComMPncPrepareSleepTimer) < UdpNmTimeoutTime	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 63]	
Default value	–	
Pre-Compile only	true	

⌋

[ECUC_ComM_00839] Definition of EcucBooleanParamDef ComMPncSupport ⌈

Parameter Name	ComMPncSupport
Parent Container	ComMGeneral
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
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
Pre-Compile only	true

⌋

[ECUC_ComM_00558] Definition of EcucBooleanParamDef ComMResetAfterForcingNoComm ⌈

Parameter Name	ComMResetAfterForcingNoComm
Parent Container	ComMGeneral
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_ComM_00897] Definition of EcucBooleanParamDef ComMSynchronized PncShutdownEnabled ⌈

Parameter Name	ComMSynchronizedPncShutdownEnabled
Parent Container	ComMGeneral
Upstream mapping defined	–
Description	Enables or disables support of synchronized PNC shutdown. FALSE: synchronized PNC shutdown is disabled TRUE: synchronized PNC shutdown is enabled NOTE: This is only possible for ECU that has the role of an top-level PNC coordinator or intermediate PNC within the PNC network Dependency: Parameter can only be set to TRUE if ComMPncGatewayEnabled is set to TRUE.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00695] Definition of EcucBooleanParamDef ComMSynchronous WakeUp ⌈

Parameter Name	ComMSynchronousWakeUp
Parent Container	ComMGeneral
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
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
Pre-Compile only	true

⌋

[ECUC_ComM_00557] Definition of EcucFloatParamDef ComMTMinFullCom ModeDuration ⌈

Parameter Name	ComMTMinFullComModeDuration
Parent Container	ComMGeneral
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_ComM_00559] Definition of EcucBooleanParamDef ComMWakeupInhibitionEnabled ⌈

Parameter Name	ComMWakeupInhibitionEnabled
Parent Container	ComMGeneral
Upstream mapping defined	–
Description	true if wake up inhibition functionality enabled. true: Enabled false: Disabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00783] Definition of EcucReferenceDef ComMGlobalNvMBlockDescriptor ⌈

Parameter Name	ComMGlobalNvMBlockDescriptor
Parent Container	ComMGeneral
Upstream mapping defined	–
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. Dependency: Derived from NvM configuration
Multiplicity	0..1
Type	Symbolic name reference to NvMBlockDescriptor
Pre-Compile only	true

⌋

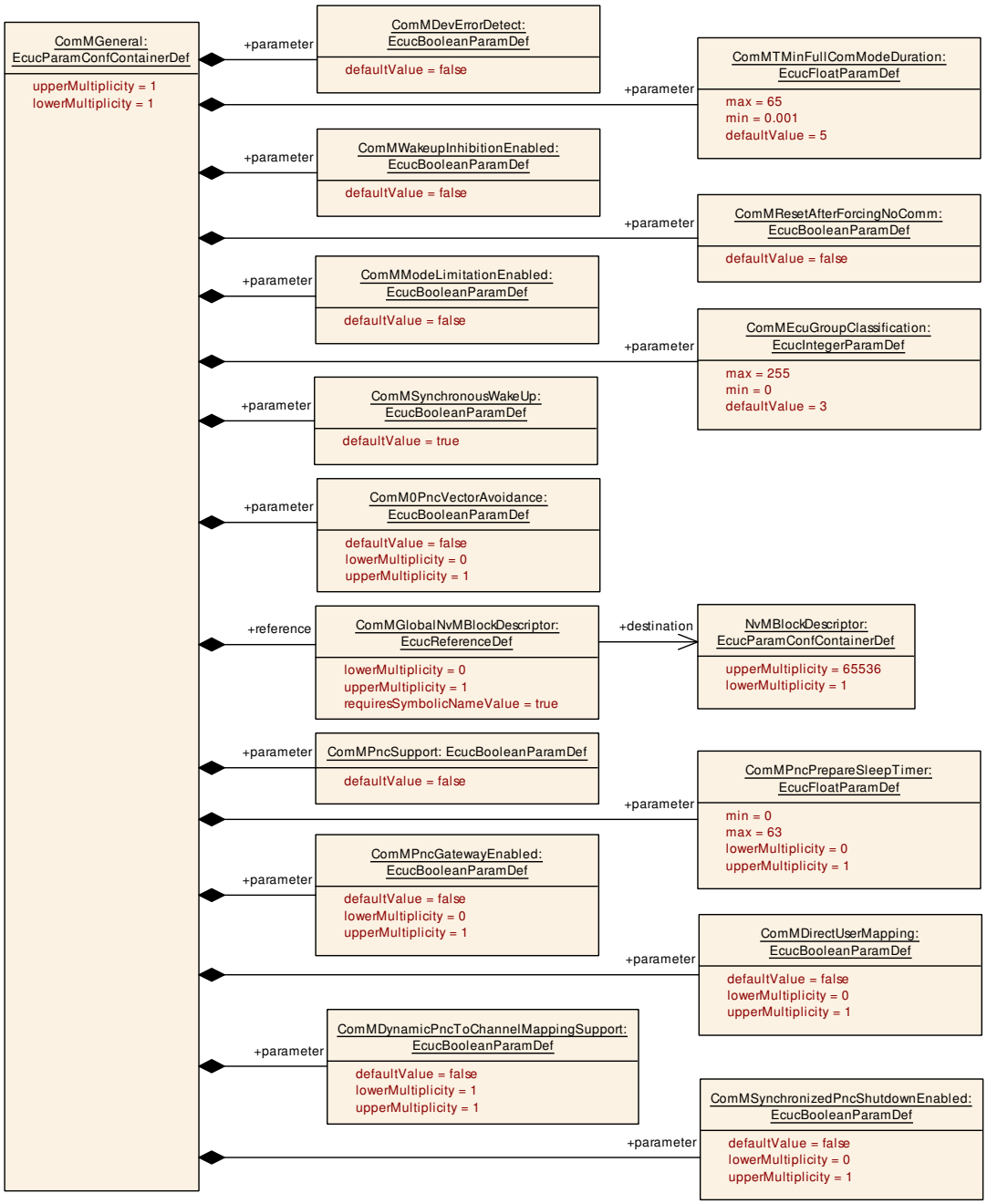


Figure 10.2: Configuration ComMGeneral

10.2.3 ComMConfigSet

[ECUC_ComM_00879] Definition of EcucParamConfContainerDef ComMConfig Set ⌈

Container Name	ComMConfigSet
Parent Container	ComM
Upstream mapping defined	–
Description	This container contains the configuration parameters and sub containers of the AUTOSAR ComM module.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	Multiplicity	ECUC ID
ComMEcuGroupClassificationNvMStorage	1	[ECUC_ComM_00902]

Included Containers		
Container Name	Multiplicity	Description
ComMChannel	1..256	This container contains the configuration (parameters) of the bus channel(s). The channel parameters shall be harmonized within the whole communication stack.
ComMPnc	0..504	This container contains the configuration of the partial network cluster (PNC).
ComMUser	0..65535	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.

⌋

[ECUC_ComM_00902] Definition of EcucBooleanParamDef ComMEcuGroupClassificationNvMStorage ⌈

Parameter Name	ComMEcuGroupClassificationNvMStorage
Parent Container	ComMConfigSet
Upstream mapping defined	–
Description	If this parameter is set to "true", the ComMEcuGroupClassification state shall be stored in the block pointed to by ComMGlobalNvMBlockDescriptor. Dependency: If the parameter is set to true, a valid NvM block reference must be given in the ComMGlobalNvMBlockDescriptor pointing to a NvM block with sufficient space.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

10.2.4 ComMUser

[ECUC_ComM_00653] Definition of EcucParamConfContainerDef ComMUser ⌈

Container Name	ComMUser
Parent Container	ComMConfigSet
Upstream mapping defined	[4, CP_TPS_BSWModuleDescriptionTemplate]
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.
Multiplicity	0..65535
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMUserIdentifier	1	[ECUC_ComM_00654]
ComMUserEcucPartitionRef	0..1	[ECUC_ComM_00786]

No Included Containers

⌋

[ECUC_ComM_00654] Definition of EcucIntegerParamDef ComMUserIdentifier ⌈

Parameter Name	ComMUserIdentifier	
Parent Container	ComMUser	
Upstream mapping defined	–	
Description	An identifier that is needed to refer to a user in the system which is designated to request Communication Modes. ImplementationType: ComM_UserHandleType 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.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	0 .. 65534	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_ComM_00786] Definition of EcucReferenceDef ComMUserEcucPartition Ref ⌈

Parameter Name	ComMUserEcucPartitionRef
Parent Container	ComMUser
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

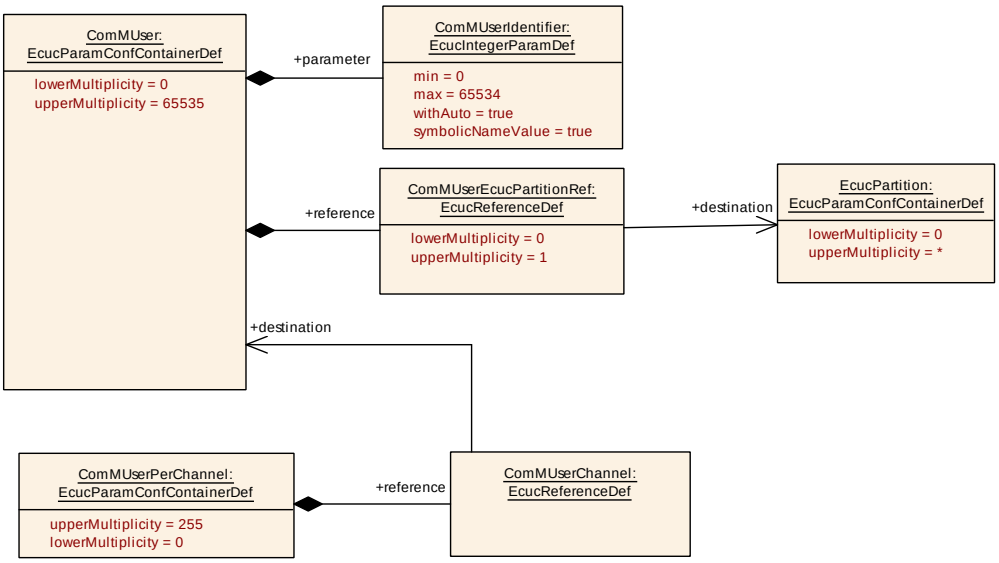


Figure 10.3: Configuration ComMUser

10.2.5 ComMChannel

[ECUC_ComM_00565] Definition of EcucParamConfContainerDef ComMChannel

⌈

Container Name	ComMChannel
Parent Container	ComMConfigSet
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	This container contains the configuration (parameters) of the bus channel(s). The channel parameters shall be harmonized within the whole communication stack.
Multiplicity	1..256
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMBusType	1	[ECUC_ComM_00567]
ComMCDDBusPrefix	0..1	[ECUC_ComM_00888]
ComMChannelId	1	[ECUC_ComM_00635]
ComMDynamicPncToChannelMappingEnabled	0..1	[ECUC_ComM_00896]
ComMFullCommRequestNotificationEnabled	1	[ECUC_ComM_00787]
ComMMainFunctionPeriod	1	[ECUC_ComM_00556]
ComMNoCom	1	[ECUC_ComM_00571]
ComMNoWakeup	1	[ECUC_ComM_00569]
ComMNoWakeUpInhibitionNvmStorage	1	[ECUC_ComM_00789]
ComMPncEnabled	0..1	[ECUC_ComM_00901]
ComMPncGatewayType	0..1	[ECUC_ComM_00842]

▽

△

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMWakeupSleepRequestEnabled	0..1	[ECUC_ComM_00898]
ComMChannelPartitionRef	0..1	[ECUC_ComM_00894]
ComMManageReference	0..*	[ECUC_ComM_00893]

Included Containers		
Container Name	**Multiplicity**	**Description**
ComMNetworkManagement	1	This container contains the configuration parameters of the networkmanagement.
ComMUserPerChannel	0..255	This container contains a list of identifiers that are needed to refer to a user in the system which is linked to a channel.

⌋

[ECUC_ComM_00567] Definition of EcucEnumerationParamDef ComMBusType

⌈

Parameter Name	ComMBusType	
Parent Container	ComMChannel	
Upstream mapping defined	[11, CP_TPS_SystemTemplate]	
Description	Identifies the bus type of the channel.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	COMM_BUS_TYPE_CAN	CAN bus
	COMM_BUS_TYPE_CDD	Cdd bus (complex device driver)
	COMM_BUS_TYPE_ETH	Ethernet bus
	COMM_BUS_TYPE_FR	FlexRay bus
	COMM_BUS_TYPE_INTERNAL	Internal communication (no bus interface)
	COMM_BUS_TYPE_LIN	Lin bus
Pre-Compile only	true	

⌋

[ECUC_ComM_00888] Definition of EcucStringParamDef ComMCDDBusPrefix ⌈

Parameter Name	ComMCDDBusPrefix
Parent Container	ComMChannel
Upstream mapping defined	–
Description	Prefix to be used for API calls to CDD. Dependency: Only applicable if ComMBusType equals COMM_BUS_TYPE_CDD.
Multiplicity	0..1
Type	EcucStringParamDef
Default value	–
Regular Expression	–
Pre-Compile only	true

⌋

[ECUC_ComM_00635] Definition of EcucIntegerParamDef ComMChannelId ⌈

Parameter Name	ComMChannelId
Parent Container	ComMChannel
Upstream mapping defined	–
Description	Channel identification number of the corresponding channel. Dependency: Shall be harmonized with channel IDs of networkmanagement and the bus interfaces.
Multiplicity	1
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)
Range	0 .. 255
Default value	–
With Auto	true
Pre-Compile only	true

⌋

[ECUC_ComM_00896] Definition of EcucBooleanParamDef ComMDynamicPncToChannelMappingEnabled ⌈

Parameter Name	ComMDynamicPncToChannelMappingEnabled
Parent Container	ComMChannel
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	Channel-specific parameter to enable the dynamic PNC-to-channel-mapping feature. False: Dynamic PNC-to-channel-mapping is disabled True: Dynamic PNC-to-channel-mapping is enabled Dependency: Shall only be TRUE if ComMDynamicPncToChannelMappingSupport is TRUE and ComMNmVariant is set to FULL for this ComMChannel.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00787] Definition of EcucBooleanParamDef ComMFullCommRequestNotificationEnabled ⌈

Parameter Name	ComMFullCommRequestNotificationEnabled
Parent Container	ComMChannel
Upstream mapping defined	–
Description	Defines if the optional SenderReceiver Port of Interface ComM_CurrentChannel Request will be provided for this channel. True means enabled. False means disabled Dependency: Shall be stored none volatile (value must be kept during a reset).
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00556] Definition of EcucFloatParamDef ComMMainFunctionPeriod ⌈

Parameter Name	ComMMainFunctionPeriod	
Parent Container	ComMChannel	
Upstream mapping defined	–	
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 .. INF[	
Default value	0.02	
Pre-Compile only	true	

⌋

[ECUC_ComM_00571] Definition of EcucBooleanParamDef ComMNoCom ⌈

Parameter Name	ComMNoCom
Parent Container	ComMChannel
Upstream mapping defined	–
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_Limit ChannelToNoComMode(). Dependency: ComMModeLimitationEnabled
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00569] Definition of EcucBooleanParamDef ComMNoWakeup ⌈

Parameter Name	ComMNoWakeup
Parent Container	ComMChannel
Upstream mapping defined	–
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
Pre-Compile only	true

⌋

[ECUC_ComM_00789] Definition of EcucBooleanParamDef ComMNoWakeUpInhibitionNvmStorage ⌈

Parameter Name	ComMNoWakeUpInhibitionNvmStorage
Parent Container	ComMChannel
Upstream mapping defined	–
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 ComMGlobal NvmBlockDescriptor. 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.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	–
Pre-Compile only	true

⌋

[ECUC_ComM_00901] Definition of EcucBooleanParamDef ComMPncEnabled ⌈

Parameter Name	ComMPncEnabled	
Parent Container	ComMChannel	
Upstream mapping defined	–	
Description	Defines whether the partial networking is enabled for this ComMChannel. true: Enabled false: Disabled Dependency: ComMPncSupport	
Multiplicity	0..1	
Type	EcucBooleanParamDef	
Default value	false	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_ComM_00842] Definition of EcucEnumerationParamDef ComMPncGatewayType ⌈

Parameter Name	ComMPncGatewayType	
Parent Container	ComMChannel	
Upstream mapping defined	[11, CP_TPS_SystemTemplate]	
Description	Identifies the Partial Network Gateway behaviour of a ComMChannel. Dependency: Parameter shall not be used for managed channel (shall neither be set to COMM_GATEWAY_TYPE_ACTIVE nor COMM_GATEWAY_TYPE_PASSIVE).	
Multiplicity	0..1	
Type	EcucEnumerationParamDef	
Range	COMM_GATEWAY_TYPE_ACTIVE	Active PNC gateway
	COMM_GATEWAY_TYPE_PASSIVE	Passive PNC gateway

▽

△

Default value	COMM_GATEWAY_TYPE_ACTIVE
Pre-Compile only	true

⌋

[ECUC_ComM_00898] Definition of EcucBooleanParamDef ComMWakeupSleepRequestEnabled ⌈

Parameter Name	ComMWakeupSleepRequestEnabled
Parent Container	ComMChannel
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	Used for communication channels where the corresponding hardware support wake-up and/or sleep request capability on the network, e.g. OA TC10 compatible PHYs for Ethernet. Dependency: Only applicable if ComMBusType equals COMM_BUS_TYPE_ETH and the used Ethernet hardware (e.g. PHY, Ethernet switch) is compatible with the OA TC10 specification.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00894] Definition of EcucReferenceDef ComMChannelPartitionRef ⌈

Parameter Name	ComMChannelPartitionRef
Parent Container	ComMChannel
Upstream mapping defined	–
Description	Reference to EcucPartition, where the according ComMChannel is assigned to.
Multiplicity	0..1
Type	Reference to EcucPartition
Pre-Compile only	true

⌋

[ECUC_ComM_00893] Definition of EcucReferenceDef ComMManageReference ⌈

Parameter Name	ComMManageReference
Parent Container	ComMChannel
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	Represents the reference between a ComMChannel with role managing channel and a ComMChannel with role managed channel.
Multiplicity	0..*
Type	Reference to ComMChannel
Pre-Compile only	true

⌋

[SWS_ComM_00690]

Upstream requirements: SRS_BSW_00167

⌈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.⌋

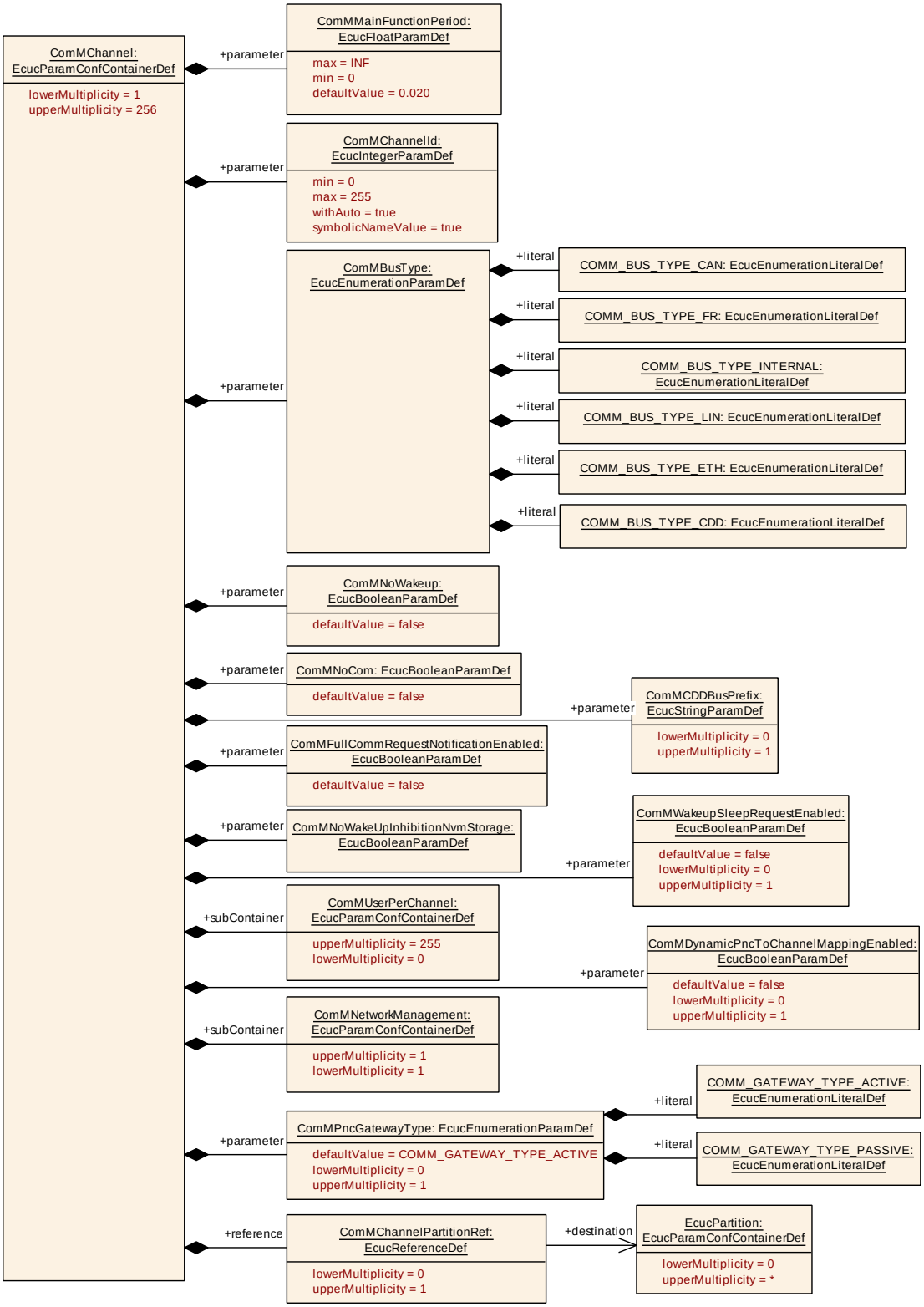


Figure 10.4: Configuration ComMChannel

10.2.6 ComMNetworkManagement

[ECUC_ComM_00607] Definition of EcucParamConfContainerDef ComMNetwork Management ⌈

Container Name	ComMNetworkManagement
Parent Container	ComMChannel
Upstream mapping defined	–
Description	This container contains the configuration parameters of the networkmanagement.
Multiplicity	1
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMNmLightTimeout	0..1	[ECUC_ComM_00606]
ComMNmVariant	1	[ECUC_ComM_00568]
ComMPncNmRequest	1	[ECUC_ComM_00886]

No Included Containers

⌋

[ECUC_ComM_00606] Definition of EcucFloatParamDef ComMNmLightTimeout ⌈

Parameter Name	ComMNmLightTimeout	
Parent Container	ComMNetworkManagement	
Upstream mapping defined	[11, CP_TPS_SystemTemplate]	
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. Dependency: Only used if ComMNmVariant is configured as ComMLight	
Multiplicity	0..1	
Type	EcucFloatParamDef	
Range	[0 .. 255]	
Default value	10	
Pre-Compile only	true	

⌋

[ECUC_ComM_00568] Definition of EcucEnumerationParamDef ComMNmVariant ⌈

Parameter Name	ComMNmVariant
Parent Container	ComMNetworkManagement
Upstream mapping defined	[11, CP_TPS_SystemTemplate]

▽

△

Description	Defines the functionality of the networkmanagement. Shall be harmonized with NM configuration. Dependency: ComMNmVariant shall be NONE if ComMBusType = COMM_BUS_TYPE_INTERNAL. ComMNmVariant shall be LIGHT for managed channels. ComMNm Variant shall be FULL for managing channels.	
Multiplicity	1	
Type	EcucEnumerationParamDef	
Range	FULL	AUTOSAR NM is available (default).
	LIGHT	No AUTOSAR NM is available, but functionality to shut down a channel.
	NONE	No NM available
	PASSIVE	AUTOSAR NM running in passive mode available.
	SLAVE_ACTIVE	No NM is available. This is used for e.g. LIN slaves.
	SLAVE_PASSIVE	No NM is available. This used for e.g. Ethernet communication channels with OA TC10 compliant hardware.
Default value	FULL	
Pre-Compile only	true	

⌋

[ECUC_ComM_00886] Definition of EcucBooleanParamDef ComMPncNmRequest ⌈

Parameter Name	ComMPncNmRequest
Parent Container	ComMNetworkManagement
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	If this parameter equals true, then Nm shall be requested again by calling Nm_Network Request under either the following conditions: • every time a FULL Communication is requested due to a change in the PNC state machine to COMM_PNC_REQUESTED • if a shutdown for a PNC coincides with a PNC request of the same PNC Dependency: It shall only be possible to set ComMPncNmRequest to TRUE, if Com MNmVariant is FULL.
Multiplicity	1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

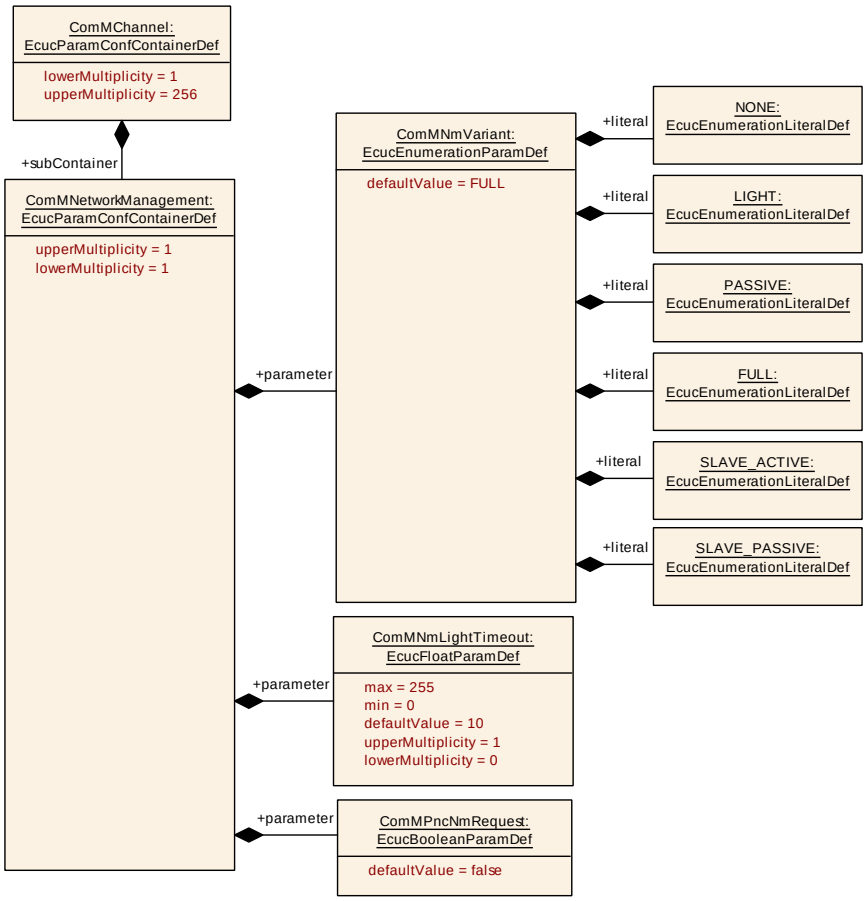


Figure 10.5: Configuration ComMNetworkManagement

10.2.7 ComMUserPerChannel

[ECUC_ComM_00657] Definition of EcucParamConfContainerDef ComMUserPerChannel ⌈

Container Name	ComMUserPerChannel
Parent Container	ComMChannel
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
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.
Multiplicity	0..255
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMUserChannel	1	[ECUC_ComM_00658]

No Included Containers

⌋

[ECUC_ComM_00658] Definition of EcucReferenceDef ComMUserChannel ⌈

Parameter Name	ComMUserChannel
Parent Container	ComMUserPerChannel
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	Reference to the ComMUser that corresponds to this channel user. ImplementationType: COMM_UserHandleType
Multiplicity	1
Type	Reference to ComMUser
Pre-Compile only	true

⌋

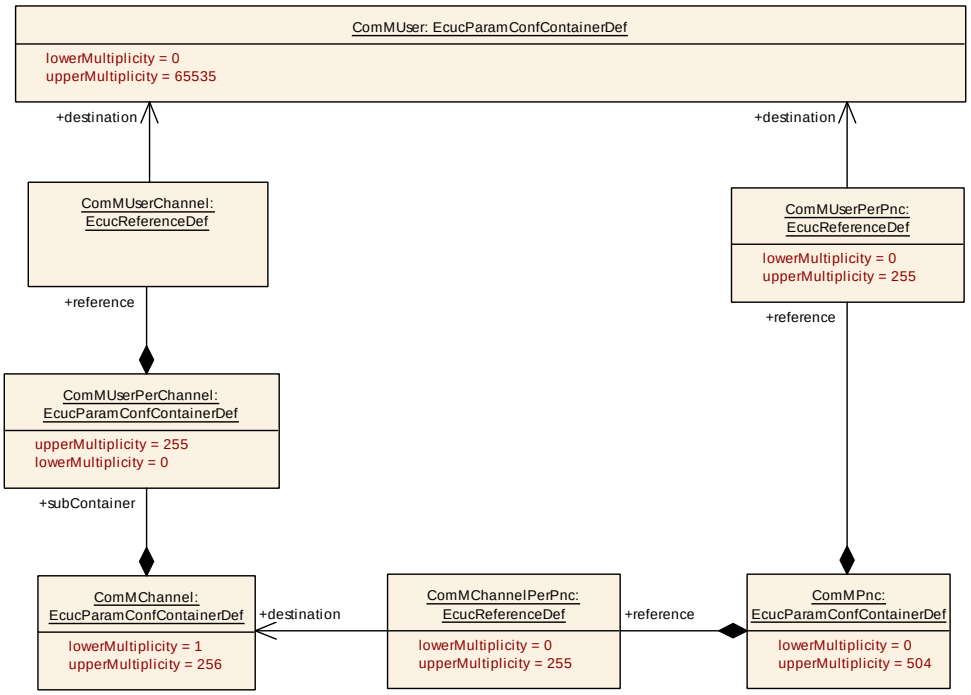


Figure 10.6: Configuration ComMUserPerChannel and ComUserPerPNC

10.2.8 ComMPnc

[ECUC_ComM_00843] Definition of EcucParamConfContainerDef ComMPnc ⌈

Container Name	ComMPnc
Parent Container	ComMConfigSet
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	This container contains the configuration of the partial network cluster (PNC).
Multiplicity	0..504
Pre-Compile only	true
Configuration Parameters	

Included Parameters		
Parameter Name	**Multiplicity**	**ECUC ID**
ComMPncId	1	[ECUC_ComM_00874]
ComMPncWakeupSleepRequestEnabled	0..1	[ECUC_ComM_00899]
ComMChannelPerPnc	0..255	[ECUC_ComM_00880]
ComMChannelPerTxOnlyPnc	0..255	[ECUC_ComM_00900]
ComMPncEthIfSwitchPortGroupRef	0..1	[ECUC_ComM_00891]
ComMUserPerPnc	0..255	[ECUC_ComM_00876]

No Included Containers

⌋

[ECUC_ComM_00874] Definition of EcucIntegerParamDef ComMPncId ⌈

Parameter Name	ComMPncId	
Parent Container	ComMPnc	
Upstream mapping defined	[11, CP_TPS_SystemTemplate]	
Description	Partial network cluster identification number.	
Multiplicity	1	
Type	EcucIntegerParamDef (Symbolic Name generated for this parameter)	
Range	8 .. 511	
Default value	–	
With Auto	true	
Pre-Compile only	true	

⌋

[ECUC_ComM_00899] Definition of EcucBooleanParamDef ComMPncWakeupSleepRequestEnabled ⌈

Parameter Name	ComMPncWakeupSleepRequestEnabled
Parent Container	ComMPnc
Upstream mapping defined	[11, CP_TPS_SystemTemplate]
Description	Used for PNCs where a requested PNC shall report an active communication request towards the BswM. The BswM forward the active communication request to the lower layer communication channels where the used hardware support wake-up and/or sleep request capability on the network, e.g. OA TC10 compatible PHYs for Ethernet. This is used e.g. for Ethernet Switch port group switching.
Multiplicity	0..1
Type	EcucBooleanParamDef
Default value	false
Pre-Compile only	true

⌋

[ECUC_ComM_00880] Definition of EcucReferenceDef ComMChannelPerPnc ⌈

Parameter Name	ComMChannelPerPnc
Parent Container	ComMPnc
Upstream mapping defined	[11, CP_TPS_SystemTemplate]

▽

△

Description	Reference to the ComMChannel that is required for this PNC. ImplementationType: NetworkHandleType	
Multiplicity	0..255	
Type	Reference to ComMChannel	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_ComM_00900] Definition of EcucReferenceDef ComMChannelPerTxOnlyPnc ⌈

Parameter Name	ComMChannelPerTxOnlyPnc	
Parent Container	ComMPnc	
Upstream mapping defined	–	
Description	Reference to the ComMChannel that is required for this PNC. This PNC is considered to be only transmitted on this channel as internal PNC request. ImplementationType: NetworkHandleType	
Multiplicity	0..255	
Type	Reference to ComMChannel	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_ComM_00891] Definition of EcucReferenceDef ComMPncEthIfSwitchPortGroupRef ⌈

Parameter Name	ComMPncEthIfSwitchPortGroupRef	
Parent Container	ComMPnc	
Upstream mapping defined	[11, CP_TPS_SystemTemplate]	
Description	Reference to the PortGroups that correspond to this PNC. Note: This is only for documentation.	
Multiplicity	0..1	
Type	Symbolic name reference to EthIfSwitchPortGroup	
Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

[ECUC_ComM_00876] Definition of EcucReferenceDef ComMUserPerPnc ⌈

Parameter Name	ComMUserPerPnc
Parent Container	ComMPnc
Upstream mapping defined	–
Description	Reference to the ComMUsers that correspond to this PNC. ImplementationType: COMM_UserHandleType
Multiplicity	0..255
Type	Reference to ComMUser

▽

△

Post-Build Multiplicity	Loadable	Selectable
Post-Build Value	Loadable	Selectable

⌋

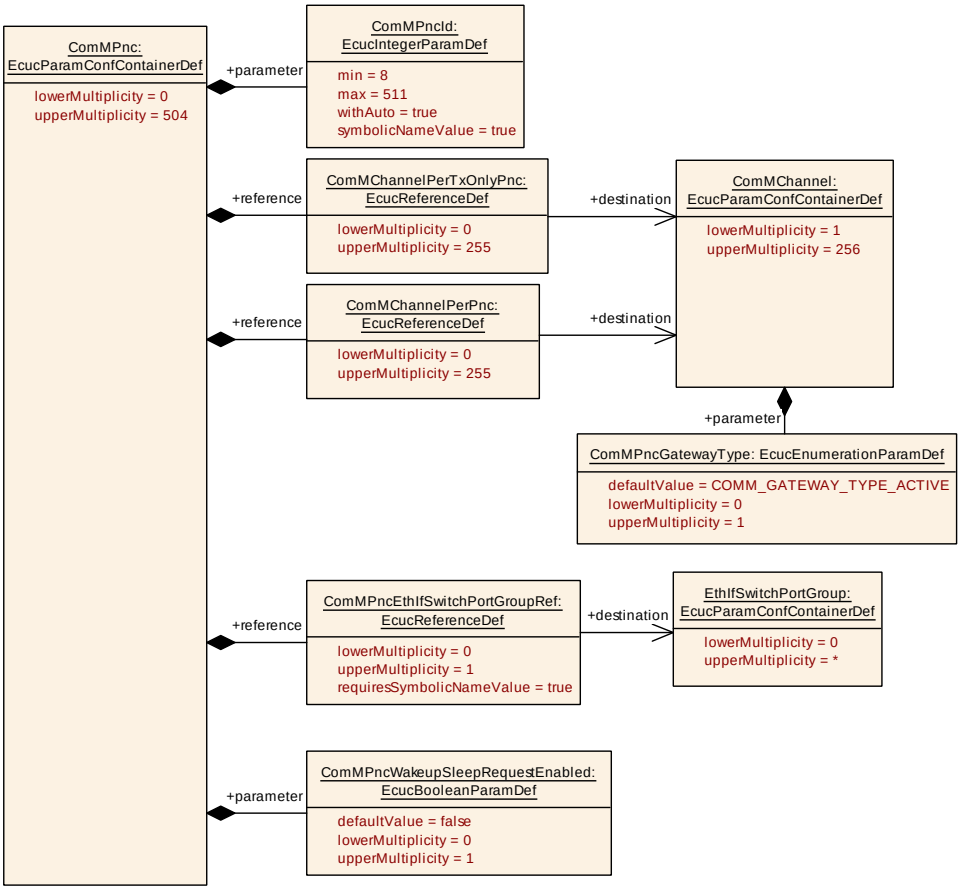


Figure 10.7: Configuration ComMPnc

10.3 Published Information

[SWS_ComM_00418]

Upstream requirements: SRS_BSW_00004

⌈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).⌋

A Not applicable requirements

[SWS_ComM_NA_00499]

Upstream requirements: SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00344, SRS_BSW_00375, SRS_BSW_00383, SRS_BSW_00384, SRS_BSW_00388, SRS_BSW_00389, SRS_BSW_00390, SRS_BSW_00392, SRS_BSW_00393, SRS_BSW_00395, SRS_BSW_00396, SRS_BSW_00398, SRS_BSW_00399, SRS_BSW_00403, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00419, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00452, SRS_BSW_00458, SRS_BSW_00461, SRS_BSW_00469, SRS_BSW_00470, SRS_BSW_00471, SRS_BSW_00472, SRS_BSW_00478, SRS_BSW_00490, SRS_BSW_00491, SRS_ModeMgm_09244

⌈These requirements are not applicable to this specification.⌋

B Change history of AUTOSAR traceable items

Please note that the lists in this chapter also include traceable items that have been removed from the specification in a later version. These items do not appear as hyperlinks in the document.

B.1 Traceable item history of this document according to AUTOSAR Release R22-11

No change history due to document migration.

B.2 Traceable item history of this document according to AUTOSAR Release R23-11

B.2.1 Added Specification Items in R23-11

[SWS_ComM_00051] [SWS_ComM_00066] [SWS_ComM_00069] [SWS_ComM_00071] [SWS_ComM_00073] [SWS_ComM_00079] [SWS_ComM_00083] [SWS_ComM_00084] [SWS_ComM_00085] [SWS_ComM_00091] [SWS_ComM_00092] [SWS_ComM_00103] [SWS_ComM_00108] [SWS_ComM_00110] [SWS_ComM_00124] [SWS_ComM_00133] [SWS_ComM_00138] [SWS_ComM_00140] [SWS_ComM_00141] [SWS_ComM_00142] [SWS_ComM_00143] [SWS_ComM_00146] [SWS_ComM_00147] [SWS_ComM_00151] [SWS_ComM_00156] [SWS_ComM_00157] [SWS_ComM_00162] [SWS_ComM_00163] [SWS_ComM_00176] [SWS_ComM_00182] [SWS_ComM_00191] [SWS_ComM_00215] [SWS_ComM_00218] [SWS_ComM_00219] [SWS_ComM_00224] [SWS_ComM_00234] [SWS_ComM_00242] [SWS_ComM_00261] [SWS_ComM_00266] [SWS_ComM_00275] [SWS_ComM_00288] [SWS_ComM_00295] [SWS_ComM_00296] [SWS_ComM_00301] [SWS_ComM_00302] [SWS_ComM_00303] [SWS_ComM_00313] [SWS_ComM_00322] [SWS_ComM_00355] [SWS_ComM_00370] [SWS_ComM_00374] [SWS_ComM_00383] [SWS_ComM_00390] [SWS_ComM_00391] [SWS_ComM_00392] [SWS_ComM_00402] [SWS_ComM_00418] [SWS_ComM_00419] [SWS_ComM_00429] [SWS_ComM_00459] [SWS_ComM_00464] [SWS_ComM_00470] [SWS_ComM_00472] [SWS_ComM_00485] [SWS_ComM_00488] [SWS_ComM_00500] [SWS_ComM_00552] [SWS_ComM_00582] [SWS_ComM_00583] [SWS_ComM_00599] [SWS_ComM_00602] [SWS_ComM_00610] [SWS_ComM_00612] [SWS_ComM_00619] [SWS_ComM_00620] [SWS_ComM_00625] [SWS_ComM_00637] [SWS_ComM_00662] [SWS_ComM_00663] [SWS_ComM_00664] [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_00686] [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_00818] [SWS_ComM_00820] [SWS_ComM_00825] [SWS_ComM_00826] [SWS_ComM_00827] [SWS_ComM_00828] [SWS_ComM_00829] [SWS_ComM_00839] [SWS_ComM_00840] [SWS_ComM_00842] [SWS_ComM_00845] [SWS_ComM_00846] [SWS_ComM_00847] [SWS_ComM_00848] [SWS_ComM_00858] [SWS_ComM_00861] [SWS_ComM_00863] [SWS_ComM_00864] [SWS_ComM_00865] [SWS_ComM_00866] [SWS_ComM_00867] [SWS_ComM_00868] [SWS_ComM_00869] [SWS_ComM_00870] [SWS_ComM_00871] [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_00893] [SWS_ComM_00894] [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_00915] [SWS_ComM_00920] [SWS_ComM_00924] [SWS_ComM_00926] [SWS_ComM_00929] [SWS_ComM_00932] [SWS_ComM_00938] [SWS_ComM_00940] [SWS_ComM_00947] [SWS_ComM_00948] [SWS_ComM_00950] [SWS_ComM_00952] [SWS_ComM_00953] [SWS_ComM_00957] [SWS_ComM_00958] [SWS_ComM_00962] [SWS_ComM_00963] [SWS_ComM_00972] [SWS_ComM_00976] [SWS_ComM_00978] [SWS_ComM_00979] [SWS_ComM_00980] [SWS_ComM_00982] [SWS_ComM_00987] [SWS_ComM_00990] [SWS_ComM_00991] [SWS_ComM_00994] [SWS_ComM_00995] [SWS_ComM_00996] [SWS_ComM_01000] [SWS_ComM_01001] [SWS_ComM_01005] [SWS_ComM_01006] [SWS_ComM_01007] [SWS_ComM_01008] [SWS_ComM_01009] [SWS_ComM_01010] [SWS_ComM_01011] [SWS_ComM_01012] [SWS_ComM_01014] [SWS_ComM_01015] [SWS_ComM_01016] [SWS_ComM_01017] [SWS_ComM_01018] [SWS_ComM_01019] [SWS_ComM_01020] [SWS_ComM_01022] [SWS_ComM_01023] [SWS_ComM_01024] [SWS_ComM_01025] [SWS_ComM_01026] [SWS_ComM_01028] [SWS_ComM_01029] [SWS_ComM_01034] [SWS_ComM_01035] [SWS_ComM_01036] [SWS_ComM_01037] [SWS_ComM_01038] [SWS_ComM_01039] [SWS_ComM_01040] [SWS_ComM_01041] [SWS_ComM_01042] [SWS_ComM_01043] [SWS_ComM_01044] [SWS_ComM_01045] [SWS_ComM_01046] [SWS_ComM_01047] [SWS_ComM_01048] [SWS_ComM_01049] [SWS_ComM_01056] [SWS_ComM_01057] [SWS_ComM_01058] [SWS_ComM_01059] [SWS_ComM_01060] [SWS_ComM_01061] [SWS_ComM_01062] [SWS_ComM_01063] [SWS_ComM_01064] [SWS_ComM_01065] [SWS_ComM_01066] [SWS_ComM_01067] [SWS_ComM_01068] [SWS_ComM_01069] [SWS_ComM_01070] [SWS_ComM_01071] [SWS_ComM_01072] [SWS_ComM_01073]

[SWS_ComM_01074] [SWS_ComM_01075] [SWS_ComM_01076] [SWS_ComM_01077] [SWS_ComM_01078] [SWS_ComM_01079] [SWS_ComM_01080] [SWS_ComM_01081] [SWS_ComM_01082] [SWS_ComM_01083] [SWS_ComM_01084] [SWS_ComM_01085] [SWS_ComM_01086] [SWS_ComM_01087] [SWS_ComM_01088] [SWS_ComM_01089] [SWS_ComM_01090] [SWS_ComM_01091] [SWS_ComM_01092] [SWS_ComM_01093] [SWS_ComM_01094] [SWS_ComM_01095] [SWS_ComM_01096] [SWS_ComM_01097] [SWS_ComM_01098] [SWS_ComM_01099] [SWS_ComM_01100] [SWS_ComM_01101] [SWS_ComM_91000] [SWS_ComM_91001] [SWS_ComM_91002] [SWS_ComM_91013] [SWS_ComM_91015] [SWS_ComM_91017] [SWS_ComM_91019] [SWS_ComM_91021] [SWS_ComM_91024] [SWS_ComM_91026] [SWS_ComM_91027] [SWS_ComM_91028] [SWS_ComM_91029] [SWS_ComM_91030] [SWS_ComM_91102] [SWS_ComM_91107] [SWS_ComM_91108] [SWS_ComM_91109] [SWS_ComM_91110] [SWS_ComM_NA_00499]

B.2.2 Changed Specification Items in R23-11

none

B.2.3 Deleted Specification Items in R23-11

none

B.2.4 Added Constraints in R23-11

Number	Heading
[SWS_ComM_CONSTR_00001]	
[SWS_ComM_CONSTR_00002]	
[SWS_ComM_CONSTR_00003]	
[SWS_ComM_CONSTR_00004]	
[SWS_ComM_CONSTR_00005]	

▽

△

Number	Heading
[SWS_ComM_CONSTR_00006]	

Table B.1: Added Constraints in R23-11

B.2.5 Changed Constraints in R23-11

none

B.2.6 Deleted Constraints in R23-11

none

B.3 Traceable item history of this document according to AUTOSAR Release R24-11

B.3.1 Added Specification Items in R24-11

none

B.3.2 Changed Specification Items in R24-11

Number	Heading
[ECUC_ComM_00843]	Definition of EcucParamConfContainerDef ComMPnc
[ECUC_ComM_00891]	Definition of EcucReferenceDef ComMPncEthIfSwitchPortGroupRef
[SWS_ComM_00820]	Definition of imported datatypes of module ComM
[SWS_ComM_01000]	Definition of ClientServerInterface ComM_UserRequest
[SWS_ComM_01049]	
[SWS_ComM_01082]	
[SWS_ComM_91102]	Definition of ClientServerInterface ComM_PncToChannelMapping
[SWS_ComM_91108]	Definition of ClientServerInterface ComM_DynamicPncToChannelMapping

Table B.2: Changed Specification Items in R24-11

B.3.3 Deleted Specification Items in R24-11

none

B.3.4 Added Constraints in R24-11

none

B.3.5 Changed Constraints in R24-11

none

B.3.6 Deleted Constraints in R24-11

none

B.4 Traceable item history of this document according to AUTOSAR Release R25-11

B.4.1 Added Specification Items in R25-11

Number	Heading
[ECUC_ComM_00901]	Definition of EcucBooleanParamDef ComMPncEnabled
[ECUC_ComM_00902]	Definition of EcucBooleanParamDef ComMEcuGroupClassificationNvMStorage
[SWS_ComM_01102]	Call of `ComM_SetECUGroupClassification`
[SWS_ComM_01103]	Call of communication inhibition function
[SWS_ComM_01104]	Non volatile storage of `ComMEcuGroupClassification`
[SWS_ComM_01105]	Default values of `ComMEcuGroupClassification`
[SWS_ComM_01106]	Default values of `ComMNoWakeup`
[SWS_ComM_01107]	Initialization value of ComM inhibition status "limit to `COMM_NO_COMMUNICATION` mode"

Table B.3: Added Specification Items in R25-11

B.4.2 Changed Specification Items in R25-11

Number	Heading
[ECUC_ComM_00563]	Definition of EcucIntegerParamDef ComMEcuGroupClassification
[ECUC_ComM_00565]	Definition of EcucParamConfContainerDef ComMChannel
[ECUC_ComM_00569]	Definition of EcucBooleanParamDef ComMNoWakeup
[ECUC_ComM_00607]	Definition of EcucParamConfContainerDef ComMNetworkManagement
[ECUC_ComM_00653]	Definition of EcucParamConfContainerDef ComMUser
[ECUC_ComM_00657]	Definition of EcucParamConfContainerDef ComMUserPerChannel
[ECUC_ComM_00843]	Definition of EcucParamConfContainerDef ComMPnc
[ECUC_ComM_00879]	Definition of EcucParamConfContainerDef ComMConfigSet
[SWS_ComM_00140]	Storage of Inhibit counter for rejected mode requests
[SWS_ComM_00157]	Non volatile storage of `ComMNoWakeup` status
[SWS_ComM_00829]	Definition of optional interfaces requested by module ComM
[SWS_ComM_00864]	Read non-volatile parameters in `ComM_Init`
[SWS_ComM_00908]	
[SWS_ComM_00953]	Execute PNC functionality upfront to `ComMChannel` related actions
[SWS_ComM_01045]	Handling for a call of `ComM_PnLearningRequest` and PNC learning phase is active
[SWS_ComM_01046]	Handling for a call of `ComM_PnLearningRequest` and PNC learning phase is note active
[SWS_ComM_01048]	Handling for a call of `ComM_UpdatePncMembership` and PNC learning phase is active

Table B.4: Changed Specification Items in R25-11

B.4.3 Deleted Specification Items in R25-11

Number	Heading
[ECUC_ComM_00878]	Definition of EcucBooleanParamDef ComMPncEnabled

Table B.5: Deleted Specification Items in R25-11

B.4.4 Added Constraints in R25-11

Number	Heading
[SWS_ComM_CONSTR_00007]	Enabling or disabling of the PNC functionality at post-build time shall be exclusively allowed for CAN and FlexRay

Table B.6: Added Constraints in R25-11

B.4.5 Changed Constraints in R25-11

none

B.4.6 Deleted Constraints in R25-11

none

B.5 Traceable item history of this document according to AUTOSAR Release R26-11

B.5.1 Added Specification Items in R26-11

none

B.5.2 Changed Specification Items in R26-11

Number	Heading
[ECUC_ComM_00554]	Definition of EcucParamConfContainerDef ComMGeneral
[ECUC_ComM_00555]	Definition of EcucBooleanParamDef ComMDevErrorDetect
[ECUC_ComM_00556]	Definition of EcucFloatParamDef ComMMainFunctionPeriod
[ECUC_ComM_00557]	Definition of EcucFloatParamDef ComMTMinFullComModeDuration
[ECUC_ComM_00558]	Definition of EcucBooleanParamDef ComMResetAfterForcingNoComm
[ECUC_ComM_00559]	Definition of EcucBooleanParamDef ComMWakeupInhibitionEnabled

▽

△

Number	Heading
[ECUC_ComM_00560]	Definition of EcucBooleanParamDef ComMModeLimitationEnabled
[ECUC_ComM_00563]	Definition of EcucIntegerParamDef ComMEcuGroupClassification
[ECUC_ComM_00565]	Definition of EcucParamConfContainerDef ComMChannel
[ECUC_ComM_00567]	Definition of EcucEnumerationParamDef ComMBusType
[ECUC_ComM_00568]	Definition of EcucEnumerationParamDef ComMNmVariant
[ECUC_ComM_00569]	Definition of EcucBooleanParamDef ComMNoWakeup
[ECUC_ComM_00571]	Definition of EcucBooleanParamDef ComMNoCom
[ECUC_ComM_00606]	Definition of EcucFloatParamDef ComMNmLightTimeout
[ECUC_ComM_00607]	Definition of EcucParamConfContainerDef ComMNetworkManagement
[ECUC_ComM_00635]	Definition of EcucIntegerParamDef ComMChannelId
[ECUC_ComM_00653]	Definition of EcucParamConfContainerDef ComMUser
[ECUC_ComM_00654]	Definition of EcucIntegerParamDef ComMUserIdentifier
[ECUC_ComM_00657]	Definition of EcucParamConfContainerDef ComMUserPerChannel
[ECUC_ComM_00658]	Definition of EcucReferenceDef ComMUserChannel
[ECUC_ComM_00695]	Definition of EcucBooleanParamDef ComMSynchronousWakeUp
[ECUC_ComM_00783]	Definition of EcucReferenceDef ComMGlobalNvMBlockDescriptor
[ECUC_ComM_00786]	Definition of EcucReferenceDef ComMUserEcucPartitionRef
[ECUC_ComM_00787]	Definition of EcucBooleanParamDef ComMFullCommRequestNotification Enabled
[ECUC_ComM_00789]	Definition of EcucBooleanParamDef ComMNoWakeUpInhibitionNvmStorage
[ECUC_ComM_00839]	Definition of EcucBooleanParamDef ComMPncSupport
[ECUC_ComM_00840]	Definition of EcucBooleanParamDef ComMDirectUserMapping
[ECUC_ComM_00841]	Definition of EcucFloatParamDef ComMPncPrepareSleepTimer

▽

△

Number	Heading
[ECUC_ComM_00842]	Definition of EcucEnumerationParamDef ComMPncGatewayType
[ECUC_ComM_00843]	Definition of EcucParamConfContainerDef ComMPnc
[ECUC_ComM_00874]	Definition of EcucIntegerParamDef ComMPncId
[ECUC_ComM_00876]	Definition of EcucReferenceDef ComMUserPerPnc
[ECUC_ComM_00879]	Definition of EcucParamConfContainerDef ComMConfigSet
[ECUC_ComM_00880]	Definition of EcucReferenceDef ComMChannelPerPnc
[ECUC_ComM_00886]	Definition of EcucBooleanParamDef ComMPncNmRequest
[ECUC_ComM_00887]	Definition of EcucBooleanParamDef ComMPncGatewayEnabled
[ECUC_ComM_00888]	Definition of EcucStringParamDef ComMCDDBusPrefix
[ECUC_ComM_00890]	Definition of EcucModuleDef ComM
[ECUC_ComM_00891]	Definition of EcucReferenceDef ComMPncEthIfSwitchPortGroupRef
[ECUC_ComM_00892]	Definition of EcucBooleanParamDef ComM0PncVectorAvoidance
[ECUC_ComM_00893]	Definition of EcucReferenceDef ComMManageReference
[ECUC_ComM_00894]	Definition of EcucReferenceDef ComMChannelPartitionRef
[ECUC_ComM_00895]	Definition of EcucBooleanParamDef ComMDynamicPncToChannelMapping Support
[ECUC_ComM_00896]	Definition of EcucBooleanParamDef ComMDynamicPncToChannelMapping Enabled
[ECUC_ComM_00897]	Definition of EcucBooleanParamDef ComMSynchronizedPncShutdown Enabled
[ECUC_ComM_00898]	Definition of EcucBooleanParamDef ComMWakeupSleepRequestEnabled
[ECUC_ComM_00899]	Definition of EcucBooleanParamDef ComMPncWakeupSleepRequest Enabled
[ECUC_ComM_00900]	Definition of EcucReferenceDef ComMChannelPerTxOnlyPnc
[ECUC_ComM_00901]	Definition of EcucBooleanParamDef ComMPncEnabled
[ECUC_ComM_00902]	Definition of EcucBooleanParamDef ComMEcuGroupClassificationNv MStorage

▽

△

Number	Heading
[SWS_ComM_00110]	Definition of API function ComM_RequestComMode
[SWS_ComM_00612]	
[SWS_ComM_00820]	Definition of imported datatypes of module ComM
[SWS_ComM_00864]	Read non-volatile parameters in `ComM_Init`
[SWS_ComM_00982]	
[SWS_ComM_01000]	Definition of ClientServerInterface ComM_UserRequest
[SWS_ComM_01036]	
[SWS_ComM_01038]	
[SWS_ComM_01060]	
[SWS_ComM_01061]	
[SWS_ComM_01062]	
[SWS_ComM_01082]	
[SWS_ComM_01100]	
[SWS_ComM_01101]	
[SWS_ComM_91013]	Definition of API function ComM_GetPncToChannelMapping
[SWS_ComM_91015]	Definition of API function ComM_UpdatePncToChannelMapping
[SWS_ComM_91021]	Definition of API function ComM_UpdatePncMembership
[SWS_ComM_91028]	Definition of callback function ComM_Nm_UpdateEIRA
[SWS_ComM_91029]	Definition of callback function ComM_Nm_UpdateERA
[SWS_ComM_91030]	Definition of callback function ComM_Nm_ForwardSynchronizedPnc Shutdown
[SWS_ComM_91102]	Definition of ClientServerInterface ComM_PncToChannelMapping
[SWS_ComM_91108]	Definition of ClientServerInterface ComM_DynamicPncToChannelMapping

Table B.7: Changed Specification Items in R26-11

B.5.3 Deleted Specification Items in R26-11

Number	Heading
[ECUC_ComM_ 00622]	Definition of EcucBooleanParamDef ComMVersionInfoApi
[SWS_ComM_00370]	Definition of API function ComM_GetVersionInfo
[SWS_ComM_01019]	
[SWS_ComM_01020]	
[SWS_ComM_01059]	

Table B.8: Deleted Specification Items in R26-11

B.5.4 Added Constraints in R26-11

none

B.5.5 Changed Constraints in R26-11

none

B.5.6 Deleted Constraints in R26-11

none