<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017-12-08</td>
<td>4.3.1</td>
<td>AUTOSAR Release Management</td>
<td>• Clarification regarding communication inhibition and bus wake up inhibition</td>
</tr>
</tbody>
</table>
| 2016-11-30 | 4.3.0   | AUTOSAR Release Management          | • Added the possibility to switch ethernet 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 use case                                |
|            |         |                                     | • 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”               |
## Document Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
</table>
| 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 ComMPnclId 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 |


## Document Change History

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
</table>
| 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 |
<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Changed by</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-01-24</td>
<td>2.1.19</td>
<td>AUTOSAR Administration</td>
<td>• Changed features&lt;br&gt;• Restart (silent com. -&gt; full com.) now possible even if mode limitation is active&lt;br&gt;• Channel state machine changed&lt;br&gt;• Sequence diagrams changed&lt;br&gt;• New services to upper layers&lt;br&gt;• Mode indication API to RTE changed&lt;br&gt;• New calls to other modules&lt;br&gt;• Usage of channel specific API (EcuM and ComM) to indicate that a communication channel has been woken up and has gone to sleep&lt;br&gt;• API for NM control changed (Nm_PassiveStartUp, Nm_NetworkRequest, Nm_NetworkRelease)&lt;br&gt;• Legal disclaimer revised&lt;br&gt;• Release Notes added&lt;br&gt;• “Advice for users” revised&lt;br&gt;• “Revision Information” added</td>
</tr>
<tr>
<td>2005-05-31</td>
<td>1.0</td>
<td>AUTOSAR Administration</td>
<td>• Initial Release</td>
</tr>
</tbody>
</table>
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 ................................................. 10  
2 Acronyms and definitions ..................................................................... 11  
3 Related documentation ........................................................................ 13 
  3.1 Input documents ............................................................................... 13  
  3.2 Related standards and norms ............................................................ 14  
  3.3 Related specification ....................................................................... 15  
4 Constraints and assumptions ................................................................. 16 
  4.1 Limitations ...................................................................................... 16  
  4.2 Applicability to car domains ............................................................ 16  
5 Dependencies to other modules ............................................................. 17 
  5.1 File structure .................................................................................. 17  
  5.1.1 Header file structure ................................................................. 17  
  5.2 AUTOSAR Runtime Environment (RTE) ......................................... 18  
  5.3 ECU State Manager (EcuM) ............................................................ 18  
  5.4 Basic Software Mode Manager (BswM) ........................................... 19  
  5.5 NVRAM Manager ........................................................................... 19  
  5.6 Diagnostic Communication Manager (DCM) .................................... 19  
  5.7 LIN State Manager .......................................................................... 19  
  5.8 CAN State Manager ......................................................................... 20  
  5.9 FlexRay State Manager .................................................................... 20  
  5.10 Ethernet State Manager ................................................................. 20  
  5.11 Network Management (NM) ........................................................... 20  
  5.12 Default Error Tracer (DET) ............................................................. 20  
  5.13 Communication (COM) .................................................................. 20  
6 Requirements traceability ..................................................................... 21  
7 Functional specification ......................................................................... 26  
  7.1 Partial Network Cluster Management .............................................. 29 
  7.1.1 Overview .................................................................................... 29  
  7.1.2 Partial Network Cluster Management Functionality .................... 29  
  7.1.3 ComM PNC state machine .......................................................... 30  
  7.1.4 PNC Gateway ............................................................................ 37  
  7.1.5 ComM User to PNC Relations ..................................................... 38  
  7.1.6 Partial Networking Configuration Hints ....................................... 39  
  7.2 ComM channel state machine ............................................................ 40  
  7.2.1 Behavior in state COMM_NO_COMMUNICATION .................... 44  
  7.2.2 Behaviour in state COMM_SILENT_COMMUNICATION ............ 47  
  7.2.3 Behaviour in state COMM_FULL_COMMUNICATION ................ 48  
  7.3 Extended functionality ..................................................................... 52  
  7.3.1 Communication inhibition .......................................................... 52  
  7.4 Bus communication management .................................................... 56  
  7.5 Network management dependencies .............................................. 57  
  7.6 Bus error management .................................................................... 58  
  7.6.1 Network Start Indication ............................................................ 58
7.7 Test support requirements .......................................................... 58
  7.7.1 Inhibited Full Communication Request Counter ..................... 58
7.8 Error classification ...................................................................... 59
  7.8.1 Development errors .............................................................. 59
  7.8.2 Runtime Errors .................................................................... 59
  7.8.3 Transient Faults ................................................................... 59
7.9 Non functional requirements ....................................................... 59
7.10 Communication Manager Module Services .................................... 60
    7.10.1 Architecture .................................................................... 60
    7.10.2 Use Cases ........................................................................ 61
    7.10.3 Specification of Ports and Port Interfaces ......................... 64
    7.10.4 Runnables and Entry points ............................................. 68
8 API specification ............................................................................. 71
  8.1 Imported types .......................................................................... 71
    8.1.1 Standard types ................................................................... 71
8.2 Type definitions ........................................................................... 71
  8.2.1 ComM_InitStatusType ......................................................... 71
  8.2.2 ComM_PncModeType ............................................................ 72
  8.2.3 ComM_StateType ................................................................. 72
  8.2.4 ComM_ConfigType ............................................................... 72
8.3 Function definitions ...................................................................... 72
  8.3.1 ComM_Init ........................................................................... 73
  8.3.2 ComM_DelInit ..................................................................... 73
  8.3.3 ComM_GetState ................................................................... 74
  8.3.4 ComM_GetStatus ................................................................ 75
  8.3.5 ComM_GetInhibitionStatus ............................................... 75
  8.3.6 ComM_RequestComMode .................................................... 75
  8.3.7 ComM_GetMaxComMode ..................................................... 76
  8.3.8 ComM_GetRequestedComMode .......................................... 77
  8.3.9 ComM_GetCurrentComMode ................................................ 77
  8.3.10 ComM_PreventWakeUp ...................................................... 78
  8.3.11 ComM_LimitChannelToNoComMode .................................... 79
  8.3.12 ComM_LimitECUToNoComMode ........................................ 79
  8.3.13 ComM_ReadInhibitCounter .............................................. 80
  8.3.14 ComM_ResetInhibitCounter .............................................. 80
  8.3.15 ComM_SetECUGroupClassification ................................... 81
  8.3.16 ComM_GetVersionInfo ....................................................... 81
8.4 Callback notifications ................................................................. 81
  8.4.1 AUTOSAR Network Management Interface .......................... 82
  8.4.2 AUTOSAR Diagnostic Communication Manager Interface ... 84
  8.4.3 AUTOSAR ECU State Manager Interface ............................. 85
  8.4.4 AUTOSAR ECU State Manager and Basic Software Mode Manager Interface ......................................................... 85
  8.4.5 Bus State Manager Interface .............................................. 86
  8.4.6 COM Interface ................................................................. 86
8.5 Scheduled functions ..................................................................... 87
  8.5.1 ComM_MainFunction .......................................................... 87
8.6 Expected interfaces ....................................................................... 87
  8.6.1 Mandatory Interfaces .......................................................... 87
Specification of Communication Manager
AUTOSAR CP Release 4.3.1

8.6.2 Optional Interfaces .......................................................... 91
8.6.3 Configurable Interfaces .......................................................... 91
8.6.4 AUTOSAR COM ................................................................. 91
8.7 Service Interfaces ................................................................. 92
   8.7.1 Sender-Receiver-interfaces ................................................. 92
   8.7.2 Client-Server-interfaces ................................................... 93
   8.7.3 Mode-Switch-Interfaces .................................................... 98
   8.7.4 Implementation Data Types ................................................. 98
   8.7.5 Ports ............................................................................. 100
   8.7.6 ModeDeclarationGroups ..................................................... 102
9 Sequence diagrams ................................................................. 103
   9.1 Transmission and Reception start (CAN) .................................... 103
   9.2 Passive Wake-up (CAN) ........................................................ 104
   9.3 Network shutdown (CAN) ...................................................... 105
   9.4 Communication request ....................................................... 107
10 Configuration specification ..................................................... 108
   10.1 How to read this chapter ..................................................... 108
   10.2 Containers and configuration parameters ................................ 109
      10.2.1 ComM ....................................................................... 110
      10.2.2 ComMGeneral ............................................................. 110
      10.2.3 ComMConfigSet ........................................................... 116
      10.2.4 ComMUser ................................................................. 117
      10.2.5 ComMChannel ............................................................ 118
      10.2.6 ComMNetworkManagement .......................................... 124
      10.2.7 ComMUserPerChannel ............................................... 126
      10.2.8 ComMPnc ................................................................. 126
      10.2.9 ComMPncComSignal ................................................... 128
   10.3 Published information ......................................................... 129
11 Not applicable requirements .................................................... 130
List of Figures

Figure 1: Communication Manager Module context view .................................................. 17
Figure 2: PNC State Machine ......................................................................................... 32
Figure 3: User to Partial network and channel Mapping Use Cases .......................... 38
Figure 4: ComM channel state machine ................................................................. 41
Figure 5: ARPackage of the Communication Manager Module .............................. 61
Figure 6: SW-C requests state changes to the Communication Manager Module ... 61
Figure 7: SW-C requires state changes within the Communication Manager Module and reads out current communication state ........................................................................ 62
Figure 8: Interaction between BswM and the ComM module ........................................ 64
Figure 9: Starting transmission and reception on CAN .............................................. 103
Figure 10: Reaction on a wake-up indicated by the ECU State Manager module .. 104
Figure 11: Network shutdown (CAN) ........................................................................... 106
Figure 12: Request Communication ........................................................................... 107
Figure 13: Configuration ComM .................................................................................. 110
Figure 14: Configuration ComMGeneral ..................................................................... 116
Figure 15: Configuration ComMUser .......................................................................... 118
Figure 16: Configuration ComMChannel ................................................................. 123
Figure 17: Configuration ComMNetworkManagement .............................................. 125
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 7.3.1.2 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.
## 2 Acronyms and definitions

<table>
<thead>
<tr>
<th>Abbreviation / Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSW</td>
<td>Basic Software</td>
</tr>
<tr>
<td>BswM</td>
<td>Basic Software Mode Manager</td>
</tr>
<tr>
<td>ComM</td>
<td>Communication Manager</td>
</tr>
<tr>
<td>DCM</td>
<td>Diagnostic Communication Manager</td>
</tr>
<tr>
<td>Det</td>
<td>Default Error Tracer</td>
</tr>
<tr>
<td>EcuM</td>
<td>ECU State Manager module</td>
</tr>
<tr>
<td>I-PDU</td>
<td>Information Protocol Data Unit</td>
</tr>
<tr>
<td>NM</td>
<td>Network Management</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>SW-C</td>
<td>Software Component</td>
</tr>
<tr>
<td>VMM</td>
<td>Vehicle Message Matrix</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Term</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCM_ActiveDiagnostic indication</td>
<td>The DCM module indicates an active diagnostic session. DCM need “full communication” = COMM_FULL_COMMUNICATION for diagnostic purpose</td>
</tr>
<tr>
<td>Active wake-up</td>
<td>Wake-up caused by the hosting ECU e.g. by a sensor.</td>
</tr>
<tr>
<td>Application signal scheduling</td>
<td>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.</td>
</tr>
<tr>
<td>Bus sleep</td>
<td>No activity required on the communication bus (e.g. CAN bus sleep).</td>
</tr>
<tr>
<td>Bus communication messages</td>
<td>Bus communication messages are all messages that are sent on the communication bus. This can be either a diagnostic message or an application message.</td>
</tr>
<tr>
<td>COM Inhibition status</td>
<td>Defines whether full communication, silent communication or wake-up is allowed or not.</td>
</tr>
<tr>
<td>Communication Channel</td>
<td>The medium used to convey information from a sender (or transmitter) to a receiver.</td>
</tr>
<tr>
<td>Communication Mode</td>
<td>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</td>
</tr>
<tr>
<td>Diagnostic PDU scheduling</td>
<td>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.</td>
</tr>
<tr>
<td>ECU shut down</td>
<td>See ECU State Manager specification [6].</td>
</tr>
<tr>
<td>Fan-out</td>
<td>Same message/indication are sent to multiple destinations/receivers</td>
</tr>
<tr>
<td>Independent software component</td>
<td>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.</td>
</tr>
<tr>
<td>Passive wake-up</td>
<td>Wake-up by another ECU and propagated (e.g. by bus or wake-up-line) to the ECU currently in focus.</td>
</tr>
<tr>
<td>System User</td>
<td>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.</td>
</tr>
<tr>
<td>User</td>
<td>Concept for requestors of the ECU State Manager module and of the...</td>
</tr>
<tr>
<td>User Request</td>
<td>A User can request different Communication Modes from ComM</td>
</tr>
</tbody>
</table>

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.
3 Related documentation

3.1 Input documents

[1] List of Basic Software Modules
AUTOSAR_TR_BSWModuleList.pdf

[2] Layered Software Architecture
AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf

AUTOSAR_SRS_BSWGeneral.pdf

AUTOSAR_SRS_ModeManagement.pdf

AUTOSAR_TPS_ECUConfiguration.pdf

[6] Specification of ECU State Manager
AUTOSAR_SWS_ECUStateManager.pdf

AUTOSAR_SWS_NVRAMManager.pdf

AUTOSAR_SWS_RTE.pdf

AUTOSAR_SWS_NetworkManagementInterface.pdf

AUTOSAR_SWS_COM.pdf

AUTOSAR_SWS_DiagnosticCommunicationManager.pdf

AUTOSAR_SWS_LINInterface.pdf

AUTOSAR_SWS_FlexRayInterface.pdf

AUTOSAR_SWS_DefaultErrorTracer.pdf

AUTOSAR_SWS_CANTransceiverDriver.pdf
3.2 Related standards and norms

Not applicable.
3.3 Related specification

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

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

4.1 Limitations

The referenced deliverable AUTOSAR_SWS_ECUStateManagerFixed is set to status "obsolete" in release 4.3.1.

4.2 Applicability to car domains

No restrictions.
5 Dependencies to other modules

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

![Communication Manager Module context view](image)

Figure 1: Communication Manager Module context view

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

5.1 File structure

5.1.1 Header file structure

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

- ComM Schedule Manager: SchM_ComM.h
- RTE generated header file: Rte_ComM.h
- Default Error Tracer: Det.h
- Diagnostic Communication Manager: Dcm_Cbk.h
- NVRAM Manager: NvM.h
- ECU State Manager: EcuM.h
- Network Management Interface: Nm.h
- LIN State Manager: LinSM.h
- CAN State Manager: CanSM_ComM.h
- FlexRay State Manager: FrSM.h
- Ethernet State Manager: EthSM.h
Basic Software Mode Manager:  BswM_ComM.h 
Communication:  Com.h

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

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

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

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

5.2 AUTOSAR Runtime Environment (RTE)

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

5.3 ECU State Manager (EcuM)

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

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

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

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

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

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

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

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

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

5.5 NVRAM Manager

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

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

5.6 Diagnostic Communication Manager (DCM)

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

5.7 LIN State Manager

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

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

5.9 FlexRay State Manager

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

5.10 Ethernet State Manager

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

5.11 Network Management (NM)

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

5.12 Default Error Tracer (DET)

The DET provides services for reporting development, runtime, and transient errors. (see Section 7.8)

5.13 Communication (COM)

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

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Description</th>
<th>Satisfied by</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRS_BSW_00004</td>
<td>All Basic SW Modules shall perform a pre-processor check of the versions of all imported include files</td>
<td>SWS_ComM_00418</td>
</tr>
<tr>
<td>SRS_BSW_00005</td>
<td>Modules of the µC Abstraction Layer (MCAL) may not have hard coded horizontal interfaces</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00009</td>
<td>All Basic SW Modules shall be documented according to a common standard.</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00010</td>
<td>The memory consumption of all Basic SW Modules shall be documented for a defined configuration for all supported platforms.</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00101</td>
<td>The Basic Software Module shall be able to initialize variables and hardware in a separate initialization function</td>
<td>SWS_ComM_00146</td>
</tr>
<tr>
<td>SRS_BSW_00158</td>
<td>All modules of the AUTOSAR Basic Software shall strictly separate configuration from implementation</td>
<td>SWS_ComM_00464</td>
</tr>
<tr>
<td>SRS_BSW_00161</td>
<td>The AUTOSAR Basic Software shall provide a microcontroller abstraction layer which provides a standardized interface to higher software layers</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00162</td>
<td>The AUTOSAR Basic Software shall provide a hardware abstraction layer</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00164</td>
<td>The Implementation of interrupt service routines shall be done by the Operating System, complex drivers or modules</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00167</td>
<td>All AUTOSAR Basic Software Modules shall provide configuration rules and constraints to enable plausibility checks</td>
<td>SWS_ComM_00419</td>
</tr>
<tr>
<td>SRS_BSW_00168</td>
<td>SW components shall be tested by a function defined in a common API in the Basis-SW</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00170</td>
<td>The AUTOSAR SW Components shall provide information about their dependency from faults, signal qualities, driver demands</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00314</td>
<td>All internal driver modules shall separate the interrupt frame definition from the service routine</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00323</td>
<td>All AUTOSAR Basic Software Modules shall check passed API parameters for validity</td>
<td>SWS_ComM_00234</td>
</tr>
<tr>
<td>SRS_BSW_00325</td>
<td>The runtime of interrupt service routines and functions that are running in interrupt context shall be kept short</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>Specification</td>
<td>AUTOSAR CP Release 4.3.1</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>SRS_BSW_00327</strong></td>
<td>Error values naming convention</td>
<td>SWS_ComM_00234</td>
</tr>
<tr>
<td><strong>SRS_BSW_00331</strong></td>
<td>All Basic Software Modules shall strictly separate error and status information</td>
<td>SWS_ComM_00649</td>
</tr>
<tr>
<td><strong>SRS_BSW_00336</strong></td>
<td>Basic SW module shall be able to shutdown</td>
<td>SWS_ComM_00147</td>
</tr>
<tr>
<td><strong>SRS_BSW_00337</strong></td>
<td>Classification of development errors</td>
<td>SWS_ComM_00234</td>
</tr>
<tr>
<td><strong>SRS_BSW_00341</strong></td>
<td>Module documentation shall contains all needed informations</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td><strong>SRS_BSW_00342</strong></td>
<td>It shall be possible to create an AUTOSAR ECU out of modules provided as source code and modules provided as object code, even mixed</td>
<td>SWS_ComM_00459</td>
</tr>
<tr>
<td><strong>SRS_BSW_00343</strong></td>
<td>The unit of time for specification and configuration of Basic SW modules shall be preferably in physical time unit</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td><strong>SRS_BSW_00344</strong></td>
<td>BSW Modules shall support link-time configuration</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td><strong>SRS_BSW_00345</strong></td>
<td>BSW Modules shall support pre-compile configuration</td>
<td>SWS_ComM_00456</td>
</tr>
<tr>
<td><strong>SRS_BSW_00346</strong></td>
<td>All AUTOSAR Basic Software Modules shall provide at least a basic set of module files</td>
<td>SWS_ComM_00463</td>
</tr>
<tr>
<td><strong>SRS_BSW_00348</strong></td>
<td>All AUTOSAR standard types and constants shall be placed and organized in a standard type header file</td>
<td>SWS_ComM_00820</td>
</tr>
<tr>
<td><strong>SRS_BSW_00353</strong></td>
<td>All integer type definitions of target and compiler specific scope shall be placed and organized in a single type header</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td><strong>SRS_BSW_00357</strong></td>
<td>For success/failure of an API call a standard return type shall be defined</td>
<td>SWS_ComM_00820</td>
</tr>
<tr>
<td><strong>SRS_BSW_00358</strong></td>
<td>The return type of init() functions implemented by AUTOSAR Basic Software Modules shall be void</td>
<td>SWS_ComM_00146</td>
</tr>
<tr>
<td><strong>SRS_BSW_00361</strong></td>
<td>All mappings of not standardized keywords of compiler specific scope shall be placed and organized in a compiler specific type and keyword header</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td><strong>SRS_BSW_00369</strong></td>
<td>All AUTOSAR Basic Software Modules shall not return specific development error codes via the API</td>
<td>SWS_ComM_00649</td>
</tr>
<tr>
<td><strong>SRS_BSW_00373</strong></td>
<td>The main processing function of each AUTOSAR Basic Software Module shall be named according the defined convention</td>
<td>SWS_ComM_00429</td>
</tr>
<tr>
<td><strong>SRS_BSW_00375</strong></td>
<td>Basic Software Modules shall report wake-up reasons</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td><strong>SRS_BSW_00377</strong></td>
<td>A Basic Software Module can return a module specific types</td>
<td>SWS_ComM_00649</td>
</tr>
<tr>
<td><strong>SRS_BSW_00378</strong></td>
<td>AUTOSAR shall provide a boolean type</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00381</td>
<td>The pre-compile time parameters shall be placed into a separate configuration header file</td>
<td>SWS_ComM_00463</td>
</tr>
<tr>
<td>SRS_BSW_00385</td>
<td>List possible error notifications</td>
<td>SWS_ComM_00234</td>
</tr>
<tr>
<td>SRS_BSW_00386</td>
<td>The BSW shall specify the configuration for detecting an error</td>
<td>SWS_ComM_00234</td>
</tr>
<tr>
<td>SRS_BSW_00398</td>
<td>The link-time configuration is achieved on object code basis in the stage after compiling and before linking</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00404</td>
<td>BSW Modules shall support post-build configuration</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00405</td>
<td>BSW Modules shall support multiple configuration sets</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00406</td>
<td>A static status variable denoting if a BSW module is initialized shall be initialized with value 0 before any APIs of the BSW module is called</td>
<td>SWS_ComM_00242, SWS_ComM_00612, SWS_ComM_00858</td>
</tr>
<tr>
<td>SRS_BSW_00407</td>
<td>Each BSW module shall provide a function to read out the version information of a dedicated module implementation</td>
<td>SWS_ComM_00370</td>
</tr>
<tr>
<td>SRS_BSW_00412</td>
<td>References to configuration parameters shall be placed into a separate h-file</td>
<td>SWS_ComM_00463</td>
</tr>
<tr>
<td>SRS_BSW_00413</td>
<td>An index-based accessing of the instances of BSW modules shall be done</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00414</td>
<td>Init functions shall have a pointer to a configuration structure as single parameter</td>
<td>SWS_ComM_00146</td>
</tr>
<tr>
<td>SRS_BSW_00415</td>
<td>Interfaces which are provided exclusively for one module shall be separated into a dedicated header file</td>
<td>SWS_ComM_00463</td>
</tr>
<tr>
<td>SRS_BSW_00416</td>
<td>The sequence of modules to be initialized shall be configurable</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00417</td>
<td>Software which is not part of the SW-C shall report error events only after the DEM is fully operational.</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00422</td>
<td>Pre-de-bouncing of error status information is done within the DEM</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00423</td>
<td>BSW modules with AUTOSAR interfaces shall be describable with the means of the SW-C Template</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00424</td>
<td>BSW module main processing functions shall not be allowed to enter a wait state</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00425</td>
<td>The BSW module description template shall provide means to model the defined trigger conditions of schedulable objects</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00426</td>
<td>BSW Modules shall ensure data consistency of data which is shared between BSW modules</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00427</td>
<td>ISR functions shall be defined and</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>Specification of Communication Manager</td>
<td>AUTOSAR CP Release 4.3.1</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--------------------------</td>
<td></td>
</tr>
<tr>
<td>SRS_BSW_00428</td>
<td>A BSW module shall state if its main processing function(s) has to be executed in a specific order or sequence</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00429</td>
<td>Access to OS is restricted</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00432</td>
<td>Modules should have separate main processing functions for read/receive and write/transmit data path</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00433</td>
<td>Main processing functions are only allowed to be called from task bodies provided by the BSW Scheduler</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00437</td>
<td>Memory mapping shall provide the possibility to define RAM segments which are not to be initialized during startup</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00438</td>
<td>Configuration data shall be defined in a structure</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00439</td>
<td>Enable BSW modules to handle interrupts</td>
<td>SWS_ComM_00499</td>
</tr>
<tr>
<td>SRS_BSW_00441</td>
<td>Naming convention for type, macro and function</td>
<td>SWS_ComM_00649, SWS_ComM_00863</td>
</tr>
<tr>
<td>SRS_ModeMgm_00049</td>
<td>The Communication Manager shall initiate the wake-up and keep awake physical channels</td>
<td>SWS_ComM_00869, SWS_ComM_00870</td>
</tr>
<tr>
<td>SRS_ModeMgm_09071</td>
<td>It shall be possible to limit communication modes independently for each physical channel</td>
<td>SWS_ComM_00303</td>
</tr>
<tr>
<td>SRS_ModeMgm_09078</td>
<td>The Communication Manager shall coordinate multiple communication requests</td>
<td>SWS_ComM_00686</td>
</tr>
<tr>
<td>SRS_ModeMgm_09080</td>
<td>Each physical channel shall be controlled by an independent communication mode</td>
<td>SWS_ComM_00051</td>
</tr>
<tr>
<td>SRS_ModeMgm_09081</td>
<td>The Communication Manager shall provide an API allowing collecting communication requests</td>
<td>SWS_ComM_00110</td>
</tr>
<tr>
<td>SRS_ModeMgm_09083</td>
<td>The Communication Manager shall support two communication modes for each physical channel</td>
<td>SWS_ComM_00845, SWS_ComM_00846, SWS_ComM_00867, SWS_ComM_00868</td>
</tr>
<tr>
<td>SRS_ModeMgm_09084</td>
<td>The Communication Manager shall provide an API which allows application to query the current communication mode</td>
<td>SWS_ComM_00083</td>
</tr>
<tr>
<td>SRS_ModeMgm_09085</td>
<td>The Communication Manager shall provide an indication of communication mode changes</td>
<td>SWS_ComM_00091</td>
</tr>
<tr>
<td>SRS_ModeMgm_09087</td>
<td>The Minimum duration of communication request after wakeup shall be configurable</td>
<td>SWS_ComM_00893, SWS_ComM_00894</td>
</tr>
<tr>
<td>SRS_ModeMgm_09089</td>
<td>The Communication Manager shall be able to prevent waking up physical</td>
<td>SWS_ComM_00302</td>
</tr>
<tr>
<td>Requirement ID</td>
<td>Description</td>
<td>Related SRS and SWS Numbers</td>
</tr>
<tr>
<td>---------------</td>
<td>------------------------------------------------------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>SRS_ModeMgm_09090</td>
<td>Relationship between users and physical channels shall be configurable at pre compile time</td>
<td>SWS_ComM_00159</td>
</tr>
<tr>
<td>SRS_ModeMgm_09133</td>
<td>It shall be possible to assign physical channels to the Communication Manager</td>
<td>SWS_ComM_00327</td>
</tr>
<tr>
<td>SRS_ModeMgm_09149</td>
<td>The Communication Manager shall provide an API for querying the requested communication mode</td>
<td>SWS_ComM_00079</td>
</tr>
<tr>
<td>SRS_ModeMgm_09155</td>
<td>The Communication Manager shall provide a counter for inhibited communication requests</td>
<td>SWS_ComM_00138</td>
</tr>
<tr>
<td>SRS_ModeMgm_09156</td>
<td>It shall be provided an API to retrieve the number of inhibited &quot;Full Communication&quot; mode requests</td>
<td>SWS_ComM_00108, SWS_ComM_00224</td>
</tr>
<tr>
<td>SRS_ModeMgm_09157</td>
<td>It shall be possible to revoke a communication mode limitation, independently for each physical channel</td>
<td>SWS_ComM_00124, SWS_ComM_00156, SWS_ComM_00163</td>
</tr>
<tr>
<td>SRS_ModeMgm_09168</td>
<td>The Communication Manager shall support users that are connected to no physical channel</td>
<td>SWS_ComM_00664</td>
</tr>
<tr>
<td>SRS_ModeMgm_09172</td>
<td>It shall be possible to evaluate the current communication mode</td>
<td>SWS_ComM_00176</td>
</tr>
<tr>
<td>SRS_ModeMgm_09243</td>
<td>The Communication Manager shall be able to handle the Partial Networks on Flexray and CAN</td>
<td>SWS_ComM_00164, SWS_ComM_00959</td>
</tr>
<tr>
<td>SRS_ModeMgm_09246</td>
<td>The communication manager shall arbitrate and coordinate requests from users on physical channel and users on PNCs</td>
<td>SWS_ComM_00165</td>
</tr>
<tr>
<td>SRS_ModeMgm_09247</td>
<td>For each configured PNC an independent state machine shall be instantiated</td>
<td>SWS_ComM_00165</td>
</tr>
<tr>
<td>SRS_ModeMgm_09248</td>
<td>It shall be possible to distinguish between internal and external PNC activation requests</td>
<td>SWS_ComM_00165, SWS_ComM_01014, SWS_ComM_01015</td>
</tr>
</tbody>
</table>
7 Functional specification

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[SWS_ComM_00885] Status changes for communication allowed or not allowed in SWS_ComM_00884 shall be provided to ComM in ComM_CommunicationAllowed(<channel>, TRUE|FALSE) (SWS_ComM_00871) indications.] ()
7.1 Partial Network Cluster Management

7.1.1 Overview

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

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

ComM users are used to request and release the PNCs.

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

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

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

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

Enabling or disabling the partial network cluster management in the node shall be post-build configurable.

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

7.1.2 Partial Network Cluster Management Functionality

[SWS_ComM_00910] [PNC functionality shall only exist if the parameter ComMPncSupport is set to TRUE. (see ECUC_ComM_00839).] ()

[SWS_ComM_00911] [Enabling or disabling of the PNC functionality shall be post-build configurable using the parameter ComMPncEnabled (see ECUC_ComM_00878).] ()
Comment: The ComM module notifies the BswM about every state change of the PNC state machine by calling BswM_ComM_CurrentPncMode(). (refer to SWS_ComM_00908)

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

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

[SWS_ComM_00825] The bytelnIndex and bitIndex, in which a bit corresponding to one ComMPncNId resides, shall be determined as follows:
    byteIndex=(ComMPncNId div 8) - <PNC Vector Offset>
    bitIndex=(ComMPncNId mod 8) ] ()

Hint: The value of <PNC Vector Offset> (and <PNC Vector Length>, if needed) can be obtained from the <Bus> Network Management modules configuration. Comment: SWS_ComM_00825 defines only the calculation of the byteIndex and bitIndex, not how it shall be implemented.

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

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

[SWS_ComM_00916] The ComM module shall be able to distribute the status of a PNC (result of the PNC state machine) via one or more communications busses using one or more COM signals , as a bit vector, containing a bit which represents the status of the PNC with ComMPncComSignalDirection “TX” assigned to this PNC. (For more details, refer to Figure 2: PNC State Machine)] ()

7.1.3 ComM PNC state machine

[SWS_ComM_00953] If the PNC functionality is enabled using the configuration parameter ComMPncEnabled set to TRUE (see ECUC_ComM_00878), all actions related to PNC changes shall be executed before the channel related actions (channel related actions, see Chapter 7.3). ] ()

[SWS_ComM_00909] For every Partial Network, only one PNC state machine shall be implemented (i.e. One PNC state machine per PNC, independent of the amount of ComMChannels). ] ()
[SWS_ComM_00920]  The ComM module shall support up to 56 PNC state machines.

[SWS_ComM_00924]  The PNC state machine shall consist of the two main states:

\[
\text{COMM\_PNC\_FULL\_COMMUNICATION and } \text{COMM\_PNC\_NO\_COMMUNICATION.}
\]

[SWS_ComM_00907]  The PNC main state COMM\_PNC\_FULL\_COMMUNICATION shall consist of the sub states:

\[
\text{COMM\_PNC\_PREPARE\_SLEEP, } \text{COMM\_PNC\_READY\_SLEEP} \text{ and } \text{COMM\_PNC\_REQUESTED.}
\]

[SWS_ComM_00908]  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]  State transitions of the PNC state machines in ComM, triggered by a call to ComM\_RequestComMode() shall be executed in the ComM\_MainFunction\_<Channel\_Id> only.

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

[SWS_ComM_00944]  If at least one bit corresponding to the PNC within the Rx bitvectors with signal type "EIRA" equals ‘1’, then the bit corresponding to this PNC within ERA in ComM shall be set to ‘1’

[SWS_ComM_00945]  If the configuration parameter ComMPncGatewayEnabled (see ECUC\_ComM\_00840) is true and the parameter ComMPncGatewayType is set to COMM\_GATEWAY\_TYPE\_ACTIVE for a ComMChannel and at least one bit corresponding to the PNC within the Rx bitvectors with signal type "ERA" equals ‘1’, then the bit corresponding to this PNC within ERA in ComM shall be set to ‘1’.

[SWS_ComM_00971]  The trigger ComM\_COMCbk represents a notification by the AUTOSAR Communication module about a received signal containing PNC status information called ERA of EIRA.

[SWS_ComM_00972]  The trigger “ComMUser” represents a notification about a communication request of a ComMUser by calling the API ComM\_RequestComMode().

[SWS_ComM_00987]  Within the ComM\_MainFunction\_<Channel\_Id> of a channel that is mapped to one or more PNCs, the requested state shall be handled in the following order:

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

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

[SWS_ComM_00919] It shall be possible to assign more than one COM signal containing bits representing the PNC to one PNC using the configuration container `ComM_PncComSignal` (see [ECUC_ComM_00881]). ()

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

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

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

---

**Figure 2:** PNC State Machine

7.1.3.1 Behavior in PNC main state `COMM_PNC_NO_COMMUNICATION`

[SWS_ComM_00926] 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_00931] If the API `ComM_EcuM_WakeUpIndication()` is called in PNC state `COMM_PNC_NO_COMMUNICATION`, and the configuration switch `ComMSynchronousWakeUp` is set to TRUE (see `ECUC_ComM_00695`), 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] 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 is set to ‘1’).

[SWS_ComM_00964] If the API `ComM_EcuM_PNCWakeUpIndication(<PNC>)` is called in PNC state `PNC_NO_COMMUNICATION`, the PNC main state `PNC_NO_COMMUNICATION` shall be left and the PNC sub state `PNC_PREPARE_SLEEP` shall be entered.] ()

[SWS_ComM_00932] 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_00933] When in main state `COMM_PNC_NO_COMMUNICATION` at least one bit representing this PNC in EIRA changes to ‘1’, the main state `COMM_PNC_NO_COMMUNICATION` shall be left and the `COMM_PNC_READY_SLEEP` shall be entered.] ()

[SWS_ComM_00934] When in main state `COMM_PNC_NO_COMMUNICATION` at least one bit representing this PNC in an ERAn changes to ‘1’, the main state `COMM_PNC_NO_COMMUNICATION` shall be left and the sub state `COMM_PNC_REQUESTED` shall be entered if the parameter `ComMPncGatewayEnabled` (ECUC_ComM_00840) equals TRUE.] ()

7.1.3.2 On entry of PNC main state `COMM_PNC_NO_COMMUNICATION` from `PowerOff`

[SWS_ComM_00927] After switching on the power supply, main state `COMM_PNC_NO_COMMUNICATION` shall be entered from `PowerOff`.}()
7.1.3.3 Behavior in PNC main state COMM_PNC_FULL_COMMUNICATION

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

7.1.3.4 On entry of PNC sub state COMM_PNC_REQUESTED

[SWS_ComM_00930] [ When entering the PNC sub state COMM_PNC_REQUESTED and if ComMPncGatewayEnabled = FALSE, the API Com_SendSignal() shall be called with the value ‘1’ for the bit representing this PNC for the Com signal assigned to this PNC with ComMPncComSignalDirection “TX”. ] ()

[SWS_ComM_00992] [ When entering the PNC sub state COMM_PNC_REQUESTED and if ComMPncGatewayEnabled = TRUE, the API Com_SendSignal() shall be called with the value ‘1’ for the bits representing this PNC for the Com signals assigned to this PNC with ComMPncComSignalDirection “TX” for all ComM channels related to this PNC and having ComMPncGatewayType == COMM_GATEWAY_TYPE_ACTIVE. ] ()

[SWS_ComM_00993] [ Every time the sub state COMM_PNC_REQUESTED is entered from other states, ComM shall request COMM_FULL_COMMUNICATION for all configured ComM channels for this PNC, even if the channel is already requested. ] ()

7.1.3.5 Behavior in PNC sub state COMM_PNC_REQUESTED

[SWS_ComM_00164] [ If ComMPncGatewayEnabled == TRUE and either on entering COMM_PNC_REQUESTED or within COMM_PNC_REQUESTED, the API Com_SendSignal() shall be called with the value ”1” for the bit representing this PNC for the Com signal assigned to this PNC with ComMPncComSignalDirection == ”TX” on all ComM channels related to this PNC where ComMPncGatewayType == COMM_GATEWAY_TYPE_PASSIVE if at least one ComM user assigned to this PNC request ”Full Com” or at least one ComMPncComSignal received by Com_ReceiveSignal() from a channel where the signal attributes ComMPncComSignalDirection == ”RX” and ComMPncComSignalKind == ”ERA” and the channel attribute ComMPncGatewayType == ”COMM_GATEWAY_TYPE_ACTIVE” is set to ”1”. ] (SRS_ModeMgm_09243)

[SWS_ComM_00959] [ If ComMPncGatewayEnabled == TRUE and within COMM_PNC_REQUESTED, the API Com_SendSignal() shall be called with the value ”0” for the bit representing this PNC for the Com signal assigned to this PNC with ComMPncComSignalDirection == ”TX” on all ComM channels related to this PNC where ComMPncGatewayType == COMM_GATEWAY_TYPE_PASSIVE if all ]
ComM users assigned to this PNC request "No Com" and all ComPncComSignals received by Com_ReceiveSignal() from a channel where the signal attributes ComPncComSignalDirection == "RX" and ComPncComSignalKind == "ERA" and the channel attribute ComPncGatewayType == "COMM_GATEWAY_TYPE_ACTIVE" are set to "0". [SRS_ModeMgm_09243]

As long as at least one ComMUser assigned to this PNC requests "Full Communication", COMM_PNC_REQUESTED will be the current PNC state.

[SWS_ComM_00966]  If ComM0PncVectorAvoidance is set to TRUE and if all signals are set to '0' (because of SWS_ComM_00959) for the referenced ComMChannel(s), the ComM module shall release this ComMChannel.
As soon as at least one signal is set back to ‘1’ again, the ComM module shall request this ComMChannel again.] ()

As long as a PNC is requested remotely (i.e. at least one bit within the ERA signal assigned to this PNC equals ‘1’) and the configuration switch ComPncGatewayEnabled is set to TRUE (see ECUC_ComM_00840), COMM_PNC_REQUESTED will be the current PNC state.

[SWS_ComM_00938]  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 the configuration switch ComPncGatewayEnabled is set to FALSE.] ()

[SWS_ComM_00991]  When all ComMUsers assigned to this PNC request “No Communication” and the PNC bit in all ERA(n is equal to 0, the sub state COMM_PNC_REQUESTED shall be left and the sub state COMM_PNC_READY_SLEEP shall be entered, if the configuration switch ComPncGatewayEnabled is set to TRUE.] ()

7.1.3.6 On entry PNC sub state COMM_PNC_READY_SLEEP

[SWS_ComM_00960]  When entering the PNC sub state COMM_PNC_READY_SLEEP from COMM_PNC_REQUESTED, the API Com_SendSignal() shall be called with the value ‘0’ for the bit representing this PNC for all Com signals assigned to this PNC with ComPncComSignalDirection "TX".] ()

[SWS_ComM_00961]  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 for this PNC.] ()

7.1.3.7 Behavior in PNC sub state COMM_PNC_READY_SLEEP
As long as the PNC is requested (i.e. at least one PNC bit within EIRA equals ‘1’) and no ComMUser assigned to this PNC requests “Full Communication”, COMM_PNC_READY_SLEEP will be the current state.

[SWS_ComM_00940] If the PNC is released (i.e. all PNC bits 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_00165] In PNC sub state COMM_PNC_READY_SLEEP when at least one ComMUser assigned to this PNC requests “Full Communication” or ComMPncGatewayEnabled (ECUC_ComM_00887) equals TRUE and one bit representing this PNC in an ERAn changes to ‘1’, this state shall be left and the sub state COMM_PNC_REQUESTED shall be entered.

7.1.3.8 On entry of PNC sub state COMM_PNC_PREPARE_SLEEP

[SWS_ComM_00952] If the sub state COMM_PNC_PREPARE_SLEEP is entered, the timer ComMPncPrepareSleepTimer (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] 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] 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] When in COMM_PNC_PREPARE_SLEEP at least one PNC bit within EIRA changes to ‘1’, 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.

[SWS_ComM_00951] When in sub state COMM_PNC_PREPARE_SLEEP at least one PNC bit within ERAn changes to ‘1’ and the parameter ComMPncGatewayEnabled equals TRUE, the sub state
COMM_PNC_PREPARE_SLEEP shall be left. The timer ComMPncPrepareSleepTimer shall be stopped and the sub state COMM_PNC_REQUESTED shall be entered.]

7.1.4 PNC Gateway

[SWS_ComM_00981] If the configuration parameter ComMPncGatewayEnabled (see ECUC_ComM_00840) is TRUE, the default gateway type shall be active (COMM_GATEWAY_TYPE_ACTIVE).]

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

7.1.4.1 Active PNC Gateway

Note: Even if the configuration parameter ComMPncGatewayEnabled (see ECUC_ComM_00840) 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 2: 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 bit for a PNC is equal to zero in all ERA, no other node than the PNC gateway is requesting the PNC.

7.1.4.2 Passive PNC Gateway

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

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_00959: A PNC gateway calculates the PNCs bit value in the ERA Tx bitvectors to be sent for a passively coordinated channel, in the same manner as the bit value in ERA for an actively coordinated channel, but sets the PNC's bit to ‘0’ according to the rules of SWS_ComM_00959.

7.1.5 ComM User to PNC Relations

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

[SWS_ComM_00912] It shall be possible to map a configurable amount of ComMUsers to one or more PNCs using the parameter ComMUserPerPnc (see ECUC_ComM_00876).

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

Comment: The BusNM Filter configuration shall be 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] [ ] It shall be possible to map a configurable amount of ComMUUsers to one or more ComM channels using the parameter ComMUserPerChannel. ] ()

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

[SWS_ComM_00913] [ ] It shall be possible to map a configurable amount of PNC(s) to a configurable amount of ComM channels using the parameter ComMChannelPerPnc (see ECUC_ComM_00880). ] ()

[SWS_ComM_00996] [ ] It shall not be possible to map a ComMUUsers to a PNC and in addition to a ComM channel which is already referenced by the PNC (see figure 3 Use Case 2)] ()

Rational: Avoid redundant configuration since the channel is implicitly already referenced by the PNC.

[SWS_ComM_CONSTR_00001] [ ] ComM channel's that are referenced by a PNC are not allowed to be referenced by any ComMUUsers, if the PNC references at least one EthIfSwitchPortGroup (see figure 3 Use Case 6). 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 ComMUUsers. ] ()

Rational: If using PNC and SwitchPortGroups were derived (EcuInstance.ethSwitchPortGroupDerivation==TRUE), then the SwitchPortGroups are switched by the EthIf_SwitchPortGroupRequestMode API and not by a channel request.

7.1.6 Partial Networking Configuration Hints

Typically, every PNCSignal is sent / received on ComMChannels which are linked to this PNC.

However, this is not mandatory, but it is also allowed having a configuration where:

- PNCSignal can be sent on a ComMChannel which is _NOT_ linked to this PNC.
- A PNCSignal can be received via a ComMChannel which is _NOT_ linked to this PNC.
7.2 ComM channel state machine

[SWS_ComM_00979] [If the optional PNC functionality is enabled (see ECUC_ComM_00883), all PNC actions shall be performed before the channel related actions are executed.] ()

[SWS_ComM_00980] [If the parameter ComMPncNmRequest equals 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_00993) API Nm_NetworkRequest() shall be called, even if the current state is already "Full communication".] ()

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

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

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

**Use Case for SWS_ComM_00051:** On an ECU with CAN and LIN channel, only the LIN requires full communication to request e.g. sensor values while the CAN remains inactive.
Figure 4: ComM channel state machine
7.2.3
Entering state: SWS_ComM_00069
In state: SWS_ComM_00637, SWS_ComM_00826

7.2.3.1
sub-state COMM_FULL_COM_NETWORK_REQUESTED:
In sub-state: SWS_ComM_00869, SWS_ComM_00870, SWS_ComM_00665, SWS_ComM_00888, SWS_ComM_00889, SWS_ComM_00890

7.2.3.2
sub-state COMM_FULL_COM_READY_SLEEP
Entering sub-state: SWS_ComM_00133
In sub-state: SWS_ComM_00610, SWS_ComM_00671, SWS_ComM_00882, SWS_ComM_00883

<table>
<thead>
<tr>
<th>Transition</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM_NO_COMMUNICATION → COMM_FULL_COMMUNICATION</td>
<td>SWS_ComM_00893, SWS_ComM_00894, SWS_ComM_00694, SWS_ComM_00875, SWS_ComM_00876, SWS_ComM_01014, SWS_ComM_01015</td>
</tr>
<tr>
<td>COMM_FULL_COM_NETWORK_REQUESTED → COMM_FULL_COM_READY_SLEEP</td>
<td>SWS_ComM_00665</td>
</tr>
<tr>
<td>COMM_FULL_COM_READY_SLEEP → COMM_FULL_COM_NETWORK_REQUESTED</td>
<td>SWS_ComM_00882, SWS_ComM_00883</td>
</tr>
<tr>
<td>COMM_FULL_COMMUNICATION → COMM_SILENT_COMMUNICATION</td>
<td>SWS_ComM_00826</td>
</tr>
<tr>
<td>COMM_FULL_COM_READY_SLEEP → COMM_NO_COMMUNICATION</td>
<td>SWS_ComM_00610, SWS_ComM_00671</td>
</tr>
<tr>
<td>COMM_FULL_COMMUNICATION → COMM_NO_COMMUNICATION</td>
<td>SWS_ComM_00637</td>
</tr>
<tr>
<td>COMM_SILENT_COMMUNICATION → COMM_FULL_COMMUNICATION</td>
<td>SWS_ComM_00877, SWS_ComM_00878</td>
</tr>
<tr>
<td>COMM_SILENT_COMMUNICATION → COMM_FULL_COM_READY_SLEEP</td>
<td>SWS_ComM_00296</td>
</tr>
<tr>
<td>COMM_SILENT_COMMUNICATION → COMM_NO_COMMUNICATION</td>
<td>SWS_ComM_00295</td>
</tr>
</tbody>
</table>

Table 1: Link to detailed explanation of the channel state machine resp. transition
[SWS_ComM_00879] [ The ComM channel state machine shall consist of the three main states corresponding to the Communication Modes: COMM_NO_COMMUNICATION, COMM_SILENT_COMMUNICATION and COMM_FULL_COMMUNICATION. ] ()

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

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

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

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

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

[SWS_ComM_00896] [ Each ComM channel state machine shall only evaluate its corresponding communication status flag CommunicationAllowed according to SWS_ComM_00884 in sub-state COMM_NO_COM_REQUEST_PENDING. ] ()

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

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

Comment: If more than one user is related to the corresponding channel state machine, the ComM module has to perform a Fan-out to all users.
The internal functionality of the ComM channel state machine(s) shall be invisible for the users. The user neither needs nor shall get any information about the internal mechanisms and rules (e.g. "highest wins" strategy) of the ComM channel state machine.

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

<table>
<thead>
<tr>
<th>Communication Mode</th>
<th>Message Transmission</th>
<th>Message Reception</th>
<th>NM (COMM_NM_VARIANT=FULL)</th>
<th>Wake-up/Restart capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM_FULL_COMMUNICATION</td>
<td>On</td>
<td>On</td>
<td>Bus communication requested</td>
<td>N/A</td>
</tr>
<tr>
<td>COMM_SILENT_COMMUNICATION</td>
<td>Off</td>
<td>On</td>
<td>Bus communication released</td>
<td>• User/diagnostic request</td>
</tr>
<tr>
<td>COMM_NO_COMMUNICATION</td>
<td>Off</td>
<td>Off</td>
<td>Bus communication released</td>
<td>• Network indication</td>
</tr>
</tbody>
</table>

Table 2: Granted communication capabilities in the corresponding modes

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

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

7.2.1 Behavior in state COMM_NO_COMMUNICATION

On entering state COMM_NO_COMMUNICATION the ComM channel state machine shall go to sub-state COMM_NO_COM_NO_PENDING_REQUEST.

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.

On entering state COMM_NO_COMMUNICATION the ComM channel state machine shall switch off the transmission and reception capability. This shall be performed by the ComM channel state machine requesting the corresponding Communication Mode from the Bus State Manager module (XXSM_RequestComMode(network:=<channel state machine’s network>, mode:= COMM_NO_COMMUNICATION, see SWS_ComM_00829)).
Rationale for SWS_ComM_00073: The COMM_NO_COMMUNICATION mode forbids sending and receiving of bus communication PDUs for the corresponding channels.

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

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.

7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state

[SWS_ComM_00875] | In sub-state COMM_NO_COM_NO_PENDING_REQUEST and user requests COMM_FULL_COMMUNICATION and communication limitation is disabled (see Section 7.3.1), the ComM channel state machine shall immediately switch to sub-state COMM_NO_COM_REQUEST_PENDING.

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

Rationale for SWS_ComM_00876: A potential communication limitation (see Section 7.3.1) shall temporarily be inactive during an active diagnostic session, see SWS_ComM_00182.

[SWS_ComM_00893] | If ComM_EcuM_WakeUpIndication is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter
ComMSynchronousWakeUp=FALSE (ECUC_ComM_00695), the ComM module shall switch the requested ComM channel state machine (resp. channels) to sub-state
COMM_NO_COM_REQUEST_PENDING. (SRS_ModeMgm_09087)

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

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

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

[SWS_ComM_00694] If ComM_EcuM_WakeUpIndication is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695), the ComM module shall switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING. (SRS_ModeMgm_09248)

[SWS_ComM_01014] If ComM_EcuM_PNCWakeUpIndication(<PNC>) is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameters ComMSynchronousWakeUp=FALSE (ECUC_ComM_00695) and ComMPncSupport=TRUE (see ECUC_ComM_00839), the ComM module shall switch these ComM channel state machines (resp. channels) which are referenced by the PNC to sub-state COMM_NO_COM_REQUEST_PENDING. (SRS_ModeMgm_09248)

[SWS_ComM_01015] If ComM_EcuM_PNCWakeUpIndication(<PNC>) is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameters ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and ComMPncSupport=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. (SRS_ModeMgm_09248)

7.2.1.2 COMM_NO_COM_REQUEST_PENDING sub-state
[SWS_ComM_00895] In sub-state COMM_NO_COM_REQUEST_PENDING the ComM channel state machine shall evaluate its corresponding CommunicationAllowed flag, stored and set according to SWS_ComM_00884 and SWS_ComM_00885. If evaluated to CommunicationAllowed=TRUE, the ComM channel state machine shall immediately switch to state COMM_FULL_COMMUNICATION. ()

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

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

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

7.2.2 Behaviour in state COMM_SILENT_COMMUNICATION

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

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

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

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

[SWS_ComM_00877] In state COMM_SILENT_COMMUNICATION and user requests COMM_FULL_COMMUNICATION and communication limitation is
disabled (see Section 7.3.1), the ComM channel state machine shall switch to state COMM_FULL_COMMUNICATION.

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

Rationale for SWS_ComM_00878: A potential communication limitation (see Section 7.3.1) 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 See ComM_Nm_NetworkMode() SWS_ComM_00390, the ComM channel state machine shall switch to state COMM_FULL_COMMUNICATION and sub-state COMM_FULL_COM_READY_SLEEP.

7.2.3 Behaviour in state COMM_FULL_COMMUNICATION

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

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

[SWS_ComM_00069] On entering state COMM_FULL_COMMUNICATION the ComM channel state machine shall switch on the transmission and reception capability. This shall be performed by the ComM channel state machine requesting the corresponding Communication Mode from the Bus State Manager module (XXSM_RequestComMode(network:=<channel state machine's network>, mode:= COMM_FULL_COMMUNICATION) SWS_ComM_00829).
Rationale for SWS_ComM_00637: The COMM_FULL_COMMUNICATION mode permits sending and receiving of bus communication PDUs for the corresponding channels.

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

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

[SWS_ComM_00826] In COMM_FULL_COMMUNICATION and configuration parameter ComMNmVariant=FULL|PASSIVE (ECUC_ComM_00568) and the Network Manager module indicates ComM_Nm_PrepareBusSleepMode() (SWS_ComM_00391), the ComM state machine shall switch to state COMM_SILENT_COMMUNICATION.

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

7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state

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

[SWS_ComM_00665] On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED from COMM_NO_COM_REQUEST_PENDING and EcuM module has indicated a wake-up by ComM_EcuM_WakeUpIndication(<channel>) (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication(<PNC>) (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management.

[SWS_ComM_00902] On entering sub-state COMM_FULL_COM_NETWORK_REQUESTED and Nm module has indicated a restart, ComM_Nm_RestartIndication(<channel>) (SWS_ComM_00792), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management.
[SWS_ComM_00903] [On entering sub-state
COMM_FULL_COM_NETWORK_REQUESTED and Nm module has indicated a Network start, ComM_Nm_NetworkStartIndication(<channel>) (SWS_ComM_00383), the ComM module shall request Nm_PassiveStartup(<channel>) from the Network Management]

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

[SWS_ComM_00869] [On entering sub-state
COMM_FULL_COM_NETWORK_REQUESTED from another state or substate, if configuration parameter ComMNmVariant=FULL (ECUC_ComM_00568) and if a user has requested ComM_RequestComMode(<user>, COMM_FULL_COMMUNICATION) (SWS_ComM_00110) the ComM module shall request Nm_NetworkRequest(<channel>) from the Network Management for the corresponding NM channel.] (SRS_ModeMgm_00049)

Note: Additionally Nm_NetworkRequest may be invoked due to SWS_ComM_00980.

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

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

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

[SWS_ComM_00888] [In sub-state COMM_FULL_COM_NETWORK_REQUESTED and configuration parameter ComMNmVariant=FULL (ECUC_ComM_00568) and no user request]
ComM_RequestComMode(<user>, COMM_FULL_COMMUNICATION) and the DCM does not indicate ComM_DCM_ActiveDiagnostic(<channel>) (SWS_ComM_00873), the ComM channel state machine shall switch to sub-state COMM_FULL_COM_READY_SLEEP.  

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

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

[SWS_ComM_00890] In sub-state COMM_FULL_COMM_NETWORK_REQUESTED and the DCM does not indicate ComM_DCM_ActiveDiagnostic(<channel>) (SWS_ComM_00873) and communication limitation is requested (see section 7.3.1), ComM channel state machine shall immediately switch to sub-state COMM_FULL_COM_READY_SLEEP and cancel the timer for ComMTMinFullComModeDuration.

7.2.3.2 COMM_FULL_COM_READY_SLEEP sub-state

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

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

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

[SWS_ComM_00671] In sub-state COMM_FULL_COM_READY_SLEEP and configuration parameter ComMBusType=COMM_BUS_TYPE_INTERNAL (ECUC_ComM_00567), the ComM channel state machine shall immediately switch to state COMM_NO_COMMUNICATION.
In sub-state **COMM_FULL_COM_READY_SLEEP** and a user request **COMM_FULL_COMMUNICATION** and communication limitation is disabled (see Section 7.3.1), the ComM channel state machine shall immediately switch to sub-state **COMM_FULL_COM_NETWORK_REQUESTED**.

**Rationale for SWS_ComM_00883:** A potential communication limitation (see Section 7.3.1) shall temporarily be inactive during an active diagnostic session, see SWS_ComM_00182.

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

**Rationale for SWS_ComM_00883:** A potential communication limitation (see Section 7.3.1) shall temporarily be inactive during an active diagnostic session, see SWS_ComM_00182.

In sub-state **COMM_FULL_COM_READY_SLEEP** and configuration parameter ComMNmVariant=LIGHT (ECUC_ComM_00568) and a switch to sub-state **COMM_FULL_COM_NETWORK_REQUESTED**, due to request for **COMM_FULL_COMMUNICATION** according to requirements in SWS_ComM_00882 or SWS_ComM_00883, the timer for ComMNmLightTimeout (ECUC_ComM_00606) shall be canceled.

### 7.3 Extended functionality

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

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

Use Case: Change the wakeup inhibition via diagnostics.

**Comment:** Configurable with parameter ComMEcuGroupClassification (see ECUC_ComM_00563).

### 7.3.1 Communication inhibition

**Note:**
1. The purpose of mode inhibition is to limit the communication capabilities. For details see Section 7.3.1.1 and Section 7.3.1.2.
2. The following parameters are relevant to communication inhibition and have relationship to APIs described below.
• ComMNoCom: "request bit" of mode inhibition (limit to NoCom), can be controlled by ComM_LimitChannelToNoComMode() and ComM_LimitECUToNoComMode(), only if ComMEcuGroupClassification enable 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 enable 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_00301] [The ComM module shall offer interfaces to request and release the corresponding mode inhibitions.] ()

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

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

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

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

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

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

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

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

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

Comment: The DCM indicates the start of an active diagnostic session with ComM_DCM_ActiveDiagnostic(<channel>)(SWS_ComM_00873) and the end of a diagnostic session with ComM_DCM_InactiveDiagnostic(<channel>)(SWS_ComM_00874).
7.3.1.1 Bus wake up inhibition

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

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

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

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

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

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

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

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

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

[SWS_ComM_00066] [The ComM module shall never inhibit the “passive wake-up” capability.] ()
Rationale for **SWS_ComM_00066**: It must be always possible to react on bus wake ups indicated by the EcuM module.

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

[SWS_ComM_00157] [ComMNWakeup status must be stored non volatile.] ()

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

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

7.3.1.2 Limit to `COMM_NO_COMMUNICATION` mode

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

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

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

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

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

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

*Note:* **SWS_COMM_00841** refers only to the state machine transitions. This means, other actions like update of the inhibition status due to a limit to `COMM_NO_COMMUNICATION` shall always be performed independent of the current state.
The ComM module shall ignore requests for limit to COMM_NO_COMMUNICATION in other states than COMM_FULL_COM_NETWORK_REQUESTED.

Note: SWS_COMM_00841 and SWS_COMM_00842 describe the behaviour if a local ComM user requests FULL_COM (active request) for a dedicated ComM channel. This means, limit to COMM_NO_COMMUNICATION shall only be performed if a channel was request actively. The limit to no communication shall not be performed, if a ComM channel is remotely kept awake due to a passive wakeup.

All active user requests for communication channel X shall be ignored if the ComM Inhibition ComMNoCom=TRUE (see ECUC_ComM_00571) for the corresponding channel to guarantee entering the COMM_NO_COMMUNICATION state for channel X.

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

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

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

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

7.4 Bus communication management

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

The ComM module shall omit calls to control the communication capabilities if configuration parameter ComMBusType=COMM_BUS_TYPE_INTERNAL (ECUC_ComM_00567). (SRS_ModeMgm_09168)

Rationale for SWS_ComM_00664: Internal communication has no corresponding bus interface.
7.5 Network management dependencies

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

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

<table>
<thead>
<tr>
<th>NM variant</th>
<th>Keep bus awake capability</th>
<th>Shutdown synchronization</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td></td>
<td>No shutdown synchronization by ComM. Shutdown by switching off the power of the ECU.</td>
</tr>
<tr>
<td>LIGHT</td>
<td></td>
<td>Shutdown synchronization by ComM with means of a timeout (configured with ComMNmLightTimeout, ECUC_ComM_00606)</td>
</tr>
<tr>
<td>PASSIVE</td>
<td>ECU is not allowed to keep the bus awake</td>
<td>Shutdown synchronization by ComM with means of AUTOSAR NM.</td>
</tr>
<tr>
<td>FULL</td>
<td>ECU is allowed to keep the bus awake.</td>
<td>Shutdown synchronization by ComM with means of AUTOSAR NM.</td>
</tr>
</tbody>
</table>

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

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

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

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

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

Rationale for SWS_ComM_00667: Service Nm_NetworkRequest() is not available.
7.6 Bus error management

7.6.1 Network Start Indication

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

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

7.7 Test support requirements

7.7.1 Inhibited Full Communication Request Counter

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

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

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

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

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

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

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

7.8.1 Development errors

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

<table>
<thead>
<tr>
<th>Type or error</th>
<th>Relevance</th>
<th>Related error code</th>
<th>Value [hex]</th>
</tr>
</thead>
<tbody>
<tr>
<td>API service used without module initialization</td>
<td>Development</td>
<td>COMM_E_UNINIT</td>
<td>0x1</td>
</tr>
<tr>
<td>API service used with wrong parameters</td>
<td>Development</td>
<td>COMM_E_WRONG_PARAMETERS</td>
<td>0x2</td>
</tr>
<tr>
<td>API Service used with a null pointer</td>
<td>Development</td>
<td>COMM_E_PARAM_POINTER</td>
<td>0x3</td>
</tr>
<tr>
<td>Initialization failed</td>
<td>Development</td>
<td>COMM_E_INIT FAILED</td>
<td>0x4</td>
</tr>
</tbody>
</table>

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

[SWS_ComM_00612] If ComM is not initialized, all ComM module and all API service other than ComM_Init() (see SWS_ComM_00146), ComM_GetVersionInfo() (see SWS_COMM_00370) and ComM_GetStatus() (see SWS_COMM_00242); shall:
- not execute their normal operation,
- and return E_NOT_OK, if it has a standard return type.]

[SWS_ComM_00858] 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 otherwise (if DET is disabled) return E_NOT_OK. ]

7.8.2 Runtime Errors

There are no runtime errors.

7.8.3 Transient Faults

There are no transient faults.

7.9 Non functional requirements

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

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

7.10 Communication Manager Module Services

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

7.10.1 Architecture

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

![Diagram showing SW-C requests state changes to the Communication Manager Module](image-url)
7.10.2.3 SW-Cs explicitly wants to take influence on its communication state

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

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

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

```c
MySW-CRunnable_Init(self) {
    // SW-C wants to send and receive data
    e = Rte_Call_comRequest_RequestComMode(COMM_FULL_COMMUNICATION);
}
```
if (e == RTE_E_OK) {
    // successfully requested the Com Manager Module to move to
    // full communication mode
} else {
    // an error occurred when
    // interacting with the Com Manager module
    if (e == E_MODE_LIMITATION) {
        // a current ComMMode limitation forbids going into
        // that mode;
        // let’s ask what the maximal allowed ComMMode is
        Rte_Call_comRequest_GetMaxComMode(&max);
        if (max==COMM_NO_COMMUNICATION) {
            ...
        } else {
            // a 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 …
            }
        }
    } else {
        // a more serious error occurred …
    }
};

MySW-CRunnable_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.
7.10.3 Specification of Ports and Port Interfaces

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

7.10.3.1 Types used by the interfaces
See 8.7.4 Implementation Data Types

7.10.3.2 Ports and Port Interface for User Requests

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

7.10.3.2.2 Data Types
No data types are needed for this interface.

7.10.3.2.3 Port interface ComM_UserRequest
See 8.7.2.4 ComM_UserRequest

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

7.10.3.3.1 General approach

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

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

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

7.10.3.3.2 Port interface ComM_CurrentMode
See 8.7.3.1 ComM_CurrentMode.

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

7.10.3.4.1 General approach

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

Rationale for SWS_ComM_00734: A SW-C that wants to get informed about, which users are currently requesting FULL_COM requires the sender-receiver interface ComM_CurrentChannelRequest.'
Whenever the set of ComM users currently requesting FULL_COMM for a channel changes, the Communication Manager Module shall update the data element fullComRequestors. A change shall update the data element only, when the Communication Manager Module accepts the communication request of the ComM user.

*Rationale for SWS_ComM_00736:* Requests rejected because of active ModeLimitations will not lead to an update of the data element.

### 7.10.3.4.2 Data Types

See 8.7.4.4 ComM_UserHandleArrayType.

### 7.10.3.4.3 Port Interface ComM_CurrentChannelRequest

See 8.7.1.1 ComM_CurrentChannelRequest.

### 7.10.3.5 Ports and Port Interface for ECU Mode Limitation

#### 7.10.3.5.1 General approach

The Communication Manager Module can be configured to have an AUTOSAR port providing the client-server interface ComM_ECUModeLimitation.

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

#### 7.10.3.5.2 Port interface ComM_ECUModeLimitation

See 8.7.2.3 ComM_ECUModeLimitation.

### 7.10.3.6 Ports and Port Interface for Channel Wake up

#### 7.10.3.6.1 General approach

The Communication Manager Module can be configured to have an AUTOSAR port providing the Client-Server Interface ComM_ChannelWakeup.

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

7.10.3.6.2 Port interface ComM_ChannelWakeup

See 8.7.2.2 ComM_ChannelWakeup.

7.10.3.7 Ports and Port Interface for interface Channel Limitation

7.10.3.7.1 General approach

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

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

7.10.3.7.2 Port interface ComM_ChannelLimitation

See 8.7.2.1 ComM_ChannelLimitation.

7.10.3.8 Definition of the Service of the Communication Manager Module

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

Comment: Note that these definitions can only be completed during ECU configuration (because it depends on certain configuration parameters of the Communication Manager Module, which determine the number of ports provided by the Communication Manager Module service). Also note that the implementation of an SW-C does not depend on these definitions.
[SWS_ComM_00744]

/* This is the definition of the Communication Manager Module as a service. 
This is the 'outside-view' of the Communication Manager Module */

Service ComM {
    // port present if ComMModeLimitationEnabled (see ECUC_ComM_00560)
    ProvidePort ComM_ECUModeLimitation modeLimitation;

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

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

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

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

7.10.4 Runnables and Entry points

7.10.4.1 Internal behaviour

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

[SWS_ComM_00745]

InternalBehavior of the Communication Manager Module
{
    // Runnable entities of the Communication Manager Module
    RunnableEntity LimitECUToNoComMode
    symbol "ComM_LimitECUToNoComMode" /* see SWS_ComM_00124*/
RunnableEntity ReadInhibitCounter
symbol "ComM_ReadInhibitCounter" /* see SWS_ComM_00224 */
canBeInvokedConcurrently = FALSE

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

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

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

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

RunnableEntity PreventWakeup
symbol "ComM_PreventWakeup"
canBeInvokedConcurrently = FALSE

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

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

RunnableEntity GetRequestedComMode
symbol "ComM_GetRequestedComMode"
canBeInvokedConcurrently = TRUE

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

// the following applies if ComMModeLimitationEnabled
// (see ECUC_ComM_00560)
modelLimitation.LimitECUToNoComMode -> LimitECUToNoComMode
modelLimitation.ReadInhibitCounter -> ReadInhibitCounter
modelLimitation.ResetInhibitCounter -> ResetInhibitCounter
modelLimitation.SetECUGroupClassification -> SetECUGroupClassification

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

...
CLnnn.LimitChannelToNoComMode -> LimitChannelToNoComMode
CLnnn.GetInhibitionStatus -> GetInhibitionStatus
PortArgument {port=CLnnn,
value.type=NetworkHandleType,
value.value=Channel[nnn].COMM_CHANNEL_ID}

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

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

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

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

8.1 Imported types

8.1.1 Standard types

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Imported Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComStack_Types</td>
<td>NetworkHandleType</td>
</tr>
<tr>
<td></td>
<td>PNCHandleType</td>
</tr>
<tr>
<td>EcuM</td>
<td>EcuM_StateType</td>
</tr>
<tr>
<td>NvM</td>
<td>NvM_BlockIdType</td>
</tr>
<tr>
<td></td>
<td>NvM_RequestResultType</td>
</tr>
<tr>
<td>Std_Types</td>
<td>Std_ReturnType</td>
</tr>
<tr>
<td></td>
<td>Std_VersionInfoType</td>
</tr>
</tbody>
</table>

(SRS_BSW_00348, SRS_BSW_00357)

The ComM API uses the following extension to Std_ReturnType:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM_E_MODE_LIMITATION</td>
<td>Function call has been successfully but mode can not be granted because of mode inhibition.</td>
</tr>
</tbody>
</table>

(SRS_BSW_00331, SRS_BSW_00369, SRS_BSW_00377, SRS_BSW_00441)

8.2 Type definitions

8.2.1 ComM_InitStatusType

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComM_InitStatusType</td>
<td>Enumeration</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM_UNINIT</td>
<td>The COM Manager is not initialized or not usable. This shall be the default value after reset. This status shall have the value 0.</td>
</tr>
<tr>
<td>COMM_INIT</td>
<td>The COM Manager is initialized and usable.</td>
</tr>
</tbody>
</table>

(SRS_BSW_00441)
8.2.2 ComM_PncModeType

[SWS_ComM_00673] [ ]

<table>
<thead>
<tr>
<th>Name:</th>
<th>ComM_PncModeType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Enumeration</td>
</tr>
<tr>
<td>Range:</td>
<td></td>
</tr>
<tr>
<td>COMM_PNC_REQUESTED</td>
<td>0x00</td>
</tr>
<tr>
<td>COMM_PNC_READY_SLEEP</td>
<td>0x01</td>
</tr>
<tr>
<td>COMM_PNC_PREPARE_SLEEP</td>
<td>0x02</td>
</tr>
<tr>
<td>COMM_PNC_NO_COMMUNICATION</td>
<td>0x03</td>
</tr>
</tbody>
</table>

Description: Current mode of a PNC

8.2.3 ComM_StateType

[SWS_ComM_00674] [ ]

<table>
<thead>
<tr>
<th>Name:</th>
<th>ComM_StateType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>uint8</td>
</tr>
<tr>
<td>Range:</td>
<td></td>
</tr>
<tr>
<td>COMM_NO_COM_NO_PENDING_REQUEST</td>
<td>0</td>
</tr>
<tr>
<td>COMM_NO_COM_REQUEST_PENDING</td>
<td>1</td>
</tr>
<tr>
<td>COMM_FULL_COM_NETWORK_REQUESTED</td>
<td>2</td>
</tr>
<tr>
<td>COMM_FULL_COM_READY_SLEEP</td>
<td>3</td>
</tr>
<tr>
<td>COMM_SILENT_COM</td>
<td>4</td>
</tr>
</tbody>
</table>

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: Communication Mode='Silent Communication'

8.2.4 ComM_ConfigType

[SWS_ComM_00162] [ ]

<table>
<thead>
<tr>
<th>Name:</th>
<th>ComM_ConfigType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type:</td>
<td>Structure</td>
</tr>
<tr>
<td>Range:</td>
<td>implementation specific</td>
</tr>
</tbody>
</table>

Description: This type contains the implementation-specific post build configuration structure.

8.3 Function definitions

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

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

- ComM_Init
- ComM_GetVersionInfo
### 8.3.1 ComM_Init

**[SWS_ComM_00146]**

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

(SRS_BSW_00101, SRS_BSW_00358, SRS_BSW_00414)

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

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

### 8.3.2 ComM_DeInit

**[SWS_ComM_00147]**

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

(SRS_BSW_00336)

**[SWS_ComM_00794]** De-initialization in ComM_DeInit() shall only be performed if all channels controlled by the ComM module are in COMM_NO_COMMUNICATION mode.
Rationale for **SWS_ComM_00794**: Since the ComM_DeInit() API cannot return an error message, it must be assured that all channels are in COMM_NO_COMMUNICATION mode and COMM_NO_COM_NO_PENDING_REQUEST sub-state before ComM_DeInit() is called. E.g. the state should be checked with ComM_GetState(Channel,…) and ComM_CommunicationAllowed(Channel,TRUE) cannot be called before ComM_DeInit() has been called.] ()

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

8.3.3 ComM_GetState

[SWS_ComM_00872] [ ]

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

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

[SWS_ComM_00242] [ ]

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

(SRS_BSW_00406)

8.3.5 ComM_GetInhibitionStatus

[SWS_ComM_00619] [ ]

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_GetInhibitionStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>Std_ReturnType ComM_GetInhibitionStatus(</td>
</tr>
<tr>
<td></td>
<td>NetworkHandleType Channel,</td>
</tr>
<tr>
<td></td>
<td>ComM_InhibitionStatusType* Status</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td>Service ID[hex]:</td>
<td>0x04</td>
</tr>
<tr>
<td>Sync/Async:</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Reentrancy:</td>
<td>Non Reentrant</td>
</tr>
<tr>
<td>Parameters (in):</td>
<td>Channel</td>
</tr>
<tr>
<td></td>
<td>See NetworkHandleType</td>
</tr>
<tr>
<td>Parameters (inout):</td>
<td>None</td>
</tr>
<tr>
<td>Parameters (out):</td>
<td>Status</td>
</tr>
<tr>
<td></td>
<td>See ComM_InhibitionStatusType</td>
</tr>
<tr>
<td>Return value:</td>
<td>Std_ReturnType</td>
</tr>
<tr>
<td></td>
<td>E_OK: Successfully returned Inhibition Status</td>
</tr>
<tr>
<td></td>
<td>E_NOT_OK: Return of Inhibition Status failed</td>
</tr>
<tr>
<td>Description:</td>
<td>Returns the inhibition status of a ComM channel.</td>
</tr>
</tbody>
</table>

()]

8.3.6 ComM_RequestComMode

[SWS_ComM_00110] [ ]

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_RequestComMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>Std_ReturnType ComM_RequestComMode(</td>
</tr>
<tr>
<td></td>
<td>ComM_UserHandleType User,</td>
</tr>
<tr>
<td></td>
<td>ComM_ModeType ComMode</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td>Service ID[hex]:</td>
<td>0x05</td>
</tr>
</tbody>
</table>
### Specification of Communication Manager

**AUTOSAR CP Release 4.3.1**

**Document ID:** AUTOSAR_SWS_COMManager

<table>
<thead>
<tr>
<th><strong>Sync/Async:</strong></th>
<th>Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reentrancy:</strong></td>
<td>Reentrant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Parameters (in):</strong></th>
<th><strong>Parameters (out):</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>User Handle of the user who requests a mode</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Parameters (inout):</strong></th>
<th><strong>Return value:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Std_ReturnType</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Parameters (out):</strong></th>
<th><strong>Return value:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ComMode</strong></td>
<td>Std_ReturnType</td>
</tr>
</tbody>
</table>

**Description:**
Requesting of a Communication Mode by a user.

**Notes:**
- Internally mode COMM_SILENT_COMMUNICATION is not a valid request for a user, mode used for synchronization at shutdown.
- Valid modes are COMM_NO_COMMUNICATION and COMM_FULL_COMMUNICATION.
- The communication request could also be released due to a ComM communication inhibition.

---

### Use Case: ComM_RequestComMode

**Service name:** ComM_RequestComMode

**Syntax:**

```
Std_ReturnType ComM_RequestComMode(
    ComM_UserHandleType User,
    ComM_ModeType* ComMode
)
```

**Service ID[hex]:** 0x06

<table>
<thead>
<tr>
<th><strong>Sync/Async:</strong></th>
<th>Synchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reentrancy:</strong></td>
<td>Reentrant</td>
</tr>
<tr>
<td><strong>Parameters (in):</strong></td>
<td>User Handle of the user who requests a mode</td>
</tr>
<tr>
<td><strong>Parameters (out):</strong></td>
<td>See ComM_ModeType</td>
</tr>
</tbody>
</table>

**Return value:**

<table>
<thead>
<tr>
<th><strong>Return value:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Std_ReturnType</td>
</tr>
</tbody>
</table>

**Description:**
Function to query the maximum allowed Communication Mode of the corresponding user.

---

**Use Case:** This function provides the possibility to request the maximum possible mode (e.g. user wants to check if it is possible to get "Full Communication" mode or if a limitation/inhibition is active). This is needed for diagnosis/debugging.
[SWS_ComM_00374] If more than one channel is linked to one user request and the maximum allowed modes of the channels are different, then the function ComM_GetMaxComMode shall return the lowest mode (see SWS_ComM_00867 and SWS_ComM_00868). ()

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

8.3.8 ComM_GetRequestedComMode

[SWS_ComM_00079] Table 8.3.8-1

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_GetRequestedComMode</th>
</tr>
</thead>
</table>
| Syntax:            | Std_ReturnType ComM_GetRequestedComMode(  
|                    | ComM_UserHandleType User,  
|                    | ComM_ModeType* ComMode) |
| Service ID[hex]:   | 0x07                     |
| Sync/Async:        | Synchronous              |
| Reentrancy:        | Reentrant                |
| Parameters (in):   | User Handle of the user who requests a mode |
| Parameters (inout):| None                     |
| Parameters (out):  | ComMode Name of the requested mode |
| Return value:      | Std_ReturnType E_OK: Successfully returned requested Communication Mode  
|                    | E_NOT_OK: Return of requested Communication Mode failed |
| Description:       | Function to query the currently requested Communication Mode of the corresponding user. |

(SRS_ModeMgm_09149)

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

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

**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)

8.3.9 ComM_GetCurrentComMode

[SWS_ComM_00083] Table 8.3.9-1

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_GetCurrentComMode</th>
</tr>
</thead>
</table>
| 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 (out):
- None

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.

\[
\text{Call to Bus State Manager API: XXXSM \_GetCurrentComMode(...)}
\]

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

[SWS_ComM_00798] Configuration of ComM\_GetCurrentComMode: Relationship between users and channels. A user is statically mapped to one or more channels.

8.3.10 ComM\_PreventWakeUp

[SWS_ComM_00156]

Service name: ComM\_PreventWakeUp

Syntax:

\[
\text{Std\_ReturnType ComM\_PreventWakeUp(}
\text{NetworkHandleType Channel,}
\text{boolean Status)}
\]

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.

\[
\text{(SRS\_ModeMgm\_09157)}
\]

[SWS_ComM_00799] Configuration of ComM\_PreventWakeUp: Configurable with COMM\_WAKEUP\_INHIBITION\_ENABLED (see ECUC\_ComM_00559).

} (SRS\_ModeMgm\_09172)
### 8.3.11 ComM_LimitChannelToNoComMode

[SWS_ComM_00163] [ ]

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_LimitChannelToNoComMode</th>
</tr>
</thead>
</table>
| Syntax             | Std_ReturnType ComM_LimitChannelToNoComMode(  
|                    |   NetworkHandleType Channel,  
|                    |   boolean Status )           |
| Service ID[hex]    | 0x0b                         |
| Sync/Async         | Synchronous                  |
| Reentrancy         | Non Reentrant                |
| Parameters (in)    | Channel | 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_LimitECUToNoComMode, same functionality but for all  
|                    |   channels)                      |

[SWS_ComM_00800] [ ]

Configuration of ComM_LimitChannelToNoComMode:  
Configurable with ComMModeLimitationEnabled (see ECUC_ComM_00560) and  
COMM_RESET_AFTER_FORCING_NO_COMM (see ECUC_ComM_00558).] ()

### 8.3.12 ComM_LimitECUToNoComMode

[SWS_ComM_00124] [ ]

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_LimitECUToNoComMode</th>
</tr>
</thead>
</table>
| 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)        |
8.3.13 ComM_ReadInhibitCounter

[SWS_ComM_00224] [ ]

Service name: ComM_ReadInhibitCounter

Syntax: Std_ReturnType ComM_ReadInhibitCounter(
            uint16* CounterValue
        )

Service ID[hex]: 0x0d

Sync/Async: Synchronous

Reentrancy: Non Reentrant

Parameters (in): None

Parameters (inout): None

Parameters (out): CounterValue

Return value: Std_ReturnType

Description: This function returns the amount of rejected COMM_FULL_COMMUNICATION user requests.

] (SRS_ModeMgm_09156)

[SWS_ComM_00802] [ ]

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

8.3.14 ComM_ResetInhibitCounter

[SWS_ComM_00108] [ ]

Service name: ComM_ResetInhibitCounter

Syntax: Std_ReturnType ComM_ResetInhibitCounter(
            void
        )

Service ID[hex]: 0x0e

Sync/Async: Synchronous

Reentrancy: Non Reentrant

Parameters (in): None

Parameters (inout): None

Parameters (out): None

Return value: Std_ReturnType

Description: This function resets the Inhibited COMM_FULL_COMMUNICATION request Counter.

] (SRS_ModeMgm_09156)
8.3.15 ComM_SetECUGroupClassification

Service name: ComM_SetECUGroupClassification

Syntax: Std_ReturnType ComM_SetECUGroupClassification(
          ComM_InhibitionStatusType Status
);  

Service ID[hex]: 0x0f
Sync/Async: Synchronous
Reentrancy: Non Reentrant
Parameters (in): Status See ComM_InhibitionStatusType
Parameters (inout): None
Parameters (out): None

Return value: Std_ReturnType
E_OK: Successfully change the ECU Group Classification Status
E_NOT_OK: Change of the ECU Group Classification Status failed

Description: Changes the ECU Group Classification status (see chapter 10.2.2)

8.3.16 ComM_GetVersionInfo

Service name: ComM_GetVersionInfo

Syntax: void ComM_GetVersionInfo(
          Std_VersionInfoType* Versioninfo
);  

Service ID[hex]: 0x10
Sync/Async: Synchronous
Reentrancy: Reentrant
Parameters (in): None
Parameters (inout): None
Parameters (out): Versioninfo See Std_VersionInfoType

Return value: None

Description: This function returns the published information (for details refer to table 10.3)
[SWS_ComM_00620] All the provided indication functions shall be implemented pre-compile time.

Note: All functions in this chapter requires that the ComM module is initialized correctly.

8.4.1 AUTOSAR Network Management Interface

8.4.1.1 ComM_Nm_NetworkStartIndication

<table>
<thead>
<tr>
<th>Service name</th>
<th>ComM_Nm_NetworkStartIndication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>void ComM_Nm_NetworkStartIndication(</td>
</tr>
<tr>
<td></td>
<td>NetworkHandleType Channel)</td>
</tr>
<tr>
<td>Service ID[hex]</td>
<td>0x15</td>
</tr>
<tr>
<td>Sync/Async</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Reentrancy</td>
<td>Reentrant</td>
</tr>
<tr>
<td>Parameters (in)</td>
<td>Channel</td>
</tr>
<tr>
<td>Parameters (inout)</td>
<td>See NetworkHandleType</td>
</tr>
<tr>
<td>Parameters (out)</td>
<td>None</td>
</tr>
<tr>
<td>Return value</td>
<td>None</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
</tbody>
</table>

} ()
8.4.1.2 ComM_Nm_NetworkMode

<table>
<thead>
<tr>
<th>Service name:</th>
<th>ComM_Nm_NetworkMode</th>
</tr>
</thead>
</table>
| 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. |

8.4.1.3 ComM_Nm_PrepareBusSleepMode

<table>
<thead>
<tr>
<th>Service name:</th>
<th>ComM_Nm_PrepareBusSleepMode</th>
</tr>
</thead>
</table>
| 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) |

8.4.1.4 ComM_Nm_BusSleepMode

<table>
<thead>
<tr>
<th>Service name:</th>
<th>ComM_Nm_BusSleepMode</th>
</tr>
</thead>
</table>
| 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. |
### 8.4.1.5 ComM_Nm_RestartIndication

<table>
<thead>
<tr>
<th>Service name:</th>
<th>ComM_Nm_RestartIndication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>void ComM_Nm_RestartIndication(</td>
</tr>
<tr>
<td></td>
<td>NetworkHandleType Channel</td>
</tr>
<tr>
<td>Service ID[hex]:</td>
<td>0x1b</td>
</tr>
<tr>
<td>Sync/Async:</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Reentrancy:</td>
<td>Reentrant</td>
</tr>
<tr>
<td>Parameters (in):</td>
<td>Channel</td>
</tr>
<tr>
<td>Parameters (in/out):</td>
<td>None</td>
</tr>
<tr>
<td>Parameters (out):</td>
<td>None</td>
</tr>
<tr>
<td>Return value:</td>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
<td>If NmIf has started to shut down the coordinated busses, AND not all coordinated busses have indicated bus sleep state, AND on at least 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.</td>
</tr>
</tbody>
</table>

### 8.4.2 AUTOSAR Diagnostic Communication Manager Interface

#### 8.4.2.1 ComM_DCM_ActiveDiagnostic

<table>
<thead>
<tr>
<th>Service name:</th>
<th>ComM_DCM_ActiveDiagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>void ComM_DCM_ActiveDiagnostic(</td>
</tr>
<tr>
<td></td>
<td>NetworkHandleType Channel</td>
</tr>
<tr>
<td>Service ID[hex]:</td>
<td>0x1f</td>
</tr>
<tr>
<td>Sync/Async:</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Reentrancy:</td>
<td>Reentrant</td>
</tr>
<tr>
<td>Parameters (in):</td>
<td>Channel</td>
</tr>
<tr>
<td>Parameters (in/out):</td>
<td>None</td>
</tr>
<tr>
<td>Parameters (out):</td>
<td>None</td>
</tr>
<tr>
<td>Return value:</td>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
<td>Indication of active diagnostic by the DCM.</td>
</tr>
</tbody>
</table>

#### 8.4.2.2 ComM_DCM_InactiveDiagnostic

<table>
<thead>
<tr>
<th>Service name:</th>
<th>ComM_DCM_InactiveDiagnostic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>void ComM_DCM_InactiveDiagnostic(</td>
</tr>
<tr>
<td></td>
<td>NetworkHandleType Channel</td>
</tr>
<tr>
<td>Service ID[hex]:</td>
<td>0x20</td>
</tr>
<tr>
<td>Sync/Async:</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Reentrancy:</td>
<td>Reentrant</td>
</tr>
<tr>
<td>Parameters (in):</td>
<td>Channel</td>
</tr>
<tr>
<td>Parameters (in/out):</td>
<td>None</td>
</tr>
<tr>
<td>Parameters (out):</td>
<td>None</td>
</tr>
<tr>
<td>Return value:</td>
<td>None</td>
</tr>
<tr>
<td>Description:</td>
<td>Channel no longer needed for Diagnostic communication</td>
</tr>
</tbody>
</table>
### 8.4.3 AUTOSAR ECU State Manager Interface

#### 8.4.3.1 ComM_EcuM_WakeUpIndication

**Service name:** ComM_EcuM_WakeUpIndication  
**Syntax:**
```c
void ComM_EcuM_WakeUpIndication(
    NetworkHandleType Channel
)
```

- **Service ID** (hex): 0x2a
- **Sync/Async**: Synchronous
- **Reentrancy**: Reentrant
- **Parameters (in)**: Channel
- **Parameters (out)**: None
- **Return value**: None
- **Description**: Notification of a wake up on the corresponding channel.

#### 8.4.3.2 ComM_EcuM_PNCWakeUpIndication

**Service name:** ComM_EcuM_PNCWakeUpIndication  
**Syntax:**
```c
void ComM_EcuM_PNCWakeUpIndication(
    PNCHandleType PNCid
)
```

- **Service ID** (hex): 0x37
- **Sync/Async**: Synchronous
- **Reentrancy**: Reentrant
- **Parameters (in)**: PNCid
- **Parameters (out)**: None
- **Return value**: None
- **Description**: Notification of a wake up on the corresponding partial network cluster.

### 8.4.4 AUTOSAR ECU State Manager and Basic Software Mode Manager Interface

#### 8.4.4.1 ComM_CommunicationAllowed

**Service name:** ComM_CommunicationAllowed  
**Syntax:**
```c
void ComM_CommunicationAllowed(
    NetworkHandleType Channel,
    boolean Allowed
)
```
### Specification of Communication Manager

**AUTOSAR CP Release 4.3.1**

#### Service ID[hex]: 0x35
- **Sync/Async:** Synchronous
- **Reentrancy:** Non Reentrant

**Parameters (in):**
- Channel
- Allowed
  - TRUE: Communication is allowed
  - FALSE: Communication is not allowed

**Parameters (inout):** None

**Parameters (out):** None

**Return value:** None

**Description:**
EcuM or BswM shall indicate to ComM when communication is allowed.
If EcuM/Fixed is used: EcuM/Fixed.
If EcuM/Flex is used: BswM.

---

#### 8.4.5 Bus State Manager Interface

#### 8.4.5.1 ComM_BusSM_ModeIndication [SWS_ComM_00675] [SWS_ComM_00819]

**Service name:** ComM_BusSM_ModeIndication

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

**Service ID[hex]: 0x33**
- **Sync/Async:** Asynchronous
- **Reentrancy:** Reentrant

**Parameters (in):**
- Channel
  - See NetworkHandleType
- ComMode
  - See ComM_ModeType

**Parameters (inout):** None

**Parameters (out):** None

**Return value:** None

**Description:**
Indication of the actual bus mode by the corresponding Bus State Manager.
ComM shall propagate the indicated state to the users with means of the RTE and BswM.

---

#### 8.4.6 COM Interface

[SWS_ComM_00819]

**Service name:** ComM_COMCbk_<sn>

**Syntax:**
```c
void ComM_COMCbk_<sn>()
```

**Service ID[hex]: 0x36**
- **Sync/Async:** Synchronous
- **Reentrancy:** Non Reentrant

**Parameters (in):** None

**Parameters (inout):** None

**Parameters (out):** None
### 8.5 Scheduled functions

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

#### 8.5.1 ComM_MainFunction

**Service name:** ComM_MainFunction_<Channel_Id>

**Syntax:**

```c
void ComM_MainFunction_<Channel_Id>(
    void
)
```

**Service ID[hex]:** 0x60

**Description:**

This function shall perform the processing of the AUTOSAR ComM activities that are not directly initiated by the calls e.g. from the RTE. There shall be one dedicated Main Function for each channel of ComM.

**Precondition:** ComM shall be initialized.

(SRS_BSW_00373)

### 8.6 Expected interfaces

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

#### 8.6.1 Mandatory Interfaces

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

**API function**

<table>
<thead>
<tr>
<th>API function</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nm_PassiveStartUp</td>
<td>Nm</td>
<td>This function calls the <code>&lt;BusNm&gt;_PassiveStartUp</code> function (e.g. CanNm_PassiveStartUp function is called if channel is configured as CAN).</td>
</tr>
<tr>
<td>Nm_NetworkRequest</td>
<td>Nm</td>
<td>This function calls the <code>&lt;BusNm&gt;_NetworkRequest</code> function (e.g. CanNm_NetworkRequest function is called if channel is configured as CAN).</td>
</tr>
<tr>
<td>Specification of Communication Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTOSAR CP Release 4.3.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Function</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nm_NetworkRelease</td>
<td>Nm</td>
<td>This function calls the &lt;BusNm&gt;_NetworkRelease bus specific function (e.g. CanNm_NetworkRelease function is called if channel is configured as CAN).</td>
</tr>
<tr>
<td>Dcm_ComM_NoComModeEntered</td>
<td>Dcm</td>
<td>This call informs the Dcm module about a ComM mode change to COMM_NO_COMMUNICATION.</td>
</tr>
<tr>
<td>Dcm_ComM_SilentComModeEntered</td>
<td>Dcm</td>
<td>This call informs the Dcm module about a ComM mode change to COMM_SILENT_COMMUNICATION.</td>
</tr>
<tr>
<td>Dcm_ComM_FullComModeEntered</td>
<td>Dcm</td>
<td>This call informs the Dcm module about a ComM mode change to COMM_FULL_COMMUNICATION.</td>
</tr>
<tr>
<td>Rte_Ports_UserMode_P()[n].Switch_currentMode(RTE_MODE_ComMMode_Comm_NO_COMMUNICATION)</td>
<td>Rte</td>
<td>Indicate COMM_NO_COMMUNICATION mode to RTE.</td>
</tr>
<tr>
<td>Rte_Ports_UserMode_P()[n].Switch_currentMode(RTE_MODE_ComMMode_Comm_SILENT_COMMUNICATION)</td>
<td>Rte</td>
<td>Indicate COMM_SILENT_COMMUNICATION mode to RTE.</td>
</tr>
<tr>
<td>Rte_Ports_UserMode_P()[n].Switch_currentMode(RTE_MODE_ComMMode_Comm_FULL_COMMUNICATION)</td>
<td>Rte</td>
<td>Indicate COMM_FULL_COMMUNICATION mode to RTE.</td>
</tr>
<tr>
<td>BswM_ComM_CurrentMode</td>
<td>BswM</td>
<td>Indicate Communication Mode to BswM.</td>
</tr>
<tr>
<td>NvM_ReadBlock</td>
<td>NvM</td>
<td>NVRAM manager API for Read block.</td>
</tr>
<tr>
<td>NvM_WriteBlock</td>
<td>NvM</td>
<td>NVRAM manager API for Write block.</td>
</tr>
<tr>
<td>NvM_GetErrorStatus</td>
<td>NvM</td>
<td>NVRAM manager API for Get status.</td>
</tr>
<tr>
<td>&lt;BusSM&gt;_GetCurrentComMode</td>
<td>&lt;BusSM&gt;</td>
<td>Function to query the actual communication mode from the &lt;Bus&gt; State Manager.</td>
</tr>
<tr>
<td>&lt;BusSM&gt;_RequestComMode</td>
<td>&lt;BusSM&gt;</td>
<td>Function to request a communication mode from the &lt;Bus&gt; State Manager.</td>
</tr>
</tbody>
</table>

### 8.6.1.1 AUTOSAR NVRAM Manager module

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

Comment: See SWS_ComM_00864 and SWS_ComM_00865 when configuration data shall be read and stored For details refer to the AUTOSAR NVRAM Manager module Specification [7].

### 8.6.1.2 AUTOSAR Bus State Manager

[SWS_ComM_00962] The prefix for the StateManager APIs ("<BusSm>") shall be CanSM, LinSM, FrSM, EthSM if the Parameter ComMBusType is COMM_BUS_TYPE_CAN, COMM_BUS_TYPE_LIN, COMM_BUS_TYPE_FR or COMM_BUS_TYPE_ETH accordingly.] ()
[SWS_ComM_00957] If ComMBusType = "COMM_BUS_TYPE_CDD" the API prefix ("<BusSm>") shall be configured in the Parameter "ComMCDDBusPrefix".  

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

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

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

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

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

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

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

8.6.1.4 AUTOSAR Diagnostic Communication Manager Module  
[SWS_ComM_00266] The ComM module shall use the corresponding functions provided by DCM (see SWS_ComM_00828) to control the communication capabilities of the DCM module.  

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

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

8.6.1.5 AUTOSAR RTE interface provided by RTE to ComM for the SW-C  
[SWS_ComM_00091] The ComM module shall use the corresponding function provided by RTE to indicate modes to the users. There shall be one indication per user. Fan-out in case of a mode indication related to more than one user shall be done by the Communication Manager Module.  

(SRS_ModeMgm_09085)
[SWS_ComM_00663] If more than one channel is linked to one user request and the modes of the channels are different, the ComM module shall indicate the lowest mode to the user. ()

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

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

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

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

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

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

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

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

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

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

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

8.6.1.6 Basic Software Mode Manager (BswM)
The ComM module shall use the corresponding function provided by BswM to report the states of Communication Manager Module channels (see SWS_ComM_00828).

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

### 8.6.2 Optional Interfaces

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

<table>
<thead>
<tr>
<th>API function</th>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Det_ReportError</td>
<td>Det</td>
<td>Service to report development errors</td>
</tr>
<tr>
<td>BswM_ComM_CurrentPNCMode</td>
<td>BswM</td>
<td>Function called by ComM to indicate the current mode of the PNC</td>
</tr>
</tbody>
</table>

**8.6.2.1 AUTOSAR DET**

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

### 8.6.3 Configurable Interfaces

None.

### 8.6.4 AUTOSAR COM

<table>
<thead>
<tr>
<th>Service name</th>
<th>Com_SendSignal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td>uint8 Com_SendSignal(</td>
</tr>
<tr>
<td></td>
<td>Com.SignalIdType SignalId,</td>
</tr>
<tr>
<td></td>
<td>const void* SignalDataPtr )</td>
</tr>
<tr>
<td>Service ID[hex]:</td>
<td>0xa</td>
</tr>
<tr>
<td>Sync/Async:</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Reentrancy:</td>
<td>Non Reentrant for the same signal. Reentrant for different signals.</td>
</tr>
<tr>
<td>Parameters (in):</td>
<td>SignalId Id of signal to be sent.</td>
</tr>
<tr>
<td></td>
<td>SignalDataPtr Reference to the signal data to be transmitted.</td>
</tr>
<tr>
<td>Parameters (inout):</td>
<td>None</td>
</tr>
<tr>
<td>Parameters (out):</td>
<td>None</td>
</tr>
<tr>
<td>Return value:</td>
<td>E_OK: service has been accepted</td>
</tr>
<tr>
<td></td>
<td>COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped (or service failed due to development error)</td>
</tr>
<tr>
<td></td>
<td>COM_BUSY: in case the TP-Buffer is locked for large data types handling</td>
</tr>
<tr>
<td>Description:</td>
<td>The service Com_SendSignal updates the signal object identified by SignalId with</td>
</tr>
<tr>
<td>Service name</td>
<td>Com_ReceiveSignal</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>Syntax</td>
<td><code>uint8 Com_ReceiveSignal(</code></td>
</tr>
<tr>
<td></td>
<td><code>    Com_SignalIdType SignalId,</code></td>
</tr>
<tr>
<td></td>
<td><code>    void* SignalDataPtr</code>)</td>
</tr>
<tr>
<td>Service ID[hex]</td>
<td>0x0b</td>
</tr>
<tr>
<td>Sync/Async</td>
<td>Synchronous</td>
</tr>
<tr>
<td>Reentrancy</td>
<td>Non Reentrant for the same signal. Reentrant for different signals.</td>
</tr>
<tr>
<td>Parameters (in)</td>
<td>SignalId Id of signal to be received.</td>
</tr>
<tr>
<td>Parameters (inout)</td>
<td>None</td>
</tr>
<tr>
<td>Parameters (out)</td>
<td>SignalDataPtr Reference to the location where the received signal data shall be stored</td>
</tr>
<tr>
<td>Return value</td>
<td><code>uint8 E_OK: service has been accepted</code></td>
</tr>
<tr>
<td></td>
<td><code>COM_SERVICE_NOT_AVAILABLE: corresponding I-PDU group was stopped (or service failed due to development error)</code></td>
</tr>
<tr>
<td></td>
<td><code>COM_BUSY: in case the TP-Buffer is locked for large data types handling</code></td>
</tr>
<tr>
<td>Description</td>
<td>Com_ReceiveSignal copies the data of the signal identified by SignalId to the location specified by SignalDataPtr.</td>
</tr>
</tbody>
</table>

### 8.7 Service Interfaces

#### 8.7.1 Sender-Receiver-interfaces

#### 8.7.1.1 ComM_CurrentChannelRequest

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_CurrentChannelRequest_{channel_name}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>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</td>
</tr>
<tr>
<td>IsService</td>
<td>true</td>
</tr>
<tr>
<td>Variation</td>
<td><code>{ecuc(ComM/ComMConfigSet/ComMChannel/ComMF ullCommRequestNotificationEnabled}) == true</code></td>
</tr>
<tr>
<td></td>
<td><code>channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME})</code></td>
</tr>
<tr>
<td>Data Elements</td>
<td>fullComRequestors</td>
</tr>
<tr>
<td>Type</td>
<td>ComM_UserHandleArrayType_{channel_name}</td>
</tr>
<tr>
<td>Variation</td>
<td><code>channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME})</code></td>
</tr>
</tbody>
</table>
8.7.2 Client-Server-interfaces

8.7.2.1 ComM_ChannelLimitation

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

Operations

GetInhibitionStatus

<table>
<thead>
<tr>
<th>Comments</th>
<th>returns the inhibition status of a channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variation</td>
<td>--</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Status</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>--</td>
<td>ComM_InhibitionStatusType</td>
<td></td>
<td>OUT</td>
</tr>
</tbody>
</table>

| Possible Errors | E_OK | Operation successful |
|                 | E_NOT_OK | -- |

LimitChannelToNoComMode

<table>
<thead>
<tr>
<th>Comments</th>
<th>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)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variation</td>
<td>--</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Status</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FALSE: Limit channel to COMM_NO_COMMUNICATION disabled</td>
<td>boolean</td>
<td></td>
<td>IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TRUE: Limit channel to COMM_NO_COMMUNICATION enabled</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Possible Errors

<table>
<thead>
<tr>
<th>Possible Errors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_OK</td>
<td>Operation successful</td>
</tr>
<tr>
<td>E_NOT_OK</td>
<td>--</td>
</tr>
</tbody>
</table>

8.7.2.2 ComM_ChannelWakeup

**[SWS_ComM_00742]**

**Name**

ComM_ChannelWakeup

**Comment**

A SW-C playing the role of a "Mode Manager" can use this interface to configure the Communication Manager Module to take precautions against awakening other ECU's by starting the communication.

**IsService**

true

**Variation**

(ecuc(ComM/ComMGeneral.ComMWakeupInhibitionEnabled)) == true

**Possible Errors**

<table>
<thead>
<tr>
<th>Possible Errors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>E_OK</td>
</tr>
<tr>
<td>1</td>
<td>E_NOT_OK</td>
</tr>
</tbody>
</table>

**Operations**

GetInhibitionStatus

**Comments**

returns the inhibition status of a channel

**Variation**

--

**Parameters**

<table>
<thead>
<tr>
<th>Status</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ComM_InhibitionStatusType</td>
<td>--</td>
<td>OUT</td>
</tr>
</tbody>
</table>

**Possible Errors**

<table>
<thead>
<tr>
<th>Possible Errors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_OK</td>
<td>Operation successful</td>
</tr>
<tr>
<td>E_NOT_OK</td>
<td>--</td>
</tr>
</tbody>
</table>

PreventWakeUp

**Comments**

Changes the inhibition status COMM_NO_WAKEUP for the corresponding channel.

**Variation**

--

**Parameters**

<table>
<thead>
<tr>
<th>Status</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>boolean</td>
<td>--</td>
<td>IN</td>
</tr>
</tbody>
</table>
### Possible Errors

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_OK</td>
<td>Operation successful</td>
</tr>
<tr>
<td>E_NOT_OK</td>
<td>--</td>
</tr>
</tbody>
</table>

### 8.7.2.3 ComM_ECUModeLimitation

[SWS_ComM_00741]

#### Name
ComM_ECUModeLimitation

#### Comment
A SW-C which plays the role of a "Mode Manager" can use this interface to change the behavior of the entire ECU.

#### IsService
true

#### Variation
{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true

#### Possible Errors

<table>
<thead>
<tr>
<th>Error</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>E_OK</td>
</tr>
<tr>
<td>1</td>
<td>E_NOT_OK</td>
</tr>
</tbody>
</table>

### Operations

#### LimitECUToNoComMode

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

**Parameter**
- **Status**
  - **Comment**: FALSE: Limit ECU to COMM_NO_COMMUNICATION disabled
    TRUE: Limit ECU to COMM_NO_COMMUNICATION enabled
  - **Type**: boolean
  - **Variation**: --
  - **Direction**: IN

**Possible Errors**
- **E_OK**: Operation successful
- **E_NOT_OK**: --

#### ReadInhibitCounter

**Comments**
returns the value of the 'inhibited full communication request counter'

**Variation**
{ecuc(ComM/ComMGeneral.ComMGlobalNvMBlockDescriptor)} != NULL

**Parameters**
- **CounterValue**
  - **Comment**: --
  - **Type**: uint16
### Specification of Communication Manager

**AUTOSAR CP Release 4.3.1**

**Document ID:** AUTOSAR_SWS_COMManager

#### Variation --

#### Direction OUT

### Possible Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_OK</td>
<td>Operation successful</td>
</tr>
<tr>
<td>E_NOT_OK</td>
<td></td>
</tr>
</tbody>
</table>

---

### ResetInhibitCounter

**Comments**
reset the "inhibited full communication request counter"

**Variation**
\{ecuc(ComM/ComMGeneral.ComMGlobalNvMBlockDescriptor) \} != NULL

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_OK</td>
<td>Operation successful</td>
</tr>
<tr>
<td>E_NOT_OK</td>
<td></td>
</tr>
</tbody>
</table>

---

### SetECUGroupClassification

**Comments**
changes the ECU group classification status

**Parameters**

<table>
<thead>
<tr>
<th>Status</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Variation**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E_OK</td>
<td>Operation successful</td>
</tr>
<tr>
<td>E_NOT_OK</td>
<td></td>
</tr>
</tbody>
</table>

---

**8.7.2.4 ComM_UserRequest**

[SWS_ComM_01000] [ ]

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_UserRequest</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Comment</th>
</tr>
</thead>
</table>
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".

<table>
<thead>
<tr>
<th>IsService</th>
<th>true</th>
</tr>
</thead>
</table>

| Variation | -- |

<table>
<thead>
<tr>
<th>Possible Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 E_OK</td>
</tr>
<tr>
<td>1 E_NOT_OK</td>
</tr>
</tbody>
</table>
## Operations

### GetCurrentComMode

**Comments**
Returns the current Communication Manager Module mode for the SW-C

<table>
<thead>
<tr>
<th>Variation</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**
- ComMode

**Possible Errors**
- E_OK: Operation successful
- E_NOT_OK: --

### GetMaxComMode

**Comments**
Returns the current Communication Manager Module mode for the SW-C

<table>
<thead>
<tr>
<th>Variation</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**
- ComMode

**Possible Errors**
- E_OK: Operation successful
- E_NOT_OK: --

### GetRequestedComMode

**Comments**
Returns that last Communication Manager Module Mode requested by the SW-C

<table>
<thead>
<tr>
<th>Variation</th>
<th>Comment</th>
<th>Type</th>
<th>Variation</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>--</td>
<td>--</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**
- ComMode

**Possible Errors**
- E_OK: Operation successful
### Errors

| Errors | E_NOT_OK | -- |

### RequestComMode

| Comments | The SW-C requests that all communication channels it needs are in the provided Communication Manager Module mode |
| Variation | -- |
| Parameters | | |
| ComMode | Comment | -- |
| Type | ComM_ModeType |
| Variation | -- |
| Direction | IN |
| Possible Errors | | |
| E_OK | Operation successful |
| E_NOT_OK | -- |
| E_MODE_LIMITATION | ComMMMode cannot be granted because of ComMMMode inhibition |

#### 8.7.3 Mode-Switch-Interfaces

#### 8.7.3.1 ComM_CurrentMode

**[SWS_ComM_01001]**

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_CurrentMode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment</td>
<td>A SW-C that wants to get informed about its current Communication Manager Module Mode requires the ModeSwitchInterface ComM_CurrentMode.</td>
</tr>
<tr>
<td>IsService</td>
<td>true</td>
</tr>
<tr>
<td>Variation</td>
<td>--</td>
</tr>
<tr>
<td>ModeGroup</td>
<td>currentMode</td>
</tr>
</tbody>
</table>

#### 8.7.4 Implementation Data Types

#### 8.7.4.1 ComM_InhibitionStatusType

**[SWS_ComM_00669]**

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_InhibitionStatusType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Bitfield</td>
</tr>
<tr>
<td>Derived</td>
<td>uint8</td>
</tr>
</tbody>
</table>
### Elements

<table>
<thead>
<tr>
<th>Kind</th>
<th>Name</th>
<th>Mask</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>WakeupInhibitionActive</td>
<td>0x01</td>
<td>Bit 0 (LSB): Wake Up inhibition active</td>
</tr>
<tr>
<td></td>
<td>LimitedToNoCom</td>
<td>0x02</td>
<td>Bit 1: Limit to COMM_NO_COMMUNICATION mode</td>
</tr>
</tbody>
</table>

**Description**

Defines whether a mode inhibition is active or not.

Inhibition status of ComM.

e.g. status=00000011 -> Wake up inhibition and limitation to COMM_NO_COMMUNICATION mode active

### 8.7.4.2 ComM_ModeType

[SWS_ComM_00672]

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_ModeType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Type</td>
</tr>
<tr>
<td>Derived from</td>
<td>uint8</td>
</tr>
</tbody>
</table>

**Description**

Current mode of the Communication Manager (main state of the state machine).

<table>
<thead>
<tr>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMM_NO_COMMUNICATION</td>
<td>ComM state machine is in “No Communication” mode. Configured channel shall have no transmission or reception capability.</td>
</tr>
<tr>
<td>COMM_SILENT_COMMUNICATION</td>
<td>ComM state machine is in “Silent Communication” mode. Configured channel shall have only reception capability, no transmission capability.</td>
</tr>
<tr>
<td>COMM_FULL_COMMUNICATION</td>
<td>ComM state machine is in “Full Communication” mode. Configured channel shall have both transmission and reception capability.</td>
</tr>
</tbody>
</table>

**Variation**

--

### 8.7.4.3 ComM_UserHandleType

[SWS_ComM_00670]

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_UserHandleType</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Type</td>
</tr>
<tr>
<td>Derived from</td>
<td>uint8</td>
</tr>
</tbody>
</table>

**Description**

Handle to identify a user.

For each user, a unique value must be defined at system generation time.
Variation

Maximum number of users is 255. Legal user IDs are in the range 0 .. 254; user ID 255 is reserved and shall have the symbolic representation `COM_M_NOT_USED_USER_ID`.

### 8.7.4.4 ComM_UserHandleArrayType

[SWS_ComM_00906]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComM_UserHandleArrayType_{channel_name}</td>
<td></td>
</tr>
</tbody>
</table>

**Kind** | Structure

**Elements**

- `numberOfRequesters` | `uint8` | --
- `handleArray` | `ComM_UserHandleSubArrayType_{channel_name}` | --

**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)}`

### 8.7.4.5 ComM_UserHandleSubArrayType

[SWS_ComM_01005]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComM_UserHandleSubArrayType_{channel_name}</td>
<td></td>
</tr>
</tbody>
</table>

**Kind** | Array

**Element type** | `ComM_UserHandleType`

**Size**

`COUNT{ecuc(ComM/ComMConfigSet/ComMChannel/ComMUserPerChannel)}`

**Elements**

This element contains the user handles of the users which keep the channel requested (if any), starting in its first entries. The size of the array `MAX_CHANNEL_REQUESTERS` is the maximum of the number of users requesting a channel.

**Variation**

`channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel.SHORT-NAME)}`

### 8.7.5 Ports

#### 8.7.5.1 ComM_CL

[SWS_ComM_01006]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CL_{channel_name}</td>
<td></td>
</tr>
</tbody>
</table>

**Kind** | ProvidedPort

**Interface** | ComM_ChannelLimitation
### 8.7.5.2 ComM_CR

[SWS_ComM_01007]

<table>
<thead>
<tr>
<th>Name</th>
<th>CR_{channel_name}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>ProvidedPort</td>
</tr>
<tr>
<td>Description</td>
<td>--</td>
</tr>
<tr>
<td>Port Defined Argument Value(s)</td>
<td>--</td>
</tr>
<tr>
<td>Type</td>
<td>NetworkHandleType</td>
</tr>
<tr>
<td>Value</td>
<td>{ecuc(ComM/ComMConfigSet/ComMChannel/ComMChannelId.value)}</td>
</tr>
<tr>
<td>Variation</td>
<td>{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel)}</td>
</tr>
</tbody>
</table>

### 8.7.5.3 ComM_CW

[SWS_ComM_01008]

<table>
<thead>
<tr>
<th>Name</th>
<th>CW_{channel_name}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>ProvidedPort</td>
</tr>
<tr>
<td>Description</td>
<td>--</td>
</tr>
<tr>
<td>Port Defined Argument Value(s)</td>
<td>--</td>
</tr>
<tr>
<td>Type</td>
<td>NetworkHandleType</td>
</tr>
<tr>
<td>Value</td>
<td>{ecuc(ComM/ComMConfigSet/ComMChannel/ComMChannelId.value)}</td>
</tr>
<tr>
<td>Variation</td>
<td>{ecuc(ComM/ComMGeneral.ComMWakeupInhibitionEnabled)} == true channel_name = {ecuc(ComM/ComMConfigSet/ComMChannel)}</td>
</tr>
</tbody>
</table>

### 8.7.5.4 ComM_modeLimitation

[SWS_ComM_01009]

<table>
<thead>
<tr>
<th>Name</th>
<th>modeLimitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>ProvidedPort</td>
</tr>
<tr>
<td>Description</td>
<td>--</td>
</tr>
<tr>
<td>Variation</td>
<td>{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true</td>
</tr>
</tbody>
</table>

### 8.7.5.5 ComM_UP

[SWS_ComM_01010]

<table>
<thead>
<tr>
<th>Name</th>
<th>ComM_ECUModeLimitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>ProvidedPort</td>
</tr>
<tr>
<td>Description</td>
<td>--</td>
</tr>
<tr>
<td>Variation</td>
<td>{ecuc(ComM/ComMGeneral.ComMModeLimitationEnabled)} == true</td>
</tr>
</tbody>
</table>
8.7.5.6 ComM.UR
[SWS_ComM_01011]

<table>
<thead>
<tr>
<th>Name</th>
<th>UM_{user_name}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>ProvidedPort</td>
</tr>
<tr>
<td>Interface</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Variation</td>
<td>user_name = (ecuc(ComM/ComMConfigSet/ComMUUser.SHORT-NAME))</td>
</tr>
</tbody>
</table>

8.7.5.6.1 ComMMode
[SWS_ComM_01012]

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMMove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>ModeDeclarationGroup</td>
</tr>
<tr>
<td>Category</td>
<td>ALPHABETIC_ORDER</td>
</tr>
<tr>
<td>Initial mode</td>
<td>COMM_NO_COMMUNICATION</td>
</tr>
<tr>
<td>On transition value</td>
<td>--</td>
</tr>
<tr>
<td>Modes</td>
<td>--</td>
</tr>
<tr>
<td>Description</td>
<td>--</td>
</tr>
</tbody>
</table>

8.7.6 ModeDeclarationGroups
9 Sequence diagrams

9.1 Transmission and Reception start (CAN)

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

![Sequence diagram for starting transmission and reception on CAN](image)

Figure 9: Starting transmission and reception on CAN
9.2 Passive Wake-up (CAN)

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

![Diagram showing the behaviour after a wake-up](image)

**Figure 10: Reaction on a wake-up indicated by the ECU State Manager module**
9.3 Network shutdown (CAN)

Figure 11 shows the possibilities to shutdown the CAN network. It can be either initiated if the last user releases his `COMM_FULL_COMMUNICATION request` or `Comm LimitChannelToNoComMode(...)` (see `SWS_ComM_00163`) is called. The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.
Status: Proposed.
Description: Release of network for CAN. Similar sequence for LIN, FlexRay and Ethernet

Comments: - none -

Figure 11: Network shutdown (CAN)
9.4 Communication request

Figure 12 shows the possibilities to start \texttt{COMM\_FULL\_COMMUNICATION} on CAN. It can be either initiated if a user requests \texttt{COMM\_FULL\_COMMUNICATION} request or DCM indicates \texttt{ComM\_DCM\_ActiveDiagnostic} (see SWS\_ComM\_00873). The behaviour is equal for LIN, FlexRay and Ethernet just with different API names.

Figure 12: Request Communication
10 Configuration specification

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

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

Chapter 10.3 specifies published information of the Communication Manager Module.

10.1 How to read this chapter

For details refer to the chapter 10.1 “Introduction to configuration specification” in SWS_BSWGeneral.
10.2 Containers and configuration parameters

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

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

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

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

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

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

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

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

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

Rationale for SWS_ComM_00464: Easy and clear configuration.

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

**SWS Item**  
ECUC_ComM_00890 :

<table>
<thead>
<tr>
<th>Module Name</th>
<th>ComM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module Description</strong></td>
<td>Configuration of the ComM (Communications Manager) module.</td>
</tr>
<tr>
<td><strong>Supported Config Variants</strong></td>
<td>VARIANT-POST-BUILD, VARIANT-PRE-COMPILE</td>
</tr>
</tbody>
</table>

**Included Containers**

<table>
<thead>
<tr>
<th>Container Name</th>
<th>Multiplicity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComMConfigSet</td>
<td>1</td>
<td>This container contains the configuration parameters and sub containers of the AUTOSAR ComM module.</td>
</tr>
<tr>
<td>ComMGeneral</td>
<td>1</td>
<td>General configuration parameters of the Communication Manager.</td>
</tr>
</tbody>
</table>

Figure 13: Configuration ComM

10.2.2 ComMGeneral

**SWS Item**  
ECUC_ComM_00554 :

<table>
<thead>
<tr>
<th>Container Name</th>
<th>ComMGeneral</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>General configuration parameters of the Communication Manager.</td>
</tr>
</tbody>
</table>

**SWS Item**  
ECUC_ComM_00892 :

<p>| Name | ComM0PncVectorAvoidance |</p>
<table>
<thead>
<tr>
<th>Parent Container</th>
<th>ComMGeneral</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Description</strong></td>
<td>This parameter avoids sending of 0-PNC-Vectors in case ComMPncGatewayEnabled is enabled.</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>0..1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Post-Build Variant Value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time</td>
</tr>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
<tr>
<td><strong>Scope / Dependency</strong></td>
<td>scope: local</td>
</tr>
</tbody>
</table>

**SWS Item** ECUC_ComM_00555 :

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMDevErrorDetect</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMGeneral</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Switches the development error detection and notification on or off.</td>
</tr>
<tr>
<td></td>
<td>• true: detection and notification is enabled.</td>
</tr>
<tr>
<td></td>
<td>• false: detection and notification is disabled.</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Post-Build Variant Value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time</td>
</tr>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
<tr>
<td><strong>Scope / Dependency</strong></td>
<td>scope: local</td>
</tr>
</tbody>
</table>

**SWS Item** ECUC_ComM_00840 :

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMDirectUserMapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMGeneral</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>Restriction</strong></td>
<td>ComMUser, which are created due to this configuration parameter, shall not be used by SWCs (only available for BswM).</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>0..1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Post-Build Variant Value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Multiplicty Configuration Class</strong></td>
<td>Pre-compile time</td>
</tr>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time</td>
</tr>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
<tr>
<td><strong>Scope / Dependency</strong></td>
<td>scope: local</td>
</tr>
</tbody>
</table>
### Specification of Communication Manager

**AUTOSAR CP Release 4.3.1**

#### SWS Item: ECUC_ComM_00563

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMEcuGroupClassification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMGeneral</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucIntegerParamDef</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>0 .. 255</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>3</td>
</tr>
<tr>
<td><strong>Post-Build Variant Value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td><strong>Post-build time</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td><strong>Post-build time</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>Scope / Dependency</strong></td>
<td>scope: local</td>
</tr>
</tbody>
</table>

- dependency: Shall be stored non volatile (value must be kept during a reset) at least if Wake up inhibition is enabled/allowed. Can be changed during runtime with ComM_SetECUGroupClassification() thus the default values shall be set only once (first ECU initialization).

#### SWS Item: ECUC_ComM_00560

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMModeLimitationEnabled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMGeneral</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>true if mode limitation functionality shall be enabled. true: Enabled false: Disabled</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Post-Build Variant Value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td><strong>Post-build time</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td><strong>Post-build time</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>Scope / Dependency</strong></td>
<td>scope: local</td>
</tr>
</tbody>
</table>

#### SWS Item: ECUC_ComM_00887

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMPncGatewayEnabled</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMGeneral</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Enables or disables support of Partial Network Gateway. False: Partial Networking Gateway is disabled True: Partial Networking Gateway is enabled</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>0..1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Post-Build Variant Value</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td><strong>Post-build time</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>Value Configuration Class</strong></td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td><strong>Post-build time</strong></td>
<td>--</td>
</tr>
<tr>
<td><strong>Scope / Dependency</strong></td>
<td>scope: local</td>
</tr>
</tbody>
</table>

- AUTOSAR confidential
### SWS Item ECUC_ComM_00841:

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMPncPrepareSleepTimer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Time in seconds the PNC state machine shall wait in COMM_PNC_PREPARE_SLEEP.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>0..1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucFloatParamDef</td>
</tr>
<tr>
<td>Range</td>
<td>[0 .. 63]</td>
</tr>
<tr>
<td>Default value</td>
<td>--</td>
</tr>
<tr>
<td>Post-Build Variant</td>
<td>false</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Class</td>
<td>X</td>
</tr>
<tr>
<td>Link time</td>
<td>--</td>
</tr>
<tr>
<td>Post-build time</td>
<td>--</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Class</td>
<td>X</td>
</tr>
<tr>
<td>Link time</td>
<td>--</td>
</tr>
<tr>
<td>Post-build time</td>
<td>--</td>
</tr>
</tbody>
</table>

#### Scope / Dependency
- scope: local
- dependency: 
  - #CanNm: (CanNmPnResetTime + ComMPncPrepareSleepTimer) < CanNmTimeoutTime
  - #FrNm: (FrNmPnResetTime + ComMPncPrepareSleepTimer) < ((FrNmReadySleepCnt +1) * FrNmRepetitionCycle * "Duration of one FlexRay Cycle")
  - #UdpNm: (UdpNmPnResetTime + ComMPncPrepareSleepTimer) < UdpNmTimeoutTime

### SWS Item ECUC_ComM_00839:

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMPncSupport</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Enables or disables support of partial networking. False: Partial Networking is disabled True: Partial Networking is enabled</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Class</td>
<td>X</td>
</tr>
<tr>
<td>Link time</td>
<td>--</td>
</tr>
<tr>
<td>Post-build time</td>
<td>--</td>
</tr>
</tbody>
</table>

#### Scope / Dependency
- scope: local

### SWS Item ECUC_ComM_00558:

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMResetAfterForcingNoComm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>ComM shall perform a reset after entering &quot;No Communication&quot; mode because of an active mode limitation to &quot;No Communication&quot; mode. true: Enabled false: Disabled</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Class</td>
<td>X</td>
</tr>
<tr>
<td>Link time</td>
<td>--</td>
</tr>
<tr>
<td>Post-build time</td>
<td>--</td>
</tr>
</tbody>
</table>

#### Scope / Dependency
- scope: local
## Scope / Dependency

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00695 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMSynchronousWakeUp</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Wake up of one channel shall lead to a wake up of all channels if true. true: Enabled false: Disabled</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>true</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td></td>
<td>Link time --</td>
</tr>
<tr>
<td></td>
<td>Post-build time --</td>
</tr>
</tbody>
</table>

## Scope / Dependency

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00557 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMTMinFullComModeDuration</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Minimum time duration in seconds, spent in the COMM_FULL_COMMUNICATION sub-state COMM_FULL_COM_NETWORK_REQUESTED.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucFloatParamDef</td>
</tr>
<tr>
<td>Range</td>
<td>[0.001 .. 65]</td>
</tr>
<tr>
<td>Default value</td>
<td>5</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td></td>
<td>Link time --</td>
</tr>
<tr>
<td></td>
<td>Post-build time --</td>
</tr>
</tbody>
</table>

## Scope / Dependency

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00622 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMVersionInfoApi</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>Switches the possibility to read the published information with the service ComM_GetPublishedInformation(). true: Enabled false: Disabled</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time X All Variants</td>
</tr>
<tr>
<td></td>
<td>Link time --</td>
</tr>
<tr>
<td></td>
<td>Post-build time --</td>
</tr>
</tbody>
</table>

## Scope / Dependency

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00559 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMWakeupInhibitionEnabled</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMGeneral</td>
</tr>
<tr>
<td>Description</td>
<td>true if wake up inhibition functionality enabled. true: Enabled false: Disabled</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
</tbody>
</table>
### Type
- **EcucBooleanParamDef**

### Default value
- **false**

### Post-Build Variant Value
- **false**

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
<th>Pre-compile time</th>
<th>Link time</th>
<th>Post-build time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td><strong>--</strong></td>
<td><strong>--</strong></td>
<td></td>
</tr>
</tbody>
</table>

| **Post-build time** | **--** |

| Scope / Dependency  | **scope: local** |

### SWS Item
- **ECUC_ComM_00783**

#### Name
- **ComMGlobalNvMBlockDescriptor**

#### Parent Container
- **ComMGeneral**

#### Description
Reference to NVRAM block containing the none volatile data. If this parameter is not configured it means that no NVRam is used at all.

#### Multiplicity
- **0..1**

#### Type
- Symbolic name reference to NvMBlockDescriptor

#### Post-Build Variant Multiplicity
- **false**

#### Post-Build Variant Value
- **false**

<table>
<thead>
<tr>
<th>Multiplicity Configuration Class</th>
<th>Pre-compile time</th>
<th>Link time</th>
<th>Post-build time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td><strong>--</strong></td>
<td><strong>--</strong></td>
<td></td>
</tr>
</tbody>
</table>

| **Post-build time** | **--** |

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
<th>Pre-compile time</th>
<th>Link time</th>
<th>Post-build time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
<td><strong>--</strong></td>
<td><strong>--</strong></td>
<td></td>
</tr>
</tbody>
</table>

| **Post-build time** | **--** |

| Scope / Dependency  | **scope: ECU** dependency: Derived from NvM configuration |

No Included Containers
10.2.3 ComMConfigSet

**SWS Item** ECUC_ComM_00879 :

**Container Name** ComMConfigSet

**Description** This container contains the configuration parameters and sub containers of the AUTOSAR ComM module.

**Configuration Parameters**

| SWS Item | ECUC_ComM_00878 :
| **Name** | ComMConfigSet |
| **Description** | Defines whether in this configuration set the partial networking is enabled. |
### 10.2.4 ComMUser

**Container Name** ComMUser

**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.

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00654</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMUserIdentifier</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMUser</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>An identifier that is needed to refer to a user in the system which is designated to request Communication Modes. ImplementationType: ComM_UserHandleType</td>
</tr>
</tbody>
</table>

**Multiplicity** 1

**Type** EcuIntegerParamDef (Symbolic Name generated for this parameter)

**Range** 0 .. 254

**Default value** --

**Post-Build Variant Value** false

**Value Configuration Class**

<table>
<thead>
<tr>
<th>Pre-compile time</th>
<th>X</th>
<th>All Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link time</td>
<td>--</td>
<td></td>
</tr>
<tr>
<td>Post-build time</td>
<td>X</td>
<td>VARIANT-POST-BUILD</td>
</tr>
</tbody>
</table>

**Scope / Dependency**
scope: local
dependency: EcuMUser: The concept of users is very similar to the concept of requestors in the ECU State Manager specification. These two parameters shall be harmonized during the configuration process.
SWS Item | ECUC_ComM_00786 :
---|---
Name | ComMUserEcucPartitionRef
Parent Container | ComMUser
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 ]
Post-Build Variant Multiplicity | false
Post-Build Variant Value | false
Multiplicity Configuration Class | Pre-compile time X All Variants
| Link time --
| Post-build time --
Value Configuration Class | Pre-compile time X All Variants
| Link time --
| Post-build time --
Scope / Dependency | scope: ECU

No Included Containers

Figure 15: Configuration ComMUser

10.2.5 ComMChannel

SWS Item | ECUC_ComM_00565 :
---|---

### Container Name
ComMChannel

### Description
This container contains the configuration (parameters) of the bus channel(s). The channel parameters shall be harmonized within the whole communication stack.

### Configuration Parameters

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00567</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMBusType</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMChannel</td>
</tr>
<tr>
<td>Description</td>
<td>Identifies the bus type of the channel.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucEnumerationParamDef</td>
</tr>
</tbody>
</table>
| Range          | COMM_BUS_TYPE_CAN --
|                | COMM_BUS_TYPE_CDD --
|                | COMM_BUS_TYPE_ETH --
|                | COMM_BUS_TYPE_FR  --
|                | COMM_BUS_TYPE_INTERNAL --
|                | COMM_BUS_TYPE_LIN  -- |
| Post-Build Variant Value | false |
| Value Configuration | Pre-compile time | X All Variants |
| Class           | Link time | -- |
| Post-build time | -- |

| Scope / Dependency | scope: ECU |

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00888</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMCDDBusPrefix</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMChannel</td>
</tr>
<tr>
<td>Description</td>
<td>Prefix to be used for API calls to CDD.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>0..1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucStringParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>--</td>
</tr>
<tr>
<td>maxLength</td>
<td>--</td>
</tr>
<tr>
<td>minLength</td>
<td>--</td>
</tr>
<tr>
<td>regularExpression</td>
<td>--</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
<td>--</td>
</tr>
<tr>
<td>Post-build time</td>
<td>--</td>
</tr>
<tr>
<td>Post-build time</td>
<td>--</td>
</tr>
</tbody>
</table>

| Scope / Dependency | scope: local
dependency: Only applicable if ComMBusType equals COMM_BUS_TYPE_CDD. |

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00635</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMChannelId</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMChannel</td>
</tr>
<tr>
<td>Description</td>
<td>Channel identification number of the corresponding channel.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucIntegerParamDef (Symbolic Name generated for this parameter)</td>
</tr>
<tr>
<td>Range</td>
<td>0 .. 255</td>
</tr>
</tbody>
</table>
### SWS Item: ECUC_ComM_00787:
- **Name**: ComMFullCommRequestNotificationEnabled
- **Parent Container**: ComMChannel
- **Description**: Defines if the optional SenderReceiver Port of Interface ComM_CurrentChannelRequest will be provided for this channel. True means enabled, False means disabled
- **Multiplicity**: 1
- **Type**: EcucBooleanParamDef
- **Default value**: false

### SWS Item: ECUC_ComM_00556:
- **Name**: ComMMainFunctionPeriod
- **Parent Container**: ComMChannel
- **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
- **Default value**: 0.02

### SWS Item: ECUC_ComM_00571:
- **Name**: ComMNoCom
- **Parent Container**: ComMChannel
- **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.
- **Multiplicity**: 1
Specification of Communication Manager
AUTOSAR CP Release 4.3.1

<table>
<thead>
<tr>
<th>Type</th>
<th>EcucBooleanParamDef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
<th>Pre-compile time X All Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope / Dependency</th>
<th>scope: local</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dependency: ComMModeLimitationEnabled</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00569 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMNoWakeup</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMChannel</td>
</tr>
<tr>
<td>Description</td>
<td>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().</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>false</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
<th>Pre-compile time X All Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope / Dependency</th>
<th>scope: local</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>dependency: Shall be stored none volatile (value must be kept during a reset).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00789 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMNoWakeUpInhibitionNvmStorage</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMChannel</td>
</tr>
<tr>
<td>Description</td>
<td>If this parameter is set to &quot;true&quot;, the NoWakeUp inhibition state of the channel shall be stored (in some implementation specific way) in the block pointed to by ComMGlobalNvmBlockDescriptor.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucBooleanParamDef</td>
</tr>
<tr>
<td>Default value</td>
<td>--</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
<th>Pre-compile time X All Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope / Dependency</th>
<th>scope: local</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00842 :</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMPncGatewayType</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMChannel</td>
</tr>
<tr>
<td>Description</td>
<td>Identifies the Partial Network Gateway behaviour of a ComMChannel.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>0..1</td>
</tr>
<tr>
<td>Type</td>
<td>EcucEnumerationParamDef</td>
</tr>
<tr>
<td>Range</td>
<td>COMM_GATEWAY_TYPE_ACTIVE --</td>
</tr>
<tr>
<td></td>
<td>COMM_GATEWAY_TYPE_PASSIVE --</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Container Name</td>
<td>Multiplicity</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>ComMNetworkManagement</td>
<td>1</td>
</tr>
<tr>
<td>ComMUserPerChannel</td>
<td>0..255</td>
</tr>
</tbody>
</table>

[SWS_ComM_00690] Configuration parameter ComMNoCom (see ECUC_ComM_00571) need not to be evaluated in case ComMModeLimitationEnabled = FALSE = Disabled (see ECUC_ComM_00560) thus it can be removed in that case to reduce/optimize the configuration.] ()
Specification of Communication Manager
AUTOSAR CP Release 4.3.1

Figure 16: Configuration ComMChannel
## 10.2.6 ComMNetworkManagement

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00607 :</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Container Name</strong></td>
<td>ComMNetworkManagement</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>This container contains the configuration parameters of the networkmanagement.</td>
</tr>
</tbody>
</table>

### Configuration Parameters

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00606 :</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>ComMNmLightTimeout</td>
</tr>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMNetworkManagement</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>0..1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucFloatParamDef</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>[0 .. 255]</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>10</td>
</tr>
<tr>
<td><strong>Post-Build Variant</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Multiplicity Value</strong></td>
<td>false</td>
</tr>
</tbody>
</table>

### Multiplicity Configuration Class

<table>
<thead>
<tr>
<th>Pre-compile time</th>
<th>Link time</th>
<th>Post-build time</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

### Value Configuration Class

<table>
<thead>
<tr>
<th>Pre-compile time</th>
<th>Link time</th>
<th>Post-build time</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

### Scope / Dependency

scope: local
dependency: Only used if ComMNmVariant is configured as ComMLight

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00568 :</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>ComMNmVariant</td>
</tr>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMNetworkManagement</td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td>Defines the functionality of the networkmanagement. Shall be harmonized with NM configuration.</td>
</tr>
<tr>
<td><strong>Multiplicity</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>EcucEnumerationParamDef</td>
</tr>
<tr>
<td><strong>Range</strong></td>
<td>FULL</td>
</tr>
<tr>
<td><strong>Default value</strong></td>
<td>FULL</td>
</tr>
<tr>
<td><strong>Post-Build Variant</strong></td>
<td>false</td>
</tr>
<tr>
<td><strong>Configuration Class</strong></td>
<td>Pre-compile time</td>
</tr>
<tr>
<td></td>
<td>Link time</td>
</tr>
<tr>
<td></td>
<td>Post-build time</td>
</tr>
</tbody>
</table>

### Value Configuration Class

<table>
<thead>
<tr>
<th>Pre-compile time</th>
<th>Link time</th>
<th>Post-build time</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

### Scope / Dependency

scope: local
dependency: ComMNmVariant shall be NONE if ComMBusType = COMM_BUS_TYPE_INTERNAL.

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00886 :</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>ComMPncNmRequest</td>
</tr>
<tr>
<td><strong>Parent Container</strong></td>
<td>ComMNetworkManagement</td>
</tr>
</tbody>
</table>
**Description**

If this parameter equals true then every time a FULL Communication is requested due to a change in the PNC state machine to COMM_PNC_REQUESTED Nm shall be called using the API Nm_NetworkRequest.

**Multiplicity**

1

**Type**

EcucBooleanParamDef

**Default value**

false

**Post-Build Variant Value**

false

**Value Configuration Class**

Pre-compile time: X All Variants

Link time: --

Post-build time: --

**Scope / Dependency**

scope: local

dependency: It shall only be possible to set ComMPncNmRequest to TRUE, if ComMNmVariant is FULL.

**No Included Containers**

---

**ComMChannel**:
EcucParamConfContainerDef

lowerMultiplicity = 1
upperMultiplicity = 256

**ComMNetworkManagement**:
EcucParamConfContainerDef

upperMultiplicity = 1
lowerMultiplicity = 1

**ComMNmVariant**:
EcucEnumerationParamDef

defaultValue = FULL

**ComMNmLightTimeout**:
EcucFloatParamDef

max = 255
min = 0
defaultValue = 10
upperMultiplicity = 1
lowerMultiplicity = 0

**ComMPncNmRequest**:
EcucBooleanParamDef

defaultValue = false

**NONE**:
EcucEnumerationLiteralDef

**LIGHT**:
EcucEnumerationLiteralDef

**PASSIVE**:
EcucEnumerationLiteralDef

**FULL**:
EcucEnumerationLiteralDef

---

Figure 17: Configuration ComMNetworkManagement
10.2.7 ComMUserPerChannel

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00657</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container Name</td>
<td>ComMUserPerChannel</td>
</tr>
<tr>
<td>Description</td>
<td>This container contains a list of identifiers that are needed to refer to a user in the system which is linked to a channel.</td>
</tr>
</tbody>
</table>

**Configuration Parameters**

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00658</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMUserChannel</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMUserPerChannel</td>
</tr>
<tr>
<td>Description</td>
<td>Reference to the ComMUser that corresponds to this channel user. ImplementationType: COMM_UserHandleType</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>Reference to [ ComMUser ]</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time: X</td>
</tr>
<tr>
<td></td>
<td>Link time: --</td>
</tr>
<tr>
<td></td>
<td>Post-build time: --</td>
</tr>
<tr>
<td>Scope / Dependency</td>
<td>scope: local</td>
</tr>
</tbody>
</table>

No Included Containers

10.2.8 ComMPnc

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00843</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container Name</td>
<td>ComMPnc</td>
</tr>
<tr>
<td>Description</td>
<td>This container contains the configuration of the partial network cluster (PNC).</td>
</tr>
</tbody>
</table>

**Configuration Parameters**

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00874</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMPncId</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMPnc</td>
</tr>
<tr>
<td>Description</td>
<td>Partial network cluster identification number.</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1</td>
</tr>
<tr>
<td>Type</td>
<td>EcuIntegerParamDef (Symbolic Name generated for this parameter)</td>
</tr>
<tr>
<td>Range</td>
<td>8 .. 63</td>
</tr>
<tr>
<td>Default value</td>
<td>--</td>
</tr>
<tr>
<td>Post-Build Variant Value</td>
<td>false</td>
</tr>
<tr>
<td>Value Configuration Class</td>
<td>Pre-compile time: X</td>
</tr>
<tr>
<td></td>
<td>Link time: --</td>
</tr>
<tr>
<td></td>
<td>Post-build time: --</td>
</tr>
<tr>
<td>Scope / Dependency</td>
<td>scope: ECU</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SWS Item</th>
<th>ECUC_ComM_00880</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ComMChannelPerPnc</td>
</tr>
<tr>
<td>Parent Container</td>
<td>ComMPnc</td>
</tr>
<tr>
<td>Description</td>
<td>Reference to the ComMChannel that is required for this PNC. ImplementationType: NetworkHandleType</td>
</tr>
<tr>
<td>Multiplicity</td>
<td>1..256</td>
</tr>
</tbody>
</table>
### Specification of Communication Manager

**AUTOSAR CP Release 4.3.1**

#### Type
Reference to [ComMChannel]

#### Post-Build Variant

<table>
<thead>
<tr>
<th>Multiplicity</th>
<th>true</th>
</tr>
</thead>
</table>

#### Post-Build Variant Value

| true |

<table>
<thead>
<tr>
<th>Multiplicity Configuration Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
</tr>
<tr>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
</tr>
<tr>
<td>Post-build time</td>
</tr>
</tbody>
</table>

#### Scope / Dependency

- scope: local

### SWS Item

**ECUC_ComM_00891**

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMPncEthIfSwitchPortGroupRef</th>
</tr>
</thead>
</table>

| Parent Container            | ComMPnc                         |

| Description                 | Reference to the PortGroups that correspond to this PNC. Note: This is only for documentation. |

| Multiplicity                | 0..255                         |

| Type                        | Symbolic name reference to [EthIfSwitchPortGroup] |

| Post-Build Variant Multiplicity | true |

| Post-Build Variant Value true |

<table>
<thead>
<tr>
<th>Multiplicity Configuration Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
</tr>
<tr>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
</tr>
<tr>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope / Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope: ECU</td>
</tr>
</tbody>
</table>

### SWS Item

**ECUC_ComM_00876**

<table>
<thead>
<tr>
<th>Name</th>
<th>ComMUserPerPnc</th>
</tr>
</thead>
</table>

| Parent Container            | ComMPnc                         |

| Description                 | Reference to the ComMUsers that correspond to this PNC. ImplementationType: COMM_UserHandleType |

| Multiplicity                | 0..255                         |

| Type                        | Reference to [ComMUser] |

| Post-Build Variant Multiplicity | true |

| Post-Build Variant Value true |

<table>
<thead>
<tr>
<th>Multiplicity Configuration Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
</tr>
<tr>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Configuration Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-compile time</td>
</tr>
<tr>
<td>Link time</td>
</tr>
<tr>
<td>Post-build time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scope / Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>scope: local</td>
</tr>
</tbody>
</table>

### Included Containers

<table>
<thead>
<tr>
<th>Container Name</th>
<th>Multiplicity</th>
<th>Scope / Dependency</th>
</tr>
</thead>
<tbody>
<tr>
<td>ComMPncComSignal</td>
<td>0..*</td>
<td>Represents the PncComSignals which are used to communicate the EIRA and ERA status of this PNC.</td>
</tr>
</tbody>
</table>
### 10.2.9 ComMPncComSignal

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

<table>
<thead>
<tr>
<th>Configuration Parameters</th>
</tr>
</thead>
</table>
| **SWS Item** | ECUC_ComM_00885 :
---|---
**Name** | ComMPncComSignalDirection
**Parent Container** | ComMPncComSignal
**Description** | Indicates the communication direction of this PncComSignal.
**Multiplicity** | 1
**Type** | EcucEnumerationParamDef
**Range** | RX -- TX
**Value** | false

<table>
<thead>
<tr>
<th>Configuration Class</th>
</tr>
</thead>
</table>
| **Pre-compile time** | X All Variants
| **Link time** | 
| **Post-build time** | 

**Scope / Dependency** | scope: local

| SWS Item | ECUC_ComM_00883 :
---|---
**Name** | ComMPncComSignalKind
**Parent Container** | ComMPncComSignal
**Description** | Indicates whether this PncComSignal represents EIRA or ERA PNC information. This parameter ComMPncComSignalKind is optional and shall be ignored when ComMPncComSignalDirection equals TX.
**Multiplicity** | 0..1
**Type** | EcucEnumerationParamDef
**Range** | EIRA -- ERA
**Value** | false

<table>
<thead>
<tr>
<th>Configuration Class</th>
</tr>
</thead>
</table>
| **Pre-compile time** | X All Variants
| **Link time** | 
| **Post-build time** | 

**Scope / Dependency** | dependency: This parameter ComMPncComSignalKind shall be ignored when ComMPncComSignalDirection equals TX.

| SWS Item | ECUC_ComM_00884 :
---|---
**Name** | ComMPncComSignalChannelRef
**Parent Container** | ComMPncComSignal
**Description** | Reference to the ComMChannel which is used to determine whether this PncComSignal shall participate in the active or passive role (via the parameter ComMPncGatewayType of the ComMChannel).
**Multiplicity** | 0..1
**Type** | Reference to [ ComMChannel ]
### Specification of Communication Manager

**AUTOSAR CP Release 4.3.1**

**Document ID:** AUTOSAR_SWS_COMManager

**Post-Build Variant**
- **Multiplicity:** false
- **Value:** false

**Multiplicity Configuration Class**
- **Pre-compile time:** X All Variants
- **Link time:** --
- **Post-build time:** --

**Value Configuration Class**
- **Pre-compile time:** X All Variants
- **Link time:** --
- **Post-build time:** --

**Scope / Dependency**
- **Scope:** local
- **Dependency:** ComMPncGatewayEnabled

---

**SWS Item**
- **ECUC_ComM_00882:**
- **Name:** ComMPncComSignalRef
- **Parent Container:** ComMPncComSignal
- **Description:** Reference to the ComSignal which is used to transport the partial network channel request information.
- **Multiplicity:** 1
- **Type:** Symbolic name reference to [ComSignal]
- **Post-Build Variant Value:** false

**Value Configuration Class**
- **Pre-compile time:** X All Variants
- **Link time:** --
- **Post-build time:** --

**Scope / Dependency**
- **Scope:** ECU

---

**No Included Containers**

---

### 10.3 Published information

[SWS_ComM_00418] The version information in the module header and source files shall be validated and consistent (e.g. by comparing the version information in the module header and source files with a pre-processor macro).] (SRS_BSW_00004)
11 Not applicable requirements

[SWS_ComM_00499] These requirements are not applicable to this specification.

(SRS_BSW_00005, SRS_BSW_00009, SRS_BSW_00010, SRS_BSW_00161, SRS_BSW_00162, SRS_BSW_00164, SRS_BSW_00168, SRS_BSW_00170, SRS_BSW_00314, SRS_BSW_00325, SRS_BSW_00341, SRS_BSW_00343, SRS_BSW_00344, SRS_BSW_00353, SRS_BSW_00361, SRS_BSW_00375, SRS_BSW_00378, SRS_BSW_00398, SRS_BSW_00404, SRS_BSW_00405, SRS_BSW_00413, SRS_BSW_00416, SRS_BSW_00417, SRS_BSW_00422, SRS_BSW_00423, SRS_BSW_00424, SRS_BSW_00425, SRS_BSW_00426, SRS_BSW_00427, SRS_BSW_00428, SRS_BSW_00429, SRS_BSW_00432, SRS_BSW_00433, SRS_BSW_00437, SRS_BSW_00438, SRS_BSW_00439)