Document Title	SWS_COMManager: Complete Change Documentation 4.3.0 - 4.3.1
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	695

Document Status	Final
Part of AUTOSAR Standard	Classic Platform
Part of Standard Release	4.3.1

Table of Contents

1 SWS_COMManager

1.1 Specification Item SWS_ComM_00110

Trace References:

SRS_ModeMgm_09081

Content:

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

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #74626: [ComM] Specification of the mode limitation seem to be incomplete

  Problem description:

  Mode limitation is devided in 2 parts:
  (1)[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)
  -> This cause an state change from "network_requested" to "network_release"

(2)[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.?()
-> This means only channel state machines in state "COMM_FULL_COM_NETWORK_REQUESTED" are limited to "COMM_NO_COMMUNICATION". But this should also be done for channels in state "COMM_NO_COMMUNICATION". Otherwise a channel in "COMM_NO_COMMUNICATION" is not limited and if a "COMM_FULL_COM" is requested, the channel permit communication and violate the limitation.

This points should be clarified:
- according to (1): the change from "network_requested" to "network_release" should not be performed, because it is not necessary. Only limitation for this channel should take place. Additonally should a mode request to "COMM_FULL_COMMUNICATION" return E_OK in mode limitation.
- according to (2): if a channel is in "COMM_NO_COMMUNICATION" the mode limitation should also be activated for this channel.

Agreed solution:

- add note below [SWS_ComM_00841]:
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.

- add note below [SWS_ComM_00842]:
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.

add note to the description of [SWS_ComM_00110]:
Note: The communication request could also be released due to a ComM communication inhibition
–Last change on issue 74626 comment 9–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.2 Specification Item SWS_ComM_00124

Trace References:

SRS_ModeMgm_09157

Content:

Service name:	ComM_LimitECUToNoComModeComM_LimitECUToNoComMode	
Syntax:	Std_ReturnType ComM_LimitECUToNoComMode(boolean Status)	
Service ID[hex]:	0x0c	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	StatusComM_LimitECUToNoCom Mode.Status	FALSE: Limit ECU to COMM_NO_COMMUNICATION disabled TRUE: Limit ECU to COMM_NO_COMMUNICATION enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed inhibition status for the ECU E_NOT_OK: Changed Change of inhibition status for the ECU failed, e.g. ComMEcuGroup Classification 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 Com M_LimitChannelToNoComMode, same functionality but for a specific channels)	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76916: [ComM] Clarification of ComM_SetECUGroupClassification() / ComMEcuGroupClassification

  Problem description:

  We see some ambiguity on handling of communication inhibition, esp. ComM_SetECUGroupClassification() / ComMEcuGroupClassification.

  [Assumed usecase (example)]
  * Configuration
  * ComMEcuGroupClassification0x00
  * ComMChannel1
  * ComMNoCom:false
  * ComMNoWakeup:false

* Initial state
* ComMChannel1:COMM_NO_COMMUNICATION

* Steps (let's assume following APIs will be triggered during runtime by Diagnostic requests etc.)
Step-1: invoke ComM_PreventWakeUp(ComMChannel1, true)
Step-2: invoke ComM_LimitChannelToNoComMode(ComMChannel1, true)
Step-3: invoke ComM_SetECUGroupClassification(0x03)

In the case above, I have several questions.

Q1)
At invocation of ComM_PreventWakeUp (Step-1), wakeup inhibition will stay inactive, because of ComMEcuGroupClassification value (0x00).
In this case, which return value of ComM_PreventWakeUp is expected, E_OK or E_NOT_OK?

Q2)
At invocation of ComM_LimitChannelToNoComMode (Step-2), limit to NoCom will stay inactive, because of ComMEcuGroupClassification value (0x00).
In this case, which return value of ComM_LimitChannelToNoComMode is expected, E_OK or E_NOT_OK?

Q3)
As described in ECUC_ComM_00563, ComMEcuGroupClassification defines whether a mode inhibition (both of communication limitation features; wakeup inhibition + limit to NoCom) affects the ECU or not. And for this switching, ComM_SetECUGroupClassification() API exists.
Here, at Step-3, what will happen? Is each wakeup inhibition and limit to NoCom activated or not ?

Q4)
Even if not activated at Q3, ComM_SetECUGroupClassification() returns E_OK. Am I right?

Note that, our interpretation is:
* ComMNoCom: "request bit" of mode inhibition (limit to NoCom), can be controlled by ComM_LimitChannelToNoComMode() and ComM_LimitECUToNoComMode(), regardless of ComMEcuGroupClassification value.
* ComMNoWakeup: "request bit" of mode inhibition (wakeup inhibition), can be controlled by ComM_PreventWakeUp(), regardless of ComMEcuGroupClassification value.
* ComMEcuGroupClassification: "mask bits" of mode inhibition behavior, can be

controlled by ComM_SetECUGroupClassification(), regardless of ComMNoCom and ComMNoWakeup values

Probably the notes above can be added to the SWS (in sec. 7.3.1 Communication inhibition) to have more clarity.

Agreed solution:

SWS ComM
==========

~[SWS_ComM_00163]
Change description of Return value E_NOT_OK: Changed of inhibition status for the channel failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

~[SWS_ComM_00124]
Change description of Return value E_NOT_OK: Changed of inhibition status for the ECU failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

~[SWS_ComM_00156]
Change description of Return value E_NOT_OK: Changed of wake up status for the channel failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

+add notes in sec. 7.3.1 Communication inhibition
[...]
Note: 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
[...]
–Last change on issue 76916 comment 5–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.3 Specification Item SWS_ComM_00156

Trace References:

SRS_ModeMgm_09157

Content:

Service name:	ComM_PreventWakeUpComM_PreventWakeUp	
Syntax:	Std_ReturnType ComM_PreventWakeUp(NetworkHandleType Channel, boolean Status)	
Service ID[hex]:	0x09	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ChannelComM_PreventWake Up.Channel	See NetworkHandleType
	StatusComM_PreventWakeUp.Status	FALSE: Wake up inhibition is switched off TRUE: Wake up inhibition is switched on
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed wake up status for the channel E_NOT_OK: Changed Change of wake up status for the channel failed, e.g. ComMEcuGroup Classification disables the functionality (see ECUC_ComM_00563)
Description:	Changes the inhibition status COMM_NO_WAKEUP for the corresponding channel.	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76916: [ComM] Clarification of ComM_SetECUGroupClassification() / ComMEcuGroupClassification

  Problem description:

  We see some ambiguity on handling of communication inhibition, esp. ComM_SetECUGroupClassification() / ComMEcuGroupClassification.

  [Assumed usecase (example)]
  * Configuration
  * ComMEcuGroupClassification0x00

* ComMChannel1
* ComMNoCom:false
* ComMNoWakeup:false

* Initial state
* ComMChannel1:COMM_NO_COMMUNICATION

* Steps (let's assume following APIs will be triggered during runtime by Diagnostic requests etc.)
Step-1: invoke ComM_PreventWakeUp(ComMChannel1, true)
Step-2: invoke ComM_LimitChannelToNoComMode(ComMChannel1, true)
Step-3: invoke ComM_SetECUGroupClassification(0x03)

In the case above, I have several questions.

Q1)
At invocation of ComM_PreventWakeUp (Step-1), wakeup inhibition will stay inactive, because of ComMEcuGroupClassification value (0x00).
In this case, which return value of ComM_PreventWakeUp is expected, E_OK or E_NOT_OK?

Q2)
At invocation of ComM_LimitChannelToNoComMode (Step-2), limit to NoCom will stay inactive, because of ComMEcuGroupClassification value (0x00).
In this case, which return value of ComM_LimitChannelToNoComMode is expected, E_OK or E_NOT_OK?

Q3)
As described in ECUC_ComM_00563, ComMEcuGroupClassification defines whether a mode inhibition (both of communication limitation features; wakeup inhibition + limit to NoCom) affects the ECU or not. And for this switching, ComM_SetECUGroupClassification() API exists.
Here, at Step-3, what will happen? Is each wakeup inhibition and limit to NoCom activated or not ?

Q4)
Even if not activated at Q3, ComM_SetECUGroupClassification() returns E_OK. Am I right?

Note that, our interpretation is:
* ComMNoCom: "request bit" of mode inhibition (limit to NoCom), can be controlled by ComM_LimitChannelToNoComMode() and ComM_LimitECUToNoComMode(), regardless of ComMEcuGroupClassification value.

* ComMNoWakeup: "request bit" of mode inhibition (wakeup inhibition), can be controlled by ComM_PreventWakeUp(), regardless of ComMEcuGroupClassification value.
* ComMEcuGroupClassification: "mask bits" of mode inhibition behavior, can be controlled by ComM_SetECUGroupClassification(), regardless of ComMNoCom and ComMNoWakeup values

Probably the notes above can be added to the SWS (in sec. 7.3.1 Communication inhibition) to have more clarity.

Agreed solution:

SWS ComM
==========

~[SWS_ComM_00163]
Change description of Return value E_NOT_OK: Changed of inhibition status for the channel failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

~[SWS_ComM_00124]
Change description of Return value E_NOT_OK: Changed of inhibition status for the ECU failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

~[SWS_ComM_00156]
Change description of Return value E_NOT_OK: Changed of wake up status for the channel failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

+add notes in sec. 7.3.1 Communication inhibition
[...]
Note: 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

[...]
–Last change on issue 76916 comment 5–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.4 Specification Item SWS_ComM_00163

Trace References:

SRS_ModeMgm_09157

Content:

Service name:	ComM_LimitChannelToNoComModeComM_LimitChannelToNoComMode	
Syntax:	Std_ReturnType ComM_LimitChannelToNoComMode(NetworkHandleType Channel, boolean Status)	
Service ID[hex]:	0x0b	
Sync/Async:	Synchronous	
Reentrancy:	Non Reentrant	
Parameters (in):	ChannelComM_LimitChannelToNoCom Mode.Channel	See NetworkHandleType
	StatusComM_LimitChannelToNoCom Mode.Status	FALSE: Limit channel to COMM_NO_COMMUNICATION disabled TRUE: Limit channel to COMM_NO_COMMUNICATION enabled
Parameters (inout):	None	
Parameters (out):	None	
Return value:	Std_ReturnType	E_OK: Successfully changed inhibition status for the channel E_NOT_OK: Changed Change of inhibition status for the channel failed, e.g. ComMEcuGroup Classification 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 Com M_LimitECUToNoComMode, same functionality but for all channels)	

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #76916: [ComM] Clarification of ComM_SetECUGroupClassification() / ComMEcuGroupClassification

  Problem description:

We see some ambiguity on handling of communication inhibition, esp. ComM_SetECUGroupClassification() / ComMEcuGroupClassification.

[Assumed usecase (example)]
* Configuration
* ComMEcuGroupClassification0x00
* ComMChannel1
* ComMNoCom:false
* ComMNoWakeup:false

* Initial state
* ComMChannel1:COMM_NO_COMMUNICATION

* Steps (let's assume following APIs will be triggered during runtime by Diagnostic requests etc.)
Step-1: invoke ComM_PreventWakeUp(ComMChannel1, true)
Step-2: invoke ComM_LimitChannelToNoComMode(ComMChannel1, true)
Step-3: invoke ComM_SetECUGroupClassification(0x03)

In the case above, I have several questions.

Q1)
At invocation of ComM_PreventWakeUp (Step-1), wakeup inhibition will stay inactive, because of ComMEcuGroupClassification value (0x00).
In this case, which return value of ComM_PreventWakeUp is expected, E_OK or E_NOT_OK?

Q2)
At invocation of ComM_LimitChannelToNoComMode (Step-2), limit to NoCom will stay inactive, because of ComMEcuGroupClassification value (0x00).
In this case, which return value of ComM_LimitChannelToNoComMode is expected, E_OK or E_NOT_OK?

Q3)
As described in ECUC_ComM_00563, ComMEcuGroupClassification defines whether a mode inhibition (both of communication limitation features; wakeup inhibition + limit to NoCom) affects the ECU or not. And for this switching, ComM_SetECUGroupClassification() API exists.
Here, at Step-3, what will happen? Is each wakeup inhibition and limit to NoCom activated or not ?

Q4)
Even if not activated at Q3, ComM_SetECUGroupClassification() returns E_OK. Am

I right?

Note that, our interpretation is:
* ComMNoCom: "request bit" of mode inhibition (limit to NoCom), can be controlled by ComM_LimitChannelToNoComMode() and ComM_LimitECUToNoComMode(), regardless of ComMEcuGroupClassification value.
* ComMNoWakeup: "request bit" of mode inhibition (wakeup inhibition), can be controlled by ComM_PreventWakeUp(), regardless of ComMEcuGroupClassification value.
* ComMEcuGroupClassification: "mask bits" of mode inhibition behavior, can be controlled by ComM_SetECUGroupClassification(), regardless of ComMNoCom and ComMNoWakeup values

Probably the notes above can be added to the SWS (in sec. 7.3.1 Communication inhibition) to have more clarity.

Agreed solution:

SWS ComM
==========

~[SWS_ComM_00163]
Change description of Return value E_NOT_OK: Changed of inhibition status for the channel failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

~[SWS_ComM_00124]
Change description of Return value E_NOT_OK: Changed of inhibition status for the ECU failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

~[SWS_ComM_00156]
Change description of Return value E_NOT_OK: Changed of wake up status for the channel failed, e.g. ComMEcuGroupClassification disable the functionality (see ECUC_ComM_00563)

+add notes in sec. 7.3.1 Communication inhibition
[...]
Note: 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
[...]
–Last change on issue 76916 comment 5–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.5 Specification Item SWS_ComM_00215

Trace References:

none

Content:

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

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP

– replace in text below 7.1.4.1 Active PNC Gateway
–Last change on issue 77808 comment 1–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.6 Specification Item SWS_ComM_00234

Trace References:

SRS_BSW_00323, SRS_BSW_00327, SRS_BSW_00337, SRS_BSW_00385, SRS_BSW_00386

Content:

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

Type or error	Relevance	Related error code	Value [hex]
API service used without module initialization	Development	COMM_E_NOT_INITED UNINIT	0x1
API service used with wrong parameters	Development	COMM_E_WRONG_PARAMETERS	0x2
API Service used with a null pointer	Development	COMM_E_PARAM_POINTER	0x3
Initialization failed	Development	COMM_E_INIT_FAILED	0x4

Table [REF]: Error classification

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #59085: Rollout of 'Runtime errors'

  Problem description:

  Inconsistencies in SWS with semantics of Default errors
  –Last change on issue 59085 comment 26–

  Agreed solution:

  solution in Column "G" of the new attachment https://www.autosar.org/bugzilla/attachment.cgi?id=4604

  Notes:
  - It is not enough just to migrate the error from one classification table to another.

Please also check the related requirements (and background information) which is referring to that error and adapt them if needed.
- The review task of the ITs shall be done by the WP to which the specification "belongs".

*** BSW UML Model ***
SWS_CanNm:
———-
Chapter 8.6.1 Optional Interfaces:
Add within SWS_CanNm_00325 the API function Det_ReportRunTimeError

SWS_LinIf:
———-
SWS_LinIf_00359: add Det_ReportRuntimeError

SWS_UdpNm:
———-
Replace UDPNM_E_NO_INIT with UDPNM_E_UNINIT in description of API UdpNm_MainFunction_<Instance Id> (SWS_UdpNm_00234)

*** ECUC XML ***
Not affected. No configuration of runtime error reporting required (see SWS BSW General).
–Last change on issue 59085 comment 88–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.7 Specification Item SWS_ComM_00612

Trace References:

SRS_BSW_00406

Content:

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 Com M_GetStatus() (see SWS_COMM_00242); shall:

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

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."
  - "module raises the Default error XXX_E_TRANSITION"
  - "The DET provides services to store default errors"

...

The correct text would be:
- sub chapter is called "7.x Development errors"
- "[SWS_xxx_yyyyy]
In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"

- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:

- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer" –Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.8 Specification Item SWS_ComM_00665

Trace References:

none

Content:

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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "
  [SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
  "

  Agreed solution:

  === ComM ===
  ~7.2 ComM channel state machine
  extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to

COMM_NO_COM_REQUEST_PENDING:
ComM_EcuM_PNCWakeUpIndication(PNC)or
ComM_EcuM_WakeUpIndication(ChX) or
ComM_Nm_RestartIndication(ChX) or
ComM_Nm_NetworkStartIndication(ChX)

info box below Figure 4:
+ PNC - PNC Id

~Table 1
State: COMM_NO_COMMUNICATION
Section / Requirement:
In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.9 Specification Item SWS_ComM_00694

Trace References:

none

Content:

In If ComM_EcuM_WakeUpIndication is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter Com MSynchronousWakeUp=TRUE (ECUC_ComM_00695)and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.(SRS_ModeMgm_09248)

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "
  [SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state

machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
"

Agreed solution:

=== ComM ===
~7.2 ComM channel state machine
extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
ComM_EcuM_PNCWakeUpIndication(PNC)or
ComM_EcuM_WakeUpIndication(ChX) or
ComM_Nm_RestartIndication(ChX) or
ComM_Nm_NetworkStartIndication(ChX)

info box below Figure 4:
+ PNC - PNC Id

~Table 1
State: COMM_NO_COMMUNICATION
Section / Requirement:
In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

~[SWS_ComM_00893]If ComM_EcuM_WakeUpIndication is called in sub-state

COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComM-SynchronousWakeUp=FALSE (ECUC_ComM_00695), the ComM module shall switch the requested ComM channel state machine (resp. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.10 Specification Item SWS_ComM_00805

Trace References:

none

Content:

Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

original finding was:

According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
"
...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
Such requirment are not implementation relevant. Please remove this and add as note.
–Last change on issue 75637 comment 2–

Agreed solution:

General:
- Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
- Add general requirement as note to ch. 8.3 Function definitions
- Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

=== EhtIf ===
~8.3 Function definitions
+Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
EthIf_Init, EthIf_GetVersionInfo

~8.3.2 EthIf_SetControllerMode
-[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to

Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()

-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition

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

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.11 Specification Item SWS_ComM_00806

Trace References:

none

Content:

Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

  original finding was:

  According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
  "
  ...
  [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  "
  Such requirment are not implementation relevant. Please remove this and add as note.
  –Last change on issue 75637 comment 2–

  Agreed solution:

  General:
  - Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  - Add general requirement as note to ch. 8.3 Function definitions
  - Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

  === EhtIf ===

~8.3 Function definitions
+Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
EthIf_Init, EthIf_GetVersionInfo

~8.3.2 EthIf_SetControllerMode
-[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334

-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (inter-

rupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."

"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.12 Specification Item SWS_ComM_00808

Trace References:

none

Content:

Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

  original finding was:

  According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
  "
  ...
  [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()

"
Such requirment are not implementation relevant. Please remove this and add as note.
–Last change on issue 75637 comment 2–

Agreed solution:

General:
- Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
- Add general requirement as note to ch. 8.3 Function definitions
- Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

=== EhtIf ===
~8.3 Function definitions
+Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
EthIf_Init, EthIf_GetVersionInfo

~8.3.2 EthIf_SetControllerMode
-[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable

-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just

passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()

-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM

module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.13 Specification Item SWS_ComM_00810

Trace References:

none

Content:

Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

  original finding was:

  According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
  "
  ...
  [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  "
  Such requirment are not implementation relevant. Please remove this and add as note.
  –Last change on issue 75637 comment 2–

  Agreed solution:

  General:
  - Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  - Add general requirement as note to ch. 8.3 Function definitions
  - Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

  === EhtIf ===
  ~8.3 Function definitions
  +Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
  EthIf_Init, EthIf_GetVersionInfo

  ~8.3.2 EthIf_SetControllerMode
  -[SWS_EthIf_00038]

  ~8.3.3 EthIf_GetControllerMode
  -[SWS_EthIf_00044]

  ~8.3.8 EthIf_ CheckWakeup

-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1

If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."

"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.14 Specification Item SWS_ComM_00812

Trace References:

none

Content:

Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

  original finding was:

  According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
  "
  ...
  [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  "
  Such requirment are not implementation relevant. Please remove this and add as note.
  –Last change on issue 75637 comment 2–

  Agreed solution:

  General:
  - Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  - Add general requirement as note to ch. 8.3 Function definitions
  - Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

=== EhtIf ===
~8.3 Function definitions
+Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
EthIf_Init, EthIf_GetVersionInfo

~8.3.2 EthIf_SetControllerMode
-[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with tem-

plate <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()

-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.15 Specification Item SWS_ComM_00814

Trace References:

none

Content:

Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

  original finding was:

  According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
  "

...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
Such requirment are not implementation relevant. Please remove this and add as note.
–Last change on issue 75637 comment 2–

Agreed solution:

General:
- Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
- Add general requirement as note to ch. 8.3 Function definitions
- Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

=== EhtIf ===
~8.3 Function definitions
+Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
EthIf_Init, EthIf_GetVersionInfo

~8.3.2 EthIf_SetControllerMode
-[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr

-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===

General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()

-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition

+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.16 Specification Item SWS_ComM_00816

Trace References:

none

Content:

Caveats of ComM_BusSM_ModeIndication(...): The Communication Manager Module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

  Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

  original finding was:

  According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
  "
  ...
  [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  "
  Such requirment are not implementation relevant. Please remove this and add as note.
  –Last change on issue 75637 comment 2–

  Agreed solution:

  General:
  - Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
  Caveat: The function requires previous initialization (EthIf_Init). ?()
  - Add general requirement as note to ch. 8.3 Function definitions
  - Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

  === EhtIf ===
  ~8.3 Function definitions
  +Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
  EthIf_Init, EthIf_GetVersionInfo

  ~8.3.2 EthIf_SetControllerMode
  -[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not

execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition

+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.17 Specification Item SWS_ComM_00825

Trace References:

none

Content:

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

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

bitIndex=(ComMPncId mod 8)

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #72284: [ComM] pncVectorOffset is not a configuration parameter of the module

  Problem description:

  SWS_ComM_00825 refers to pncVectorOffset, however this is a System Template attribute and not a configuration parameter of the module.

  A solution similar to how CanNm, FrNm and UdpNm are specified is needed, i.e.
  - Introduce ComMPncVectorOffset and ComMPncVectorLength parameters to ComMConfigSet
  - Update the System Template Appendix D to derive these parameters from SystemTemplate::System.pncVectorOffset and SystemTemplate::System.pncVectorLength repsectively
  - Update SWS_ComM_00825 to refer to ComMPncVectorOffset

  Agreed solution:

  - Update SWS_ComM_00825 as follows:
  The byteIndex and bitIndex, in which a bit corresponding to one ComMPncId resides, shall be determined as follows:

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

- Add the following hint below SWS_ComM_00825
Hint: The value of <PNC Vector Offset> (and <PNC Vector Length> if needed) can be obtained from the <Bus> Network Management modules configuration

- Correct the SWS ID in the comment following SWS_ComM_00825 from "ComM825 defines only ... " to "SWS_ComM_00825 defines only ... "
–Last change on issue 72284 comment 14–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.18 Specification Item SWS_ComM_00842

Trace References:

none

Content:

The ComM module shall ignore requests for limit to COMM_NO_COMMUNICATION in other states than COMM_FULL_COM_NETWORK_REQUESTED.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73693: Clarification regarding WakeUp inhibition and LimitToNoCom

  Problem description:

  Hello,

  1) In case wake-up inhibition is used, only the transition from COMM_NO_COM_NO_PENDING to COMM_NO_COM_REQUEST_PENDING shall be inhibited ?
  The transition from COMM_NO_COM_REQUEST_PENDING to COMM_FULL_COMMUNICATION is allowed ?
  Scenario:
  - SM is in state COMM_NO_COM_NO_PENDING_REQUEST
  - Communication allowed is false
  - user requests full com
  - mainfunction is called (SM goes to COMM_NO_REQUEST_PENDING)

- Wakeup inhibition is called
- mainfunction
- communication allowed is called (SM goes to COMM_FULL_COMMUNICATION)

2) In case limit to no com is used, this functionality is used only in case SM is in Full Communication to bring back the SM to No communication or also it shall inhibit the wakeup ?

3) Regarding requirement SWS_ComM_00842 by ignoring requests means user requests or limit(channel/ecu)tonocom requests ?

Best Regards,
Lorant

Agreed solution:

~[SWS_ComM_00842] ?The ComM module shall ignore requests for limit to COMM_NO_COMMUNICATION in other states than COMM_FULL_COM_NETWORK_REQUESTED.?()
–Last change on issue 73693 comment 11–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.19 Specification Item SWS_ComM_00858

Trace References:

SRS_BSW_00406

Content:

If ComM is not initialized and default error detection has been switched on development error detection is enabled by ComMDevErrorDetect (see ECUC_ComM_00555), the : the function shall check that the service ComMmodule shall report a _Init was previously called. If the check fails, the function shall raise the development error COMM_E_NOT_INITED (by using the Det_ReportError service of the Default Error Tracer module) for all ComM module API services other than ComM_Init() and Com M_GetVersionInfo(), and ComM_GetStatus()UNINIT otherwise (if DET is disabled) return E_NOT_OK.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #73570: No "default error" in AUTOSAR

  Problem description:

  The DET was renamed from development error tracer to default error tracer.

  This change was most of the time done automatically and unfortunately renamed "developement error" to "default error".

  "default error" should always be followed by "tracer", otherwise, "development error" is probably the right term.

  This could increase the impact (compared to my selection of impacted document, but formally, the configuration parameters *DevErrorDetect are not using the correct description:
  "Switches the Default Error Tracer (Det) detection and notification..."

  The parameter switches on/off the developement error detection. The DET does not need to be detected and can be present even when the parameter is set to false.

  Agreed solution:

  Rename "default error" to "development error" in all impacted documents, but not in an automated way (Do not change "default error tracer" to "developement error tracer"!)

  Blueprint/Example:
  - sub chapter is now called "7.x Default errors"
  - "[SWS_xxx_yyyyy]
  In case default error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected errors to the DET. ()"
  - "[SWS_xxx_yyyyy]
  If default error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the default error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
  - "In case default errors are enabled,..."
  - "module raises the Default error XXX_E_TRANSITION"
  - "The DET provides services to store default errors"
  ...

  The correct text would be:
  - sub chapter is called "7.x Development errors"
  - "[SWS_xxx_yyyyy]

In case development error detection is enabled for the xxxx module: The xxxx module shall check API parameters for validity and report detected development errors to the DET. ()"
- "[SWS_xxx_yyyyy]
If development error detection is enabled: the function shall check that the service xxx_Init was previously called. If the check fails, the function shall raise the development error XXX_E_NOT_INITIALIZED otherwise (if DET is disabled) return E_NOT_OK. ()"
- "In case development errors are enabled,..."
- "module raises the development error XXX_E_TRANSITION"
- "The DET provides services to store development errors"

Solution for SWS_RTE:
– SWS_RTE —
- Change 4.8 Default errors to 4.8 Development errors
- Change "Errors which can occur at runtime in the RTE are classified as default errors" to "Errors which can occur at runtime in the RTE are classified as development errors"
- Remove [SWS_Rte_07676]
- Change [SWS_RTE_06611]"If a violation is detected the RTE shall report a default error to the DET." to "If a violation is detected the RTE shall report a development error to the DET."
- Change [SWS_Rte_06631]
[SWS_Rte_06631] d The RTE shall use the OS Application Identifier as the Instance Id to enable the developer to identify in which runtime section of the RTE the error occurs. This Instance ID is even unique across multi cores and so implicitly allows the development error to be traced to a specific core. c(SRS_BSW_00337)

SRS_Libraries:
- In chapter "3 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_SPALGeneral:
- In chapter "6.1.1.3.1 [SRS_SPAL_00157] ...": Rename "Development Error Tracer" to "Default Error Tracer"
- In chapter "6.1.1.4.2 [SRS_SPAL_12448] ...": Rename "Development Error Tracer" to "Default Error Tracer"

SRS_FlashTest:
- In chapter "6.1 Functional Requirements": Rename "Development Error Tracer" to

"Default Error Tracer"
- In chapter "7 References":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_MFXLibrary:
- In chapter "2 Acronyms and abbreviations": Rename "Development Error Tracer" to "Default Error Tracer"

SWS_MemoryAbstractionInterface:
- In chapter "3.1 Input documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_FlexRayNetworkManagement:
- In chapter "3.3 Related AUTOSAR documents":
Rename "Development Error Tracer" to "Default Error Tracer"
Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_CANStateManager:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_PDURouter:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"

SWS_EEPROMDriver:
- In chapter "3.1 Input documents": Rename "AUTOSAR_SWS_DevelopmentErrorTracer" to "AUTOSAR_SWS_DefaultErrorTracer"
–Last change on issue 73570 comment 47–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.20 Specification Item SWS_ComM_00890

Trace References:

none

Content:

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

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #71673: [ComM]: Handling of ComMTMinFullComModeDuration timer during Mode Inhibition (LimitChannelToNoComMode)

  Problem description:

  1. When the NmVariant= Light/None, on entering COMM_FULL_COM_NETWORK_REQUESTED, ComMTMinFullComModeDuration is started. Before the expiry of this timer, application indicates mode inhibition by calling ComM_LimitChannelToNoComMode. In this case whether ComM has to cancel the timer and enter COMM_FULL_COM_READY_SLEEP and start the NmLightTimeout timer or ComM has to wait in COMM_FULL_COM_NETWORK_REQUESTED till the expiry of this ComMTMinFullComModeDuration timer.

  2. When the NmVariant= Light/None, on entering COMM_FULL_COM_READY_SLEEP due to mode limitation (ComM_LimitChannelToNoComMode), if the application disables the mode limitation and gives a user request, should ComM enter COMM_FULL_COM_NETWORK_REQUESTED? IN case ComM enters COMM_FULL_COM_NETWORK_REQUESTED again, then should the timer ComMTMinFullComModeDuration be loaded again and started?

  Clarification needed.
  –Last change on issue 71673 comment 18–

Agreed solution:

~[SWS_ComM_00890] In sub-state COMM_FULL_COM_NETWORK_REQUESTED and the DCM does not indicate ComM_DCM_ActiveDiagnostic(<channel>)(SWS_ComM_00873) and communication limitation is requested (see section 7.3.1), ComM channel state machine shall immediately switch to sub-state COMM_FULL_COM_READY_SLEEP and cancel the timer for ComMTMinFull-ComModeDuration .(SRS_ModeMgm_09071)
–Last change on issue 71673 comment 14–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.21 Specification Item SWS_ComM_00893

Trace References:

SRS_ModeMgm_09087

Content:

In If ComM_EcuM_WakeUpIndication is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and a wake-up-indication is indicated by the EcuM module, ComM_EcuM_configuration parameter ComMSynchronousWake UpIndication() SWS=FALSE (ECUC_ComM_0027500695), the ComM module shall switch the requested ComM channel state machine shall immediately switch (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "

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

machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
"

Agreed solution:

=== ComM ===
~7.2 ComM channel state machine
extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
ComM_EcuM_PNCWakeUpIndication(PNC)or
ComM_EcuM_WakeUpIndication(ChX) or
ComM_Nm_RestartIndication(ChX) or
ComM_Nm_NetworkStartIndication(ChX)

info box below Figure 4:
+ PNC - PNC Id

~Table 1
State: COMM_NO_COMMUNICATION
Section / Requirement:
In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

~[SWS_ComM_00893]If ComM_EcuM_WakeUpIndication is called in sub-state

COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComM-SynchronousWakeUp=FALSE (ECUC_ComM_00695), the ComM module shall switch the requested ComM channel state machine (resp. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.22 Specification Item SWS_ComM_00910

Trace References:

none

Content:

PNC functionality shall only exist if the parameter ComMPncSupport is set to TRUE. (see SWSECUC_ComM_00839_Conf).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

Agreed solution:

- Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
- Additional:
– replace in text below SWS_ComM_00966
– replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
– replace in text below 7.1.4.1 Active PNC Gateway
–Last change on issue 77808 comment 1–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.23 Specification Item SWS_ComM_00911

Trace References:

none

Content:

Enabling or disabling of the PNC functionality shall be post-build configurable using the parameter ComMPncEnabled (see SWSECUC_ComM_00878_Conf).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "
  [SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
  "

  Agreed solution:

=== ComM ===
~7.2 ComM channel state machine
extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
ComM_EcuM_PNCWakeUpIndication(PNC)or
ComM_EcuM_WakeUpIndication(ChX) or
ComM_Nm_RestartIndication(ChX) or
ComM_Nm_NetworkStartIndication(ChX)

info box below Figure 4:
+ PNC - PNC Id

~Table 1
State: COMM_NO_COMMUNICATION
Section / Requirement:
In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.24 Specification Item SWS_ComM_00912

Trace References:

none

Content:

It shall be possible to map a configurable amount of ComMUsers to one or more PNCs using the parameter ComMUserPerPnc (see SWSECUC_ComM_00876_Conf).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.25 Specification Item SWS_ComM_00913

Trace References:

none

Content:

It shall be possible to map a configurable amount of PNC(s) to a configurable amount of ComM channels using the parameter ComMChannelPerPnc (see SWSECUC_ComM_00880_Conf).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.26 Specification Item SWS_ComM_00919

Trace References:

none

Content:

It shall be possible to assign more than one COM signal containing bits representing the PNC to one PNC using the configuration container ComMPncComSignal (see SWSECUC_ComM_00881_Conf).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.27 Specification Item SWS_ComM_00931

Trace References:

none

Content:

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 ComM695ECUC_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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

Agreed solution:

- Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
- Additional:
– replace in text below SWS_ComM_00966
– replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
– replace in text below 7.1.4.1 Active PNC Gateway
–Last change on issue 77808 comment 1–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.28 Specification Item SWS_ComM_00934

Trace References:

none

Content:

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 (SWSECUC_ComM_00840_Conf) equals TRUE.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

- Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
- Additional:
– replace in text below SWS_ComM_00966
– replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
– replace in text below 7.1.4.1 Active PNC Gateway
–Last change on issue 77808 comment 1–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.29 Specification Item SWS_ComM_00945

Trace References:

none

Content:

If the configuration parameter ComMPncGatewayEnabled (see SWSECUC_ComM_00840_Conf) is true and the parameter ComMPncGatewayType is set to COMM_GATEWAY_TYPE_ACTIVE for a ComMChannel and at least one bit corresponding to the PNC within the Rx bitvectors with signal type "ERA" equals ’1’, then the bit corresponding to this PNC within ERA in ComM shall be set to ’1’.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state

COMM_PNC_PREPARE_SLEEP
– replace in text below 7.1.4.1 Active PNC Gateway
–Last change on issue 77808 comment 1–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.30 Specification Item SWS_ComM_00947

Trace References:

none

Content:

When the timer ComMPncPrepareSleepTimer (see SWSECUC_ComM_00841_Conf) expires, the PNC sub state COMM_PNC_PREPARE_SLEEP shall be left and the PNC main state COMM_PNC_NO_COMMUNICATION shall be entered.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.31 Specification Item SWS_ComM_00952

Trace References:

none

Content:

If the sub state COMM_PNC_PREPARE_SLEEP is entered, the timer ComMPncPrepare SleepTimer (see SWSECUC_ComM_00841_Conf) shall be started with the configured initial value.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.32 Specification Item SWS_ComM_00953

Trace References:

none

Content:

If the PNC functionality is enabled using the configuration parameter ComMPncEnabled set to TRUE (see SWSECUC_ComM_00878_Conf), all actions related to PNC changes shall be executed before the channel related actions (channel related actions, see Chapter 7.3).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "
  [SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
  "

  Agreed solution:

  === ComM ===
  ~7.2 ComM channel state machine
  extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
  ComM_EcuM_PNCWakeUpIndication(PNC)or
  ComM_EcuM_WakeUpIndication(ChX) or
  ComM_Nm_RestartIndication(ChX) or
  ComM_Nm_NetworkStartIndication(ChX)

  info box below Figure 4:
  + PNC - PNC Id

  ~Table 1
  State: COMM_NO_COMMUNICATION
  Section / Requirement:
  In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

  Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
  Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

Problem description:

The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

Agreed solution:

- Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
- Additional:
– replace in text below SWS_ComM_00966
– replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
– replace in text below 7.1.4.1 Active PNC Gateway
–Last change on issue 77808 comment 1–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.33 Specification Item SWS_ComM_00964

Trace References:

none

Content:

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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "
  [SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST

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

Agreed solution:

=== ComM ===
~7.2 ComM channel state machine
extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
ComM_EcuM_PNCWakeUpIndication(PNC)or
ComM_EcuM_WakeUpIndication(ChX) or
ComM_Nm_RestartIndication(ChX) or
ComM_Nm_NetworkStartIndication(ChX)

info box below Figure 4:
+ PNC - PNC Id

~Table 1
State: COMM_NO_COMMUNICATION
Section / Requirement:
In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

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

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.34 Specification Item SWS_ComM_00965

Trace References:

none

Content:

Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #75637: [diverse] Remove requirements like initializiation requirements regarding the <Module>_Init API

  Problem description:

Generally a Caveat can never start a requirement. For this all basic Software module shall be checked and cleaned up for this formulation.

original finding was:

According to the discussion of RfC# 67678 we detect requirements regarding the initializiation of the certain api's, e.g.:
"
...
[SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
"
Such requirment are not implementation relevant. Please remove this and add as note.
–Last change on issue 75637 comment 2–

Agreed solution:

General:
- Remove requirements which state that previous initialization of <Modul>_Init() is required, which are formulated like a note, e.g. : [SWS_EthIf_00213] ?
Caveat: The function requires previous initialization (EthIf_Init). ?()
- Add general requirement as note to ch. 8.3 Function definitions
- Remove or reformulate all other requirements, which start with caveat. Most of them are simply a note or implementation hint.

=== EhtIf ===
~8.3 Function definitions
+Note: All functions in this chapter requires previous initialization (EthIf_Init), except the following ones:
EthIf_Init, EthIf_GetVersionInfo

~8.3.2 EthIf_SetControllerMode
-[SWS_EthIf_00038]

~8.3.3 EthIf_GetControllerMode
-[SWS_EthIf_00044]

~8.3.8 EthIf_ CheckWakeup
-[SWS_EthIf_00249]

~8.3.9 EthIf_GetPhysAddr
-[SWS_EthIf_00066]

~8.3.10 EthIf_SetPhysAddr
-[SWS_EthIf_00138]

~8.3.11 EthIf_UpdatePhysAddrFilter
-[SWS_EthIf_00144]

~8.3.12 EthIf_GetPortMacAddr
-[SWS_EthIf_00195]

~8.3.13 EthIf_GetArlTable
-[SWS_EthIf_00201]

8.3.14 EthIf_GetBufferLevel
-[SWS_EthIf_00207]

~8.3.15 EthIf_GetDropCount
-[SWS_EthIf_00213]

~8.3.16 EthIf_StoreConfiguration
-[SWS_EthIf_00218]

~8.3.17 EthIf_ResetConfiguration
-[SWS_EthIf_00223]

8.3.18 EthIf_GetCurrentTime
-[SWS_EthIf_00159]

~8.3.19 EthIf_EnableEgressTimeStamp
-[SWS_EthIf_00165]

~8.3.20 EthIf_GetEgressTimeStamp
-[SWS_EthIf_00171]

~8.3.21 EthIf_GetIngressTimeStamp
-[SWS_EthIf_00177]

~8.3.22 EthIf_SetCorrectionTime
-[SWS_EthIf_00183]

~8.3.23 EthIf_SetGlobalTime
-[SWS_EthIf_00189]

~8.3.24 EthIf_ProvideTxBuffer
-[SWS_EthIf_00074]

~8.3.25 EthIf_Transmit
-[SWS_EthIf_00081]

=== CanIf ===
General note for below items which contain phrases like "The CAN Interface module must be initialized after Power ON": this is not needed anymore cause already stated in SWS_CANIF_00661. So e.g. in cases where result from CanDrv is just passed through the whole requirement can be deleted.

Also it is assumed, that the mindset like defined in SWS_CANIF_00661 is there for all upper layer functions like <User_TxConfirmation> which might be called by CanIf.

Because if CanDrv would not be already initialised it would return E_NOT_OK which is passed through:
-SWS_CANIF_00312
-SWS_CANIF_00316

Because already guarded by SWS_CANIF_00661:
-SWS_CANIF_00334
-SWS_CANIF_00339
-SWS_CANIF_00344
-SWS_CANIF_00349

~SWS_CANIF_00661
All CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall not execute their normal operation and return E_NOT_OK unless the CanIf has been initialized with a preceding call of CanIf_Init().

+SWS_CANIF_????1
If the switch CANIF_PUBLIC_DEV_ERROR_DETECT is enabled, all CanIf API services other than CanIf_Init() and CanIf_GetVersionInfo() shall report to the DET (using CANIF_E_UNINIT) unless the CanIf has been initialized with a preceding call of CanIf_Init().

-SWS_CANIF_00323 change to
Note: During the call of CanIf_Transmit() the buffer of PduInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00329
CanIf_ReadRxPduData() shall not be used for CanIfRxSduId, which are defined to receive multiple CAN-Ids (range reception).

Add note below SWS_CANIF_00329
Note: During the call of of CanIf_ReadRxPduData() the buffer of CanIfRxInfoPtr is controlled by CanIf and this buffer should not be accessed for read/write from another call context. After return of this call the ownership changes to the upper layer.

~SWS_CANIF_00356
CanIf_SetDynamicTxId() shall not be interrupted by CanIf_Transmit(), if the same L-SDU ID is handled.

-SWS_CANIF_00407 change to
Note: The call context of CanIf_CheckValidation() is either on interrupt level (interrupt mode) or on task level (polling mode).
Caveat: The corresponding CAN controller and transceiver must be switched on via CanTrcv_SetOpMode(Transceiver, CANTRCV_TRCVMODE_NORMAL) and Can_SetControllerMode(Controller, CAN_CS_STARTED) and the corresponding mode indications must have been called.

Remove all following requirements and instead add there a Note with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
-SWS_CANIF_00737 CanIf_GetTxConfirmationState()
-SWS_CANIF_00870 CanIf_SetBaudrate()

Remove all following requirements and instead add there a Note with template <FUNC>

"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
-SWS_CANIF_00401 CanIf_CheckWakeup()
-SWS_CANIF_00413 CanIf_TxConfirmation()
-SWS_CANIF_00422 CanIf_RxIndication()
-SWS_CANIF_00432 CanIf_ControllerBusOff()
-SWS_CANIF_00818 CanIf_ConfirmPnAvailability()
-SWS_CANIF_00807 CanIf_ClearTrcvWufFlagIndication()
-SWS_CANIF_00811 CanIf_CheckTrcvWakeFlagIndication()
-SWS_CANIF_00703 CanIf_ControllerModeIndication()
-SWS_CANIF_00709 CanIf_TrcvModeIndication()
-SWS_CANIF_00887 <User_TriggerTransmit>()
-SWS_CANIF_00437 <User_TxConfirmation>()

-SWS_CANIF_00440 change to
Note: Until <User_RxIndication>() returns, CanIf will not access PduInfoPtr. The PduInfoPtr is only valid and can be used by upper layers, until the indication returns. CanIf guarantees that the number of configured bytes for this PduInfoPtr is valid.
Note: The call context of <User_RxIndication>() is either on interrupt level (interrupt mode) or on task level (polling mode).

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is either on interrupt level (interrupt mode) or on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00455 <User_ValidateWakeupEvent>()
-SWS_CANIF_00822 <User_ConfirmPnAvailability>()
-SWS_CANIF_00793 <User_ClearTrcvWufFlagIndication>()
-SWS_CANIF_00799 <User_CheckTrcvWakeFlagIndication>()

Remove all following requirements and instead add there two Notes with template <FUNC>
"Note: The call context of <FUNC> is on task level (polling mode)."
"Note: The callback service <FUNC> is in general re-entrant for multiple CAN Controller usage, but not for the same CAN Controller"
-SWS_CANIF_00688 <User_ControllerModeIndication>()

=== ComM ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (ComM_Init), except the following ones:
ComM_Init, ComM_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the ComM module is initialized correctly.
-[SWS_ComM_00805] Caveats of ComM_Nm_NetworkStartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00806] Caveats of ComM_Nm_NetworkMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00808] Caveats of ComM_Nm_PrepareBusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00810] Caveats of ComM_Nm_BusSleepMode: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00812] Caveats of ComM_Nm_RestartIndication: The ComM module is initialized correctly.()
-[SWS_ComM_00814] Caveats of ComM_EcuM_WakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00965] Caveats of ComM_EcuM_PNCWakeUpIndication: The Communication Manager Module is initialized correctly.()
-[SWS_ComM_00816] Caveats of ComM_BusSM_ModeIndication(): The Communication Manager Module is initialized correctly.()

=== LdCom ===
~ ch. 8.3 Function definition
+Note: All functions in this chapter requires previous initialization (LdCom_Init), except the following ones:
LdCom_Init, LdCom_GetVersionInfo

~ ch. 8.4 Callback definition
+Note: All functions in this chapter requires that the LdCom module is initialized correctly.
–Last change on issue 75637 comment 23–

BW-C-Level:

Application	Specification	Bus
1	1	1

1.35 Specification Item SWS_ComM_00979

Trace References:

none

Content:

If the optional PNC functionality is enabled (see SWSECUC_ComM_00839_Conf00883), all PNC actions shall be performed before the channel related actions are executed.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.36 Specification Item SWS_ComM_00980

Trace References:

none

Content:

If the parameter ComMPncNmRequest equals TRUE (see SWSECUC_ComM_00886_conf), if the "FULL Communication" is requested due to a change in the PNC state machine to COMM_PNC_REQUESTED (see SWS_ComMCOMM_00993) API Nm_NetworkRequest() shall be called, even if the current state is already "Full communication".

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.37 Specification Item SWS_ComM_00981

Trace References:

none

Content:

If the configuration parameter ComMPncGatewayEnabled (see SWSECUC_ComM_00840_Conf) is TRUE, the default gateway type shall be active (COMM_GATEWAY_TYPE_ACTIVE).

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77808: [ComM] Replace outdated references to SWS_ComM_00xxx_Conf (now ECUC_ComM_00xxx)

  Problem description:

  The SWS ComM still contains loads of references to SWS_ComM_00xxx_Conf (very old tag for EcuC parameters). These shall be replaced by references to ECUC_ComM_00xxx.

  Agreed solution:

  - Replace SWS_ComM_00xxx_Conf by ECUC_ComM_00xxx used in certain requirement (see affected spec items)
  - Additional:
  – replace in text below SWS_ComM_00966
  – replace in text below 7.1.3.9 Behavior in PNC sub state COMM_PNC_PREPARE_SLEEP
  – replace in text below 7.1.4.1 Active PNC Gateway
  –Last change on issue 77808 comment 1–

  BW-C-Level:

Application	Specification	Bus
1	1	1

1.38 Specification Item SWS_ComM_01014

Trace References:

SRS_ModeMgm_09248

Content:

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

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
"
[SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
"

Agreed solution:

=== ComM ===
~7.2 ComM channel state machine
extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
ComM_EcuM_PNCWakeUpIndication(PNC)or
ComM_EcuM_WakeUpIndication(ChX) or
ComM_Nm_RestartIndication(ChX) or
ComM_Nm_NetworkStartIndication(ChX)

info box below Figure 4:
+ PNC - PNC Id

~Table 1
State: COMM_NO_COMMUNICATION
Section / Requirement:
In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
+[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

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

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1

1.39 Specification Item SWS_ComM_01015

Trace References:

SRS_ModeMgm_09248

Content:

If ComM_EcuM_PNCWakeUpIndication(<PNC>) is called in sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameters Com MSynchronousWakeUp=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.

RfCs affecting this spec item between releases 4.3.0 and 4.3.1:

- RfC #77667: [ComM] Handling of ComM channel state machine if PNC wake-up-indication is triggered

  Problem description:

  Handling of ComM channel state machine is missing in case a PNC wake-up-indication is triggered. A similar requirement as for wake-up-indication is needed:
  "
  [SWS_ComM_00694] In sub-state COMM_NO_COM_NO_PENDING_REQUEST and configuration parameter ComMSynchronousWakeUp=TRUE (ECUC_ComM_00695) and a wake-up-indication of a channel is indicated by the EcuM, the ComM module shall immediately switch all ComM channel state machines (resp. channels) to sub-state COMM_NO_COM_REQUEST_PENDING.
  "

  Agreed solution:

  === ComM ===
  ~7.2 ComM channel state machine
  extend the transitions in the ComM channel statemachine (Figure 4) from COMM_NO_COM_NO_PENDING_REQUEST to COMM_NO_COM_REQUEST_PENDING:
  ComM_EcuM_PNCWakeUpIndication(PNC)or
  ComM_EcuM_WakeUpIndication(ChX) or
  ComM_Nm_RestartIndication(ChX) or
  ComM_Nm_NetworkStartIndication(ChX)

  info box below Figure 4:
  + PNC - PNC Id

  ~Table 1
  State: COMM_NO_COMMUNICATION
  Section / Requirement:
  In sub-state COMM_NO_COM_NO_PENDING_REQUEST: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

  Transition: COMM_NO_COMMUNICATION -> COMM_FULL_COMMUNICATION
  Requirement: +SWS_ComM_xxxxx1, +SWS_ComM_xxxxx2

  ~7.2.1.1 COMM_NO_COM_NO_PENDING_REQUEST sub-state
  +[SWS_ComM_xxxxx1]If ComM_EcuM_PNCWakeUpIndication 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.

+[SWS_ComM_xxxxx2]If ComM_EcuM_PNCWakeUpIndication 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.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~[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. channel) to sub-state COMM_NO_COM_REQUEST_PENDING.

~7.2.3.1 COMM_FULL_COM_NETWORK_REQUESTED sub-state
~[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() (SWS_ComM_00275) or by ComM_EcuM_PNCWakeUpIndication() (SWS_ComM_91001), the ComM module shall request Nm_PassiveStartup() from the Network Management.
–Last change on issue 77667 comment 26–

BW-C-Level:

Application	Specification	Bus
1	4	1